Accepted Manuscript

Scalable Image Segmentation via Decoupled Sub-graph Compression

R.S. Medeiros, A. Wong, J. Scharcanski

PII: S0031-3203(17)30482-X
DOI: 10.1016/j.patcog.2017.11.028
Reference: PR 6381

To appear in: *Pattern Recognition*

Received date: 17 January 2017
Revised date: 21 August 2017
Accepted date: 30 November 2017

Please cite this article as: R.S. Medeiros, A. Wong, J. Scharcanski, Scalable Image Segmentation via Decoupled Sub-graph Compression, *Pattern Recognition* (2017), doi: 10.1016/j.patcog.2017.11.028

Highlights

- A scalable graph compression algorithm for image segmentation proposed.
- The input image is represented by a region graph model.
- Texton dictionaries capture the local texture features in decoupled sub-graphs.
- A graph compression algorithm reduces the graph size and segments the image.
- Local graph decoupling and recoupling operations lead to an efficient method.

Scalable Image Segmentation via Decoupled Sub-graph Compression

R. S. Medeiros[a,*], A. Wong[b], J. Scharcanski[a]

[a] *Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, RS, Brasil, 91501-970*
[b] *Department of Systems Design Engineering, University of Waterloo, Waterloo, Canada N2L 3G1*

Abstract

Dealing with large images is an on-going challenge in image segmentation, where many of the current methods run into computational and/or memory complexity issues. This work presents a novel decoupled sub-graph compression (DSC) approach for efficient and scalable image segmentation. In DSC, the image is modeled as a region graph, which is then decoupled into small sub-graphs. The sub-graphs undergo a compression process, which simplifies the graph, reducing the number of vertices and edges, while keeping the overall graph structure. Finally, the compressed sub-graphs are re-coupled and re-compressed to form a final compressed graph representing the final image segmentation. Experimental results based on a dataset of high resolution images (1000×1500) show that the DSC method achieves better segmentation performance when compared to state-of-the-art segmentation methods (PRI=0.84 and F=0.61), while having significantly lower computational and memory complexity.

Keywords: Segmentation, graph compression, decoupling, scalability.

1. Introduction

Image segmentation is a challenging problem where the goal is to partition an image into disjoint segments containing pixels that share similar character-

*Corresponding author
Email addresses: rsmedeiros@inf.ufrgs.br (R. S. Medeiros), a28wong@uwaterloo.ca (A. Wong), jacobs@inf.ufrgs.br (J. Scharcanski)

istics. This becomes more challenging when dealing with images containing complex textures, which usually demand more accurate models and representations. Furthermore, it is particularly challenging to segment large images, which is becoming especially important with the recent prevalence of digital cameras capable of megapixel resolutions (e.g., 15MP $\approx 4800 \times 3200$). Addressing the problem associated with large images with complex textures efficiently is crucial to all sorts of real-world applications such as object detection and recognition (faces, pedestrians, foreground, etc.) video surveillance, visual biometrics (iris, fingerprints), and medical image analysis (localization of tumors and lesions, measurement of tissue damage, surgical planning, etc.)[1]. On the other hand, current state-of-the-art methods for image segmentation are typically validated with small images (e.g., $\leq 480 \times 320 \approx 0,15$MP) and face great difficulty scaling to large images due to high computational costs and memory requirements.

For instance, methods based on graph cuts [2, 3, 4] or probabilistic graph matching [5] usually require building an affinity matrix, which indicates the measured similarity between every two regions in the image (e.g., ranging from pixels to sets of pixels). Once this affinity matrix is obtained, the minimum graph cut can be determined using efficient algorithms [2] that minimize the graph cut energy. The drawback of this approach is the high memory and computation costs associated with computing the affinity matrix.

To evaluate all pairs of regions has quadratic complexity of $O(N^2)$ on an image of size N (in pixels). But as the image size tends to increase by rows or columns, rather than by single pixels, we consider the image size as $N = m \times n$, where m and n denote the image dimensions (rows and columns). Since both dimensions usually have comparable magnitudes (i.e., $m/n \approxeq 1$), we assume that $N \approxeq n^2$. Hence, computing the affinity matrix becomes expensive ($O(n^4)$) for larger images, especially in terms of memory. While this limitation can be overcome by excluding from the computation the pixels that are not neighbors and unlikely to be similar (e.g., pixels that are far apart), it is still costly to allocate the memory for computing the affinity matrix. Even though sparse matrix representations provide a trade off between memory usage and processing

time as the number of elements to be processed is reduced, for larger images a large amount of memory is still required for computing the affinity matrix. Also, the segmentation quality rapidly downgrades as the number of similarities evaluated for each pixel decreases, i.e. as the number of non-zero elements per row/column in the affinity matrix diminishes by discarding pixels that are not neighbors and unlikely to be similar from the computation.

An alternative approach relies solely on the similarity of individual pixel feature vectors [6, 7, 8]. Similar to data clustering algorithms, these methods allow more efficient implementations by ignoring the spatial relationships and spatial constraints in the segmentation process, and usually lead to a decay in segmentation quality. To mitigate these effects, Zhu et al. [9] proposed the use of a very sophisticated version of the Expectation-Maximization algorithm (EM), combining multiple types pixel wise frequency and contrast features. Although it does produce better segmentation than other cluster based methods, it is not as good as graph cuts or region based techniques, and has high computational cost, due to the complexity of EM. Already [10] proposed a combination of non-negative matrix factorization (NMF) to cluster the image pixels, and level-set segmentation using an energy function based on the NMF coefficients. However, this method does not account for color information, and the NMF makes it computationally expensive even for small grayscale images.

Region based segmentation methods have been proposed for image [11, 12, 13, 14, 15], and object segmentation [16, 17, 18]. These techniques rely heavily on the spatial relationships between pixels, and often require the evaluation of pairwise pixel relationships, which tends to lead to computationally expensive methods. On the other hand, these methods tend to require less memory than graph cuts, since the segmented regions are obtained by processing local sets of pixels, rather than processing all image pixels (e.g., using global methods).

To tackle the efficiency problems mentioned above, recent works have approached the segmentation problem using multi-level models [19, 20]. On a lower level, the image is evaluated locally, dividing it in a large number of homogeneous segments with some over segmentation method. Then, in an upper

level, these segments are globally evaluated with some clustering technique that aggregates them into larger regions. This strategy is able to reduce the segmentation time significantly without large compromises in terms of segmentation quality.

In a similar fashion, [21] used the Voronoi diagram of the image, to obtain smaller regions, and then used a split and merge technique to assemble these Vorono regions in larger ones. In another study [22], a combination of superpixels, fuzzy c-means and graph cuts was used to produce hierarchical segmentation. As these algorithms allow efficient implementations, it can achieve fast segmentation, but at expense of boundary accuracy, that is limited by the superpixels quality. Another approach, proposed in [23], also assembles superpixels into larger regions, but it uses an multilevel graph to represent both, pixel-wise, region-wise and pixel-region relations. On the other hand, the evaluation of all edges in this graph largely increases the algorithms complexity.

Reliable over-segmentations (e.g., based on superpixels) can be obtained with a variety of algorithms, such as nCuts [24] or watersheds [25], and recent developments have enabled segmentations to be made at low computational costs [26]. On the other hand, the over-segmentation becomes a critical stage since the accuracy of these approaches depends on the quality of the initial over-segmentations. Therefore, unsatisfactory superpixels tend to not lead to the desired segmentation quality. To deal with this issue, the combination of multiple superpixel methods was proposed in [20] to increase the robustness to the initial over-segmentation. However, such an approach still does not allow for pixel-wise precision, and has high memory costs as the integration of multiple segmentations uses spectral clustering to combine them into a single segmentation.

Among the graphical models and approaches proposed for image processing, graph compression techniques remain mostly unexplored in the image segmentation domain. These techniques aim to summarize a given graph using fewer graph vertices and edges, and in this work we show that with an appropriate graph setup, this approach can be used successfully to perform image segmenta-

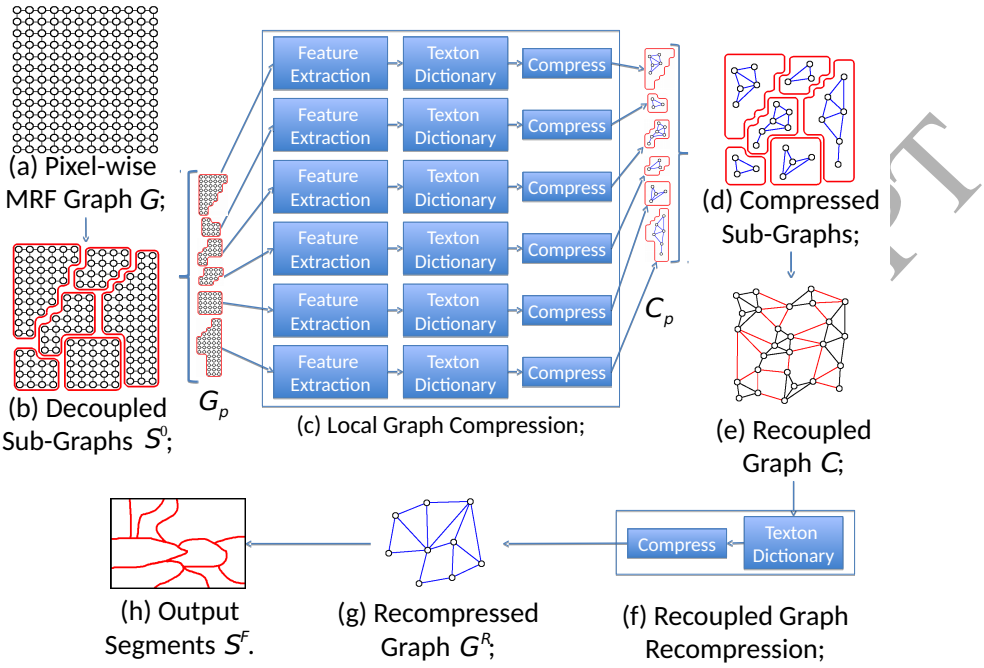


Figure 1: Overview of the proposed method: (a) initial region adjacency graph (large and dense); (b) sub-graph decoupling (small and dense); (c) sub-graph compression; (d) compressed sub-graphs (small and sparse); (e) recoupled graph (medium size and sparse); (f) recoupled graph re-compression; (g) recompressed graph $\mathcal{F}$ (small and sparse).

tion. However, this type of graph analysis requires the evaluation of all vertices and edges in the graph, leading to computationally expensive algorithms. Sometimes graph based methods may turn to be unfeasible for massive and dense graphs that frequently arise in image processing, especially in high resolution imagery.

In order to achieve scalability of the segmentation method, with a pixel-wise precision and without compromising the quality of the segmentation, a novel decoupled sub-graph compression (DSC) approach for efficient and scalable segmentation is proposed in this work. Rather than using a fully connected graph of pixel-wise similarities to represent the image, the method proposed here models the image using a region adjacency graph [11] that iteratively adapts to fit

the desired image segments. This region graph is decoupled in smaller sub-graphs, which are independently compressed and then recombined into the final segmentation, as illustrated in Fig. 1.

The DSC strategy allows the implementation of scalable segmentation algorithms with pixel-wise boundary precision. It also exploits the local maxima of the distribution of the proposed texture features that are observed when only small portions of the image are evaluated independently. By employing texton dictionaries [14] to represent the texture regions, the DSC strategy is able to explore feature locality, generating more efficient local dictionaries for compressing individual sub-graphs, which allows to consider also weaker boundaries and obtain more accurate final segmentations.

The main contributions of this work can be summarized as follows:

1. Propose a novel, fast and scalable graph compression algorithm for image segmentation;
2. Present an extension of the region graph model to explore the features locality (allowing to process weaker boundaries), as well as to robustly encompass the graph decoupling and recoupling operations (essential to the proposed strategy);
3. Introduce a robust graph recoupling methodology that correctly combines the compressed sub-graphs during the segmentation, regardless of the order in which the sub-graphs are decoupled or compressed.

This paper is organized as follows. In Sec. 2 we present the proposed graphical model and segmentation method via graph compression. Sec. 3 discusses the computational complexity of the proposed method in terms of time and memory. Afterwards, Sec. 4 presents the experiments and key findings of this work. Finally, Sec. 5 draws conclusions from the performed experiments and proposes future developments.

2. Proposed Scalable Segmentation Strategy

In this work, the image segmentation is processed as the compression of a graph $\mathcal{G}$ that represents the intrinsic characteristics of the evaluated image I. A graph compression algorithm transforms a given graph $\mathcal{G}$ into a smaller graph $\mathcal{F}$ with the same topological information (relative position of the vertices and length of the edges), as presented bellow.

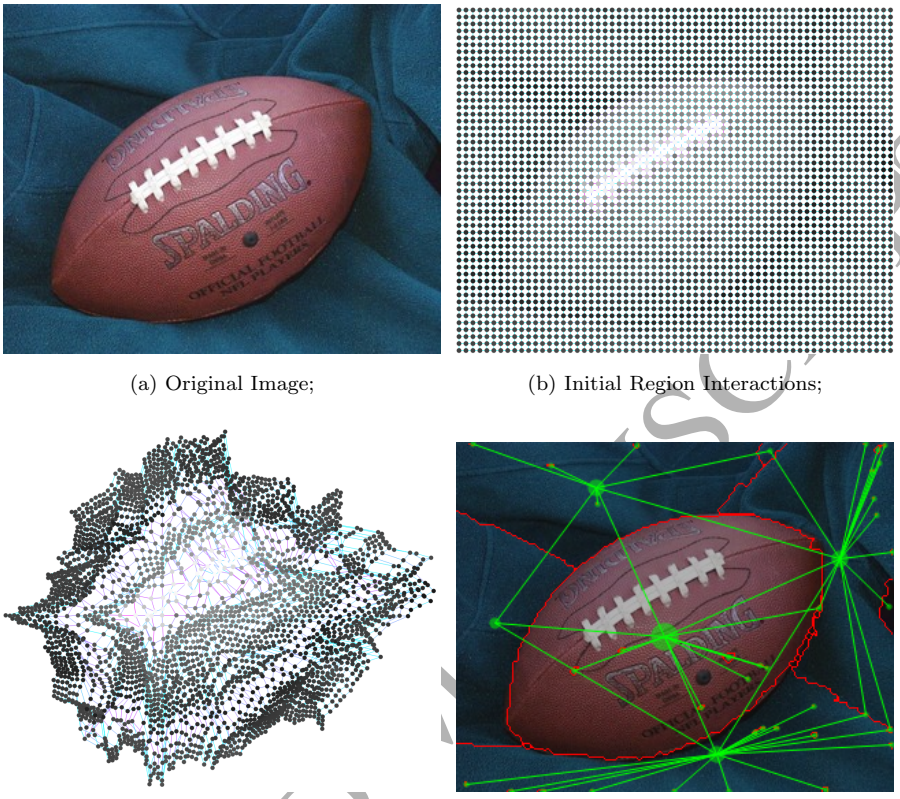
Definition 1 (Graph Compression)**.** *Let $\mathcal{G} = (V, E)$ be an arbitrary connected graph, formed by a set of vertices V and a set of edges E connecting those vertices. A graph compression is a function $\mathcal{H} : \mathcal{G} \rightarrow \mathcal{F}$ that receives as input the graph $\mathcal{G}$, and outputs another graph $\mathcal{F} = (V_{\mathcal{F}}, E_{\mathcal{F}})$ (called the compressed graph), composed of a vertex set $V_{\mathcal{F}}$ and an edge set $E_{\mathcal{F}}$, such that:*

1. $|V| < |V_{\mathcal{F}}|$,
2. $|E| < |E_{\mathcal{F}}|$,
3. *$\mathcal{F}$ has the same topology as $\mathcal{G}$,*

where $|\cdot|$ denote the number of elements of the set.

In the proposed method an image I is modeled as a region graph $\mathcal{G} = (V, E)$, where the vertices $v_i \in V = \{v_i : r_i \in S_0\}$ represent texture regions r_i ($1 \leq i \leq |S_0|$) contained in an arbitrary image segmentation S_0, and the edges $e_{ij} \in E = \{e_{ij} : r_i, r_j \in R$ and r_i is adjacent to $r_j\}$ indicate the boundaries between the regions r_i and r_j. Although any partition of I can be represented in this fashion, in this work we wish to focus on the representation of the local interactions between neighboring pixels (and sets of pixels). Therefore, we restrict S_0 to be a superset of non-empty and pairwise disjoint sets of connected pixels from the image I.

Moreover, each edge has a weight w_{ij}, indicating the interaction strength between a pair of region vertices r_i and r_j. As illustrated in Fig. 2, the weights associated to the edges can be seen as the geometric distance between the vertices in a topological representation of the graph. When defined in this way,



(a) Original Image; (b) Initial Region Interactions;

(c) Weighted Region Graph; (d) Compressed Region Graph.

Figure 2: Region graph setup: (a) the original image (50×63), (b) initial pixel-wise regions interactions using a Markov Random Field, (c) initial weighted region graph, with edges length proportional to the boundary strength, (d) compressed region graph, overlaid to the image segmentation.

the graphical structure of the image regions will be clearly related to the image structure, with similar regions vertices placed closer, and distinct regions vertices placed further away.

Consequently, any changes in this graph will lead to a new segmentation state. Since the graph compression $\mathcal{F} = \mathcal{H}(\mathcal{G})$ is a transformation that preserves the topology of $\mathcal{G}$, it groups together the vertices connected by smaller edges. Since in $\mathcal{G}$ smaller edges indicate greater similarity of the regions, the compressed vertices represent connected groups of pixels that of similar charac-

teristics in I. Hence, compressing $\mathcal{G}$ is equivalent to segmentation of image I, and the configuration of the compressed graph $\mathcal{F}$ yields the segmentation map S_F. Furthermore, to compute the compressed graph $\mathcal{F}$, all the initial edges in E must be evaluated. Since S_0 must be a partition of I in a set of connected components, the region graph will have $|V|^2 \geq |E| \geq |V| - 1$, what may result in significant computational complexity.

To make the graph compression process more efficient, this work employs a divide-and-conquer-like strategy, as illustrated in Fig. 1. The initial graph $\mathcal{G}$ is first decoupled in a collection of small sub-graphs $G_1, G_2, \cdots, G_B$ that can be processed efficiently. Next, each sub-graph G_p is transformed in a small compressed subgraph $C_p = \mathcal{H}(G_p)$ using Def. 1. Then, the compressed sub-graphs are recoupled together in a single connected graph $\mathcal{C}$, that approximates $\mathcal{H}(\mathcal{G})$ from the local compressions of the previous stage. Finally, a last graph compression is applied to $\mathcal{C}$ to ensure robustness to errors in the previous stages.

To implement this algorithm, we propose two operations, one for dividing the initial graph into smaller sub-graphs, called decoupling; and another for recombining the sub-graphs into a single graph, called recoupling. The graph decoupling operation is defined as follows:

Definition 2 (Graph Decoupling)**.** *Let $\mathcal{G} = (V, E)$ be an arbitrary connected graph, formed by a set of vertices V and a set of edges E, connecting those vertices. A decoupling of $\mathcal{G}$ is a partition of V into B non-empty subsets of vertices $\{V_1, V_2, \cdots, V_B\}$ with $1 \leq B \leq |V|$, resulting in the decoupled graph $\mathcal{G}' = ((V_1, E_1), (V_2, E_2), \cdots, (V_B, E_B))$, where (V_p, E_p) is a decoupled sub-graph.*

Def. 2 implies two additional concepts that are essential to the graph recoupling. The first concept is the decoupled sub-graph, which is related to the connected components of $\mathcal{G}'$, and is formalized in Def. 3. The second concept is the residual edges, and addresses the edges removed from a graph to disconnect the connected components. These concepts are formalized as follows in Defs. 3 and 4:

Definition 3 (Decoupled Sub-Graph)**.** *Let $\mathcal{G} = (V, E)$ be an arbitrary connected*

graph, and $\mathcal{G}' = ((V_1, E_1), (V_2, E_2), \cdots, (V_B, E_B))$ *its decoupled graph. A decoupled sub-graph* $G_p = (V_p, E_p)$ *is a sub-graph of the initial connected graph* $\mathcal{G}$*, composed of the vertices in a single subset of vertices* $V_p \in \mathcal{G}'$ *of the decoupling partition, and the set of edges* E_p *that connect these vertices:*

$$E_p = \{e_{ij} \in E : v_i, v_j \in V_p : 1 \leq p \leq B\}.$$

Definition 4 (Decoupled Residual Edges)**.** *Let* $\mathcal{G} = (V, E)$ *be an arbitrary connected graph, and* $\mathcal{G}' = ((V_1, E_1), (V_2, E_2), \cdots, (V_B, E_B))$ *its decoupled graph. The residual edges of* $\mathcal{G}'$ *are the edges that have one end in one of the decoupled vertex subsets* V_p *and the other end in another decoupled subset* V_q*, with* $1 \leq p, q \leq B$ *and* $p \neq q$*. The residual edges set* E_R *is formally given by:*

$$E_R = \{e_{ij} \in E : v_i \in V_p \wedge v_j \in V_q \wedge p \neq q\}.$$

Using the concepts above, the decoupled sub-graphs can be re-assembled into a single graph with a graph recoupling operation, formally defined as:

Definition 5 (Graph Recoupling)**.** *Let* $C_1, C_2, \cdots, C_B$ *be a list of* B *pairwise distinct decoupled sub-graphs, and let* $E_R = \{e_{ij} \in E_{\mathcal{C}} : v_i \in C_p \wedge v_j \in C_q \wedge p \neq q\}$ *be a set of residual edges for those decoupled sub-graphs, with* $C_p = (V'_p, E'_p)$*. The recoupling of* $C_1, C_2, \cdots, C_B$ *is the process of assembling all these sub-graphs in a single connected graph* $\mathcal{C} = (V_{\mathcal{C}}, E_{\mathcal{C}})$*, composed respectively by the following vertex and edge sets:*

$$V_{\mathcal{C}} = \bigcup_{p=1}^{B} [V'_p] \quad \text{and} \quad E_{\mathcal{C}} = \bigcup_{p=1}^{B} [E'_p] \cup E'_R.$$

The decoupling, compression, and recoupling operations are combined in Alg. 1, that summarizes the proposed decoupled sub-graph compression algorithm for scalable image segmentation. In the remainder of this section, we describe the details of the algorithms associated to Defs. 1, 2, 4 and 5.

Algorithm 1 Decoupled Sub-Graph Compression Segmentation Algorithm

Input: Image I

Output: Segmentation map S_F

$\mathcal{G}$ = a region graph from image I
$(G_1, G_2, \cdots, G_B)$ = decouple $\mathcal{G}$ into B sub-graphs ▷ Using Def. 2
E_R = residual edges of $(G_1, G_2, \cdots, G_B)$ ▷ Using Def. 4
for $p = 1$ to B **do**
    $G_p = \mathcal{H}(C_p)$ ▷ $\mathcal{H}(\cdot)$: graph compression Using Def. 1
end for
E'_R = updated residual edges
$\mathcal{C} = C_1 \cup C_2 \cup \cdots \cup C_B \cup E'_R$ ▷ Using Def. 5
$\mathcal{F} = \mathcal{H}(\mathcal{C})$
S_F = segmentation yielded by $\mathcal{F}$
return S_F.

2.1. Initial Region Graph

Since in the proposed graphical model a region vertex is allowed to represent any number of pixels, as well as be connected to any number of other vertices, several strategies can be used to set up the initial regions S_0 to be represented in $\mathcal{G}$, each with distinct effects. Using a single over-segmentation may produce incorrect boundaries that are hard to correct, and multiple over-segmentations require the use complex models, and cannot achieve pixel-wise precise boundaries.

In order to precisely represent the image I (of size $M \times M$), the region graph $\mathcal{G} = (V, E)$ is initially set to represent each pixel l_i as a unique region vertex $v_i \in V$, labeled $r_i \in S_0 = \{1, \cdots, M^2\}$, and the edge set E is initially configured to represent the interactions of a 4-neighborhood Markov Random Field [27] (see in Fig. 2-b the relations and in Fig. 2-c the initial graph). This scheme allows the potential to achieve fine boundary detection at pixel-wise precision. The edge weights w_{ij}, that define the topology of $\mathcal{G}$, are obtained comparing the texture descriptors of the region vertices associated to it. That will be discussed in detail in Section 2.3, as the edges are computed locally and are closely related to the compression process.

The scheme above leads to a large and dense region graph, as illustrated in Fig. 1-a. But compressing a graph requires evaluating all its vertices and edges, what could make the process computationally expensive, and even unfeasible for high resolution images. To deal with this potential issue, we propose a strategy that uses graph decoupling (Def. 2) and recoupling (Def. 5) to make the process scalable for large graphs. Moreover, by defining a large graph we intent to show that it can be handled nicely within the proposed approach.

2.2. Graph Decoupling

From definitions 3 and 4, the graph decoupling of $\mathcal{G}$ into $\mathcal{G}'$ implies in removing a cut-set of residual edges $E_R \subset E$ from the initial graph, so that $\mathcal{G}$ is partitioned into B sub-graphs $G_p = (V_p, E_p)$, $1 \leq p \leq B$. We desire all sub-graphs to be small enough to allow compression at low cost, as illustrated in Fig. 1-b.

Furthermore, according to Def. 2, the decoupling of a graph divides its vertices in B pairwise disjoint sub-sets. As in $\mathcal{G}$ each vertex is associated to exactly one pixel on I, decoupling this graph is equivalent to partitioning the image into disjoint groups of connected pixels, which is the very definition of image segmentation. Therefore, graph decoupling may be obtained as a coarse image over-segmentation, which we denote by S_D. In this scheme, each segment $s_p \in S_D$ indicates one of the sub-graphs G_p to be decoupled.

In the experiments of the proposed strategy, we evaluated a few distinct decoupling methods, for which the results are reported in Table 1. They are:

- **Coarse Compression** (C-DSC): The image is down-scaled by a factor of κ and then segmented recursively using the proposed DSC algorithm. Each decoupled sub-graph corresponds to an image segment obtained at this lower scale segmentation;

- **Mori Superpixels** [24] (SP-DSC): An over-segmentation method based on normalized graph cuts [28], which is set to find a large number of partitions, that will be the superpixels. Each decoupled sub-graph corresponds

Table 1: Comparison of Decoupling Strategies on 500×750 images

Strategy		PRI	F-measure	Time (s)
No Decoupling	(Full-GC)	0.635 ± 0.161	0.333 ± 0.127	3419.2 ± 839.4
Block	(B-DSC)	0.805 ± 0.088	0.533 ± 0.109	$\mathbf{2954.6 \pm 3113.4}$
Coarse Compression	(C-DSC)	$\mathbf{0.812 \pm 0.084}$	$\mathbf{0.568 \pm 0.132}$	6217.2 ± 1390.5
Waterpixels [25]	(WP-DSC)	0.805 ± 0.088	0.552 ± 0.115	3116.3 ± 505.0
Superpixels [24]	(SP-DSC)	$\mathbf{0.812 \pm 0.082}$	0.559 ± 0.118	4229.1 ± 773.3

Bold values indicate the best result.

to a single superpixel;

- **Waterpixels** [25] (WP-DSC): An over-segmentation methods based on watersheds. It combines a regular grid gradient to the image gradient in order to reinforce the generation of small, homogenous connected components on the watershed transform [29], which are image segments or superpixels. Each decoupled sub-graph corresponds to a single superpixel;

- **Block Partitioning** (B-DSC): The image is divided in non-overlapping square blocks of size $B \times B$. Each decoupled sub-graph corresponds to one of these image blocks;

- **No Decoupling** (Full-GC): In this case, there is a single "sub-graph" $G_1 = \mathcal{G}$, thus the graph compression is applied directly to the initial graph $\mathcal{G}$. This is included as a reference to evaluate the performance of the other decoupling methods.

As shown in Table 1, each decoupling method produces a slightly different result, regarding the cost and the quality of the decoupled segmentation S_D. But in all cases decoupling the initial graph is more efficient than a full graph compression of $\mathcal{G}$, both in terms of quality (PRI and F-measure) and cost (time). The benefits and issues of each of decoupling segmentation method will be more deeply discussed in Sec. 4.

Furthermore the proposed sub-graph compression uses a fine scale and local maxima of the edge weights (i.e. vertices interactions), that are easier to detect

within small portions of the image. In this way, the decoupled sub-graphs prevent over-compression (i.e. avoiding misrepresenting relevant boundaries), while the recoupling stage prevents under-compression (avoiding the representation of false boundaries). This makes the decoupling strategy advantageous not only in terms of efficiency, but also adds to it a potential for providing better image segmentations.

2.3. Sub-graph Compression

While graph compression have been well defined in Def. 1 it is restricted to connected graphs. Since the decoupled graph $\mathcal{G}'$ is not connected, we expand the definition of graph compression to disconnected graphs, as:

Definition 6 (Disconnected Graph Compression)**.** *Let $\mathcal{G} = (V, E)$ be an arbitrary graph with $B \geq 1$ connected components, and $G_1, G_2, \cdots, G_B$ be the sub-graphs of $\mathcal{G}$ containing exactly one connected component of $\mathcal{G}$. The graph compression of $\mathcal{G}$ is equal to the individual compression of each of its connected components G_p, with $p = 1, 2, \ldots, B$.*

Therefore, the individual compression of all decoupled sub-graphs G_p can be used to approximate the compression of $\mathcal{G}$, at least to a certain degree. The difference between these segmentations will be addressed in the Sec. 2.5.

Consider the compression of a decoupled-sub-graph $G_p = (V_p, E_p)$, as shown in Fig. 1-c (local graph compression). The proposed sub-graph compression has three main steps. First, one edge $e_{ij} \in E_p$ is selected to be evaluated. Then, its weight w_{ij} is computed locally. Finally if w_{ij} indicates a strong interaction, then the corresponding vertices are compressed (combined) into a new super-vertex. This steps are repeated for all edges in E_p. When no more compressions are possible, the compressed sub-graph C_p is obtained.

Because this process gradually transforms G_p in the compressed sub-graph C_p, we denote the sub-graph state after t vertex compressions by G_p^t, with $G_p^0 = G_p$ being the initial state, and $G_p^\Omega = C_p$ being the final state after all edges have been evaluated. The compression process is summarized in Alg. 2. Details of each step are addressed in one of the following sub-sections.

Algorithm 2 Graph Compression Algorithm

Input: Region graph $G_p = (V_p, E_p)$

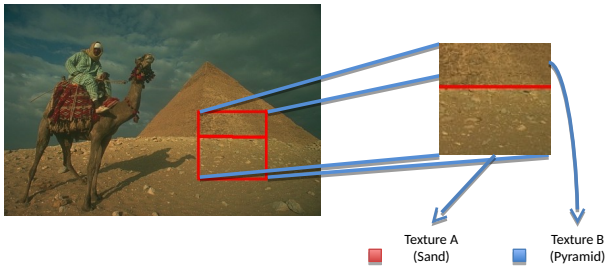
Output: Compressed graph $C_p = (V'_p, E'_p)$

Let $G_p^0 = G_p$; $t = 0$;
$\phi(v_i) = 1, \forall v_i \in V_p$
D_p = texton dictionary for G_p; ▷ Using Alg. 3
$P = \text{sort}(E_p^0)$;
for $e_{ij} = pop(P)$ **do**
    Let v_i, v_j be the vertices connected by e_{ij};
    $P = P - \{e_{ij}\}$;
    Compute w_{ij}; ▷ Using (1)
    if w_{ij} is small enough **then**
        $v_u = v_i + v_j$; $t = t + 1$;
        $G_p^t = (G_p^{t-1} - \{v_i, v_j\}) \cup \{v_u\}$;
    end if
end for
return G_p^t.

2.3.1. Edge Evaluation Order

Because the vertex compression is a local optimization, the order in which the edges are evaluated is determinant to the outcome. Since vertices with stronger interactions (smaller edge sweights) are more likely to be compressed, such edges should be analyzed first. Initially, all edges in E_p are placed in a adjacency priority queue, that will keep them sorted in ascending order of their weights w_{ij}. After an edge is analyzed, it is removed from the queue. If a compression occurs, the remaining edges are updated to reflect the new graph configuration. In this way, the next edge to be evaluated is always the one on top of the queue. When the queue becomes empty the compression ends, with the sub-graph at state $G_p^\Omega = C_p$, and the compressed graph C_p is obtained.

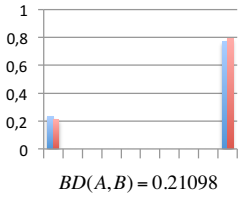
Moreover, compressing only directly connected vertices ensures that the compressed sub-graph is also a region graph (represents an image segmentation). Therefore, vertices $v_i \in G_p$ and $v_j \in G_q$ form distinct decoupled sub-graphs $G_p \neq G_q$, cannot be compressed. Consequently, each sub-graph G_p can be compressed independently from any other sub-graphs, as stated in Def. 6.



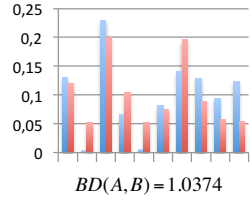
(a) Original image; (b) Decoupled region;

(c) Generic Texton Dictionary;

(d) Localized Texton Dictionary;



(e) Texton Histogram;



(f) Texton Histogram.

Figure 3: Impact of local dictionaries in the texture representation. From the original image (a) we select two adjacent textures from distinct objects within the decoupled region (b): the sand (Texture A) and the pyramid (Texture B). When using (c) the generic texton dictionary both textures have similar (e) texture histograms, but with (d) the specialized dictionary (f) their texton histograms can be distinguished from each other. BD is the Bhattacharyya distance in (1).

2.3.2. Edge Weight Computation

The proposed bottom-up strategy allows edge weights to be computed dynamically (on demand) and helps to mitigate the memory cost of the dense graphs. For this purpose, we define two properties associated to each vertex v_i: the compression level $\Phi(v_i)$ and the texton histogram $H(v_i)$. The compression level $\Phi(v_i)$ is a count of how many vertices have been compressed into that vertex, and indicates the portion of the initial graph that is contained in the super-vertex v_i. An uncompressed vertex $v_i \in V_p$, in the initial state of sub-graph G_p^0, has $\Phi(v_i) = 1$. As the graph compression progresses and the graph configuration changes, the compressed vertices will have different values of compression levels (see (2) and (3) for details on compression updates).

Algorithm 3 Texton Dictionary Algorithm

Input: Image I, Region of interest r, Number of textons K

Output: Texton dictionary D

for Every pixel $l \in r$ **do**
  Let $\mathcal{N}(l)$ be a patch around the pixel l
  Ψ = random matrix $(m \times n)$ ▷ See [30] for details
  $f(l) = \Psi * \mathcal{N}(l)$
end for
D = centroids of k-means$(f(:), K)$
return D.

The second property, texton histogram $H(v_i)$, is a statistical descriptor of the contents of the image region r_i associated to the region vertex v_i. The histograms arise from the texton dictionary approach [31, 32, 14], that use bag-of-features to represent low level texture features of the regions. To construct this dictionary, the stochastic patch features [14] are extracted from the image pixels. The extracted feature vectors are then clustered using k-means [33], and the resulting cluster centroids will be the textons (atoms) composing the texton dictionary D. The algorithm for building this dictionary is described in Alg. 3, where the region of interest r is a binary mask for image, indicating the pixels to be considered.

The texture within a region is then represented by the occurrence probability of each texton of the dictionary. More precisely, each pixel l is assigned to the texton most similar to its feature vector $f(l)$ (using L_2 norm), and $H(v_i) = \{h_c(v_i) : 1 \leq c \leq |D|\}$ counts how many pixels $l \in r_i$, was assigned to each dictionary texton. Also, these histograms are normalized to have $\sum_c h_c(v_i) = 1$, so the vertices with distinct compression level Φ can be compared fairly.

When compressing a single sub-graph G_p, the texton histograms can be made more precise and computed faster by using a dictionary D_p optimized to represent the textures of that sub-graph specifically. This D_p is obtained by providing a better set of feature samples $f(\cdot)$ to k-means when constructing the dictionary. Since G_p represents the over-segment s_p of the decoupling segmentation (S_D), pixels outside s_p are not relevant when compressing G_p.

Therefore, clustering only the feature vectors of pixels within s_p to build D_p is enough to produce a specialized dictionary, as illustrated in Fig. 3. In Alg. 3 this is achieved by selecting the region of interest as $r = s_p$, the over-segment associated to G_p. This is one of the main contributions of this work, that creates more accurate representation of the sub-graph textures that helps to dynamically detect the local maxima of the region interactions. Moreover, building D_p is faster, as there are less samples to cluster.

We then define the edge weight w_{ij} for an edge e_{ij}, linking a given pair of vertices v_i and v_j, as the measure of similarity between the histograms using the Bhattacharyya distance [34]:

$$w_{ij} = -ln\left(\sum_{c}\sqrt{h_c(v_i)\cdot h_c(v_j)}\right). \tag{1}$$

2.3.3. Vertex Compression

In the proposed method, if the weight w_{ij} of the edge on top of the priority queue is small enough, the vertices v_i and v_j are compressed into a new vertex v_u, representing the union of their associated image regions r_i and r_j, such that:

$$\Phi(v_u) = \Phi(v_i) + \Phi(v_j), \tag{2}$$

$$H(v_u) = \frac{H(v_i)\times\Phi(v_i) + H(v_j)\times\Phi(v_j)}{\Phi(v_u)}. \tag{3}$$

Also, the edges of G_p^t are updated to reflect the new vertex, so all edges that previously connected v_i or v_j to other vertices in G_p^{t-1} are directed to v_u, and any edges from v_u to itself are discarded. The graph structure and its operations (compress vertices, redirect edges and remove loop edges) are efficiently handled with a union-find structure [35] for the vertices and a list of adjacencies for the edges.

In the union-find structure, the vertices are represented by tree nodes in a vector $\bar{v}$, with one element $\bar{v}(i)$ for each vertex $v_i \in V_p$, containing the index of its parent. At state G_p^0, every vertex v_i is uncompressed, so $\bar{v}(i) = i$, meaning it is the root of that tree. When vertices v_i and v_j are compressed, we simply need to combine their trees, by making the root of one tree as the child of the other, such as $\bar{v}(i) = j$, and v_j becomes a new compressed super-vertex. Therefore,

if a vertex v_i has $\bar{v}(i) \neq i$, it has been compressed into a super-vertex, which will be the root of that tree. The operation of finding the root of the tree that contains node i is called find(i) and provides the index of the root node. The find operation has a worst case complexity of $O(n)$, but in this scenario the average case takes constant time, and the tree combination can also be done in constant time $O(1)$. By using the find($\cdot$), the adjacency list does not have to be updated often when updating the graph. Moreover, the union-find also allows to track the correspondence between the initial vertices $v_i \in G_p^0 = G_p$ and the final super-vertices $v_x \in G_p^\Omega = C_p$. This will be useful in the recoupling stage (see Sec. 2.4).

For the sake of efficiency, the proposed DSC is designed as a greedy algorithm, and compressed vertices are never uncompressed in the process. It shall be observed that local DSC optimizations may not lead to the global optimum individually, but the DSC process tends to avoid getting stuck in local minima as discussed below. To prevent undesirable excessive compressions, a statistical penalty Λ is computed based on the compression level of the vertices linked by the evaluated edge:

$$\Lambda(v_i, v_j) = \frac{f^2}{2Q}\left[\frac{ln(|V_p^0|^2)}{\Phi(v_i)} + \frac{ln(|V_p^0|^2)}{\Phi(v_j)}\right], \tag{4}$$

where $|V_p^0|$ is the number of vertices in G_p^0, Q is the regularization term controlling the size of the compressed graph, and $f = 256$ is the number of intensity levels in the image I.

Using this penalty $\Lambda(v_i, v_j)$ and the edge weights w_{ij} from (1), all pairs of region vertices (v_i, v_j) that are connected by a single edge e_{ij} are evaluated, and compressed if the vertex compression likelihood $\alpha(v_i, v_j)$ is greater than a random number $u \sim \mathcal{U}(0; 1)$, as discussed in [11]:

$$\alpha(v_i, v_j) = \exp\left[-\frac{w_{ij}}{\Lambda(v_i, v_j)}\right]. \tag{5}$$

This stochastic graph compression strategy accounts for information uncertainty (such as noisy feature samples, and the lack of information about the image contents), and thus makes the DSC less prone to local minima [14, 11].

Once all edges $e_{ij} \in E_p$ (the edge set of G_p) have been evaluated, the sub-graph will be in a state G_p^{Ω}, which contains fewer vertices and edges. this state is called the compressed sub-graph $C_p = G_p^{\Omega}$.

2.4. Graph Recoupling

While compressing the decoupled sub-graphs G_p is more efficient than compressing the initial G, it produces only disconnected components. To obtain a single segmentation map for the image I, the compressed sub-graphs $C_1, C_2, \cdots, C_B$ must be combined in a single connected region graph $\mathcal{C} = (V_{\mathcal{C}}, E_{\mathcal{C}})$ (see Fig. 1-e). Using Def. 5 it is possible to recreate a connected graph from decoupled sub-graphs, given that a set of residual edges E'_R is provided.

Let us consider the compressed sub-graph $C_p = (V'_p, E'_p)$. The graph compression algorithm presented in Sec. 2.3 ensures that its compressed vertex set V'_p represents the same image pixels as the uncompressed vertex set V_p of sub-graph G_p. Consequently, the recoupled vertex set $V_{\mathcal{C}}$, given by the union of all compressed vertex sets:

$$V_{\mathcal{C}} = \bigcup_{p=1}^{B} V'_p, \tag{6}$$

as proposed in Def. 5, represents all pixels in the image I.

Furthermore, the union find structure used in the compression stage tracks the correspondence between each uncompressed vertex $v_i \in V_p$ and its compressed super-vertex $v_x \in V'_p$, where $x = \text{find}(i)$. Therefore, the set of recouple residual edges E'_R for recoupling the compressed subgraphs, is obtained updating the decoupled residual edge set E_R, as:

$$E'_R = \{e_{xy} : e_{ij} \in E_R \wedge v_x = \text{find}(v_i) \wedge v_y = \text{find}(v_j)\}. \tag{7}$$

Since $V_{\mathcal{C}}$ has much less vertices than V, repeated edges are expected to arise from (7), of which only one sample is included in E'_R.

Combining E'_R with the edges of all compressed subgraphs $E'_1, E'_2, \cdots, E'_B$ into Def.5, we obtain the edge set $E_{\mathcal{C}}$ for the recoupled graph:

$$E_{\mathcal{C}} = \bigcup_{p=1}^{B} E'_p \bigcup E'_R. \tag{8}$$

With this configuration, the recoupled graph $\mathcal{C} = (V_\mathcal{C}, E_\mathcal{C})$ has a single connected component, and therefore is a region graph the input image I. As such, there is a segmentation map S_R associated to $\mathcal{C}$, which we call the recoupled segmentation.

2.5. Graph Re-compression

The decoupling and recoupling operations make the compression of $\mathcal{G}$ more efficient, but it is also limited to sub-optimal solutions. Because the recoupled graph includes edges from E'_R, that have not been evaluated in the sub-graph compression, $\mathcal{C}$ is only a loose approximation of a full compression of the initial graph $\mathcal{G}$. Similarly, S_R does not represent the optimal detection of boundaries.

To address issue, a final graph compression is performed on the recoupled graph $\mathcal{C}$. Given that the $\mathcal{C} = (V_\mathcal{C}, E_\mathcal{C})$ will be small (compared to $\mathcal{G}$) and sparse (with few edges in comparison to the number of the vertices), this compression can be computed efficiently. In this stage, Alg. 2 is used to transform $\mathcal{C}$ into the final graph $\mathcal{F} = \mathcal{H}(\mathcal{C})$. This stage enhances the region graph compression in a global scale, making the segmentation robust to errors in the decoupling stage at low computation cost, which is one of the contributions of this work.

Similarly to the sub-graph compression, the re-compression stage has the same three steps — select an edge, update its weight and, if it indicates a strong interaction, compress the related vertices into a new supervertex — that are repeated for all graph edges $e_{xy} \in E_\mathcal{C}$. Once all edges have been evaluated, final re-compressed region graph $\mathcal{F}$ is obtained, and the final image segmentation S_F is yielded by the union-find structure used to represent the vertices.

Since Alg. 2 does not uses any property of the decoupled sub-graphs that is not present in $\mathcal{C}$, it can be employed in this stage, using the same data structures (priority queue, union-find). In the re-compression stage, however, the weights w_{xy} of all edges $e_{xy} \in E_\mathcal{C}$ must be updated to reflect the state of $\mathcal{C}$ rather than the sub-graph C_p. To achieve this, the properties of all vertices $v_x \in V_\mathcal{C}$ must be adjusted accordingly. Because (6) does not change the sub-graph vertices, in the graph recoupling, it suffices to preserve the same compression level $\Phi(v_x)$

available in C_p for every $v_x \in V'_p$.

To compute the weights of any residual edge $e_{xy} \in E'_p$ added in the recoupling stage, the histograms $H(v_x)$ and $H(v_y)$ must be updated using the same texton dictionary. Because the specialized dictionaries $D_1, D_2, \cdots, D_B$ cannot properly represent textures classes outside their respective sub-graphs, they are not appropriate for this stage. Hence, a global texton dictionary $\mathcal{D}$ is constructed using Alg. 3. But because $\mathcal{C}$ is associated to all image pixels $l \in I$, $\mathcal{D}$ must be constructed using samples form the whole image. This is achieved by setting the region of interest r represent the whole image evenly.

Given the complexity of a clustering process, building the global dictionary $\mathcal{D}$ could compromise the scalability of the proposed method, especially in terms of memory. But since the image I is expected to be large and have high resolution, it is reasonable to presume that I presents large information redundancy within pixels neighborhoods. Therefore, region of interest r is set to uniformly select only 1 out of every $\kappa \geq 1$ pixels in the image (both vertically and horizontally). By doing so, the number of features vectors is down-sampled at a rate of $\frac{1}{\kappa^2}$, without any significant loss of representativeness of the new texton dictionary and histograms. Using $\mathcal{D}$, the histograms $H(v_x)$ of all vertices in $V_\mathcal{C}$ are updated, and the edge weights of $\mathcal{C}$ can be computed dynamically using (1), as the graph compression progresses.

3. Computational Complexity

In this section, we evaluate the time and memory asymptotic growth rate of the proposed method. As the algorithm has three stages (decoupling, compression and recoupling), their combination will result in the total processing cost. Without loosing generality, let us consider an image of size $n \times n = n^2$ pixels, where the initial graph will also have n^2 vertices, and approximately $2n^2$ edges (as it is assumed a 4-neighborhood at each pixel).

In the simplest approach for decoupling, the initial graph is divided into B sub-graphs of size $b \times b = b^2 = {}^{n^2}/_B$ vertices, with approximately $2b^2$ edges. This

operation is trivial, and can be done in constant time $T_D = O(n)$. The same operation can be done by indexing the regions of the image to be processed, i.e. the memory growth rate also is $M_D = O(n^2)$.

On the proposed compression algorithm each edge leads to one evaluation of vertices. For a graph of $2n^2$ edges, this algorithm has a complexity of $O(2n^2) = O(n^2)$. On the sub-graph compression stage, however, there are $2b^2$ edges, leading to a time growth rate of $O(2b^2) = O(b^2)$ for each sub-graph. Consequently, if $b \ll n$ (as proposed in the decoupling stage) this algorithm will be large graphs will be much more expensive to compress than many small graphs. On the other hand, the memory growth rate of this stage will be $O(Kb^2)$ per sub-graph, where K is a large constant related to the method parameters used for texture representation. Although there may be many sub-graphs for a single image, they are independent of each other and only the features and histograms pertaining the sub-graph that is being compressed need to be loaded at one time, allowing better memory management. Considering $\mathcal{G}$ divided in B sub-graphs, each with a size $b^2 = {}^{n^2}/_{B}$, the sub-graph compression stage time and memory growth will be bounded, respectively, by:

$$T_S = O(Bb^2) = O(B({}^{n}/_{B})^2) = O({}^{n^2}/_{B}),$$

$$M_S = O(Kb^2) = O(K({}^{n}/_{B})^2) = O(K{}^{n^2}/_{B^2}).$$

In the recoupling stage, the complexity comprises the steps of reconnecting the sub-graphs, rebuilding the textons dictionary and re-compressing the graph $\mathcal{C}$. For a recoupled graph $\mathcal{C}$ with v vertices, the reconnection complexity depends on the number of graph vertices taking time $O(v)$, which can be done without allocating any extra memory. The texton dictionary is built using k-means, so considering sub-sampling of the feature vectors, it has a time growth of $O({}^{Kn^2}/_{\kappa^2})$. The graph re-compression also depends on the number of graph edges, resulting in a time growth bounded by $O(v^2)$. Since the recoupled graph is much smaller than the decoupled sub-graphs, we have that $v \ll b^2 \ll n^2$, resulting in a time growth complexity of:

$$T_R = O({}^{n^2}/_{\kappa^2}) = O(v) + O({}^{n^2}/_{\kappa^2}) + O(p^2)$$

for this stage. Similarly, the memory growth rate of the recoupling stage also depends on the clustering process, which is:

$$M_R = O(Kn^2/\kappa^2).$$

Combining all stages, the resulting complexities for time and memory are, respectively:

$$O(T) = O(1) + O(n^2/B) + O(n^2/\kappa^2) = O(n^2/B + n^2/\kappa^2), \tag{9}$$

$$O(M) = max\big(O(n^2), O(Cn^2/B^2), O(Cn^2/\kappa^2)\big) = O(Cn^2/\kappa^2), \tag{10}$$

As such, if the image size changes to $2n \times 2n = 4n^2$, a full graph compression without decoupling will consume time and memory:

$$T([2n]^2) = ([2n]^2)^2 = 16n^4 \qquad \text{and} \qquad M([2n]^2) = C([2n]^2)^2 = 16Cn^4,$$

while the proposed method will use:

$$T([2n]^2) = (2n)^2/B + (2n)^2/\kappa^2 = 4n^2/B + 4n^2/\kappa^2,$$

$$M([2n]^2) = C(2n)^2/\kappa^2 = C4n^2/\kappa^2.$$

Therefore, the proposed algorithm will be efficient if all decoupled sub-graphs are small with respect to the image size (n^2).

4. Experiments and Discussion

To evaluate the quality and performance of the proposed segmentation strategy, a dataset of large images was assembled specifically for this task, allowing a fair comparison of the experiments that were conducted. This dataset consists of 60 high-resolution natural color images, all collected from the Internet. To allow a fair comparison of the evaluated methods, all images were down-sampled to 4 distinct resolutions, in a way that all have approximately the same size at each scale: 1000×1500, 500×750, 250×375 and 125×188. Moreover, since the segmentation quality may be highly subjective, a set of 3 or more hand-made segmentations is provided as groundtruths for each image. To measure

Table 2: Segmentation Cost and Quality Comparison in Large Images*.

Method	PRI	F-measure	Time (s)
B-DSC	$\mathbf{0.836 \pm 0.083}$	$\mathbf{0.591 \pm 0.106}$	4293.0 ± 3842.5
C-DSC	0.828 ± 0.085	0.553 ± 0.114	12821.5 ± 4158.7
WP-DSC	0.832 ± 0.087	0.584 ± 0.115	15740.4 ± 5171.3
SP-DSC	0.832 ± 0.083	0.575 ± 0.119	4519.8 ± 816.1
STRM [14]	0.558 ± 0.158	0.298 ± 0.124	6916.1 ± 2385.2
HCD [15]	0.522 ± 0.222	0.497 ± 0.222	1560.0 ± 520.3
FBS [7]	0.719 ± 0.103	0.409 ± 0.119	$\mathbf{183.8 \pm 137.9}$

* Images of size 1000×1500 pixels.

the segmentation quality, we use the Probabilistic Rand Index (PRI), the F-measure [15], as well as visual comparison. On the other hand, processing time and memory peak are used to evaluate the algorithms cost and verify the theoretical cost functions discussed in Section 3. All experiments were performed on a computer with a Intel Xenon 3.0GHz processor and 24GB of RAM.

Note that F-measure and PRI try to quantify the segmentation quality by distinct means, and are complementary metrics for the same problem. The PRI is defined as [15]:

$$PRI(S, GT) = \frac{1}{T} \sum_{i<j} \left[c_{ij} p_{ij} + (1 - c_{ij})(1 - p_{ij}) \right], \tag{11}$$

where S is the segmentation map being evaluated, GT is the groundtruth segmentation, c_{ij} is the event that pixels i and j have the same label in S, and p_{ij} is the probability that this event occurs in GT, and T is the total number of pixel pairs. This metric tries to quantify the segmentation quality by counting the number of pixels pairs that are correctly grouped in the same image segment, or correctly separated, in comparison with the groundtruth.

The F-measure approaches this segmentation quality evaluation differently, and is defined as the following harmonic mean [15]:

$$F(S, GT) = \frac{2Pr(S, GT)Rc(S, GT)}{Pr(S, GT) + Rc(S, GT)}, \tag{12}$$

where $Pr(\cdot)$ and $Rc(\cdot)$ are the boundary precision and recall, respectively, that

are obtained as follows:

$$Pr(S, GT) = \frac{TP}{TP + FP}, \quad (13) \qquad \text{and} \qquad Rc(S, GT) = \frac{TP}{TP + FN}, \quad (14)$$

where TP , FP and FN are the number of true positives, false positives, false negatives of the segmentation map boundaries matched to the groundtruth map boundaries, respectively. The F-measure measures the segmentation quality by the similarity of the desired boundaries and the obtained boundaries.

As such, while the PRI measures how accurate is the pixel grouping process, the F-measure indicates how accurate are the boundary locations. As these metrics evaluate distinct aspects of an image segmentation, it is possible to have better performance according to one metric and not to the other for a given segmentation. Therefore it is important to analyze both metrics to determinate the overall image segmentation quality.

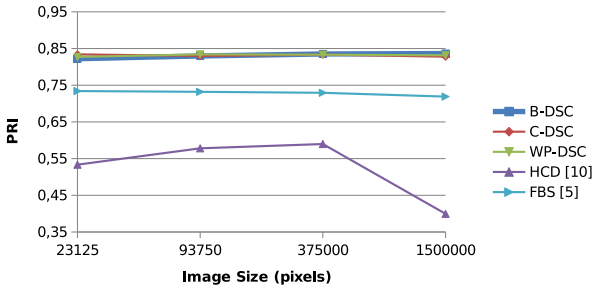
Based on the dataset described above, Table 2 presents a quantitative evaluation of the proposed algorithm decoupling variants presented in Sec. 2.2 - block partitioning (B-DSC), coarse graph compression (C-DSC), waterpixels (WP-DSC) and Mori superpixels (SP-DSC) - compared to some state-of-the-art segmentation methods on the largest image size of the dataset (1000×1500 pixels) in terms of PRI, F-measure and computation time. This comparison demonstrates that the proposed strategy achieves higher segmentation quality with lower computational cost in large images. Note that the B-DSC variant, while being the less accurate of the proposed variants, still produces segmentations of the same quality level as the best state of the art methods, at a lower computation cost. From Table 2 we can also verify that the WP-DSC and the SP-DSC decoupling strategies, that produce all sub-graphs with approximately the same size, tend to be more efficient, achieving the same quality at a lower computational cost, as explained in Section 3. In particular, we recall that the proposed decoupled sub-graph compression (DSC) is an extension of the Stochastic Texture Representation Model (STRM) [14] that emphasizes local optimizations of the segmentation via the decoupling strategy. As a consequence, a full graph

compression (Full-DSC), that performs only the compression stage, but not the decouple and recouple stages, is closely related to the STRM. Since both Full-DSC and the STRM are based on graph analysis, both require high amounts of computational resources and produce segmentations of inferior quality than any of the DSC variants, as can be verified on Tables 1 and 2.

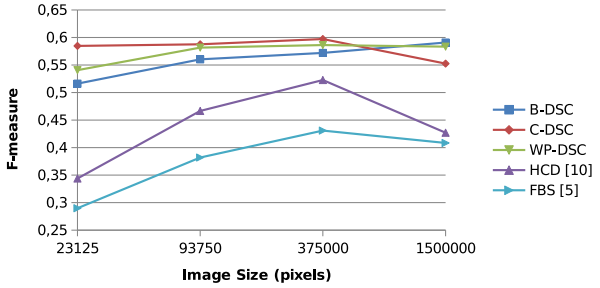
Furthermore, the metrics for the different decoupling methods reported in Table 2 indicate that the quality of the decoupling segmentation is not as relevant as the size of the initial segments obtained. Therefore, applying complex over-segmentation techniques to this initial stage may not improve significantly the quality of the final segmentation. While the C-DSC produces a much more accurate initial decoupling than the SP-DSC and WP-DSC, they all lead to final segmentations with virtually the same quality. With regards to the computation time, not only the C-DSC is more expensive to compute, but also leads to a less efficient graph decoupling, more precisely, the C-DSC variation is 1.4× slower than SP-DSC and 1.9× than WP-DSC. The main reason for this cost increase, as discussed in Section 3, is that the complexity of proposed compression algorithm is polynomial on the size of the sub-graphs, and since $a^x + b^x \leq (a+b)^x$, for any $a, b, x \geq 1$, the minimal computation time will be obtained when all sub-graphs have the same size.

Therefore, it is more advantageous to decouple the initial graph $\mathcal{G}$ in sub-graphs of approximately the same size (number of vertices and edges), than in visually homogeneous regions (that would lead to better over-segmentations), as it can be confirmed experimentally in Table 1. In that comparison, we verify that all decoupling methods produce segmentations with a similar level of quality, which is superior to Full-GC, i.e, not decoupling $\mathcal{G}$. On the other hand, the cost (in terms of time and memory) is more easily controlled if all sub-graphs have the same size, making the superpixel techniques preferable to obtain S_D (the initial decoupling) over more sophisticated strategies (like the coarse graph compression).

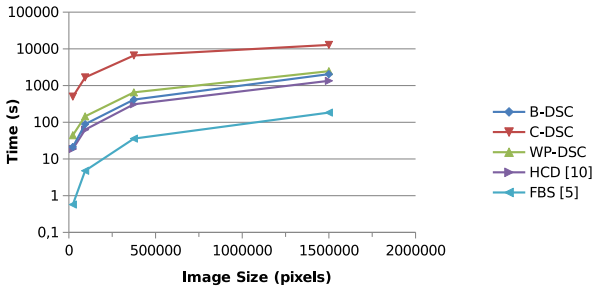
Additionally, Fig. 4 presents an extended comparison of the same methods, for all image sizes available in the proposed dataset. Some trends in terms of



(a) Average PRI comparisson;

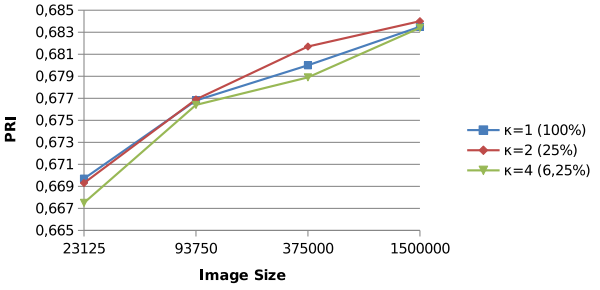


(b) Average F-measure comparisson;

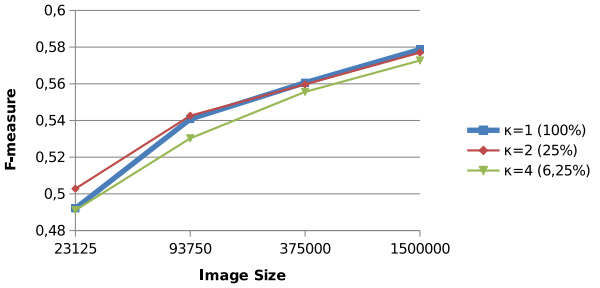


(c) Average Computation time comparisson.

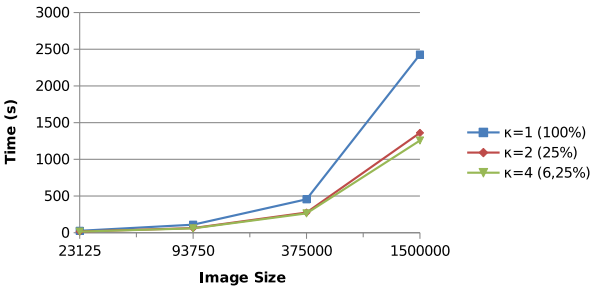
Figure 4: Comparison of quantitative segmentation metrics of the proposed DSC and some state-of-the-art methods, according to image size. The time comparison chart (c) is shown in $\log_{10}$ scale to allow proper visualization of all methods.



(a) Average PRI comparisson;

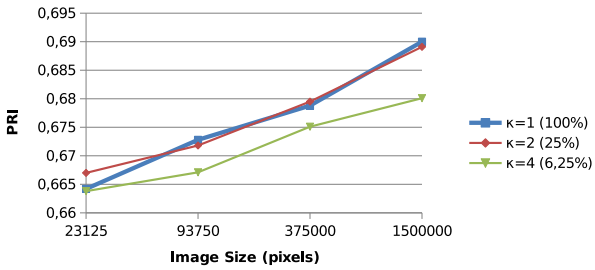


(b) Average F-measure comparisson;

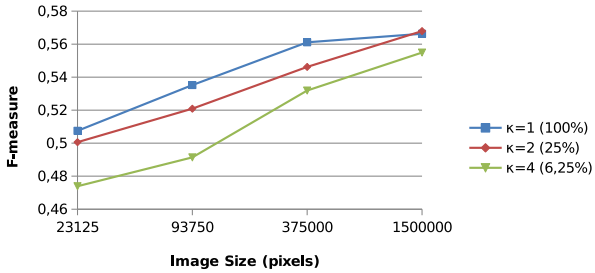


(c) Average Computation time comparisson.

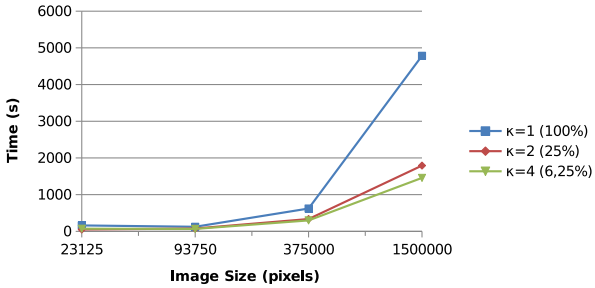
Figure 5: Comparison of quantitative segmentation metrics obtained with the proposed B-DSC, according to image size for different values of κ.



(a) Average PRI comparisson;

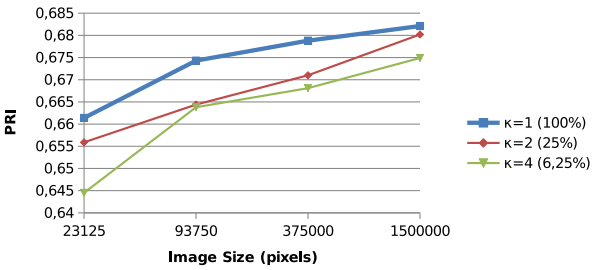


(b) Average F-measure comparisson;

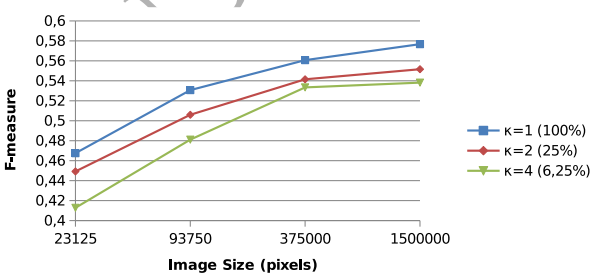


(c) Average Computation time comparisson.

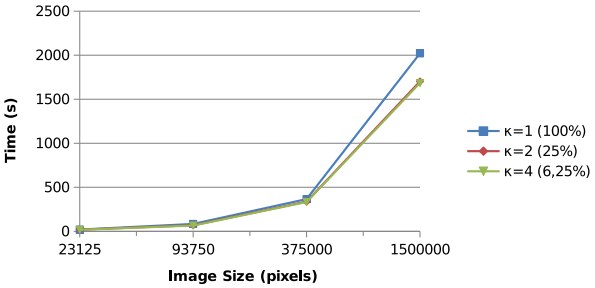
Figure 6: Comparison of quantitative segmentation metrics obtained with the proposed C-DSC, according to image size for different values of κ.



(a) Average PRI comparisson;



(b) Average F-measure comparisson;



(c) Average Computation time comparisson.

Figure 7: Comparison of quantitative segmentation metrics obtained with the proposed WP-DSC, according to image size for different values of κ.

quality (PRI and F-measure) and efficiency (time) can be verified in Fig. 4, suggesting at least two interesting findings. First, it is noticeable that better segmentations tend to be obtained at higher resolutions, which is expected due the additional information available in the image. Second, among the proposed variants of the DSC, B-DSC presented the fastest computation, while the C-DSC presented the slowest, corroborating to the hypothesis that using sub-graphs on equal size and shape lead to a more efficient segmentation computation, even if the graph $\mathcal{G}$ is decoupled into sub-graphs where the cut-set have edges that do not coincide with the image region boundaries.

Moreover, using decoupled sub-graphs that mismatches image region boundaries do not directly affect the segmentation quality based on the evaluated metrics, as the B-DSC produces segmentations os comparable quality to the other decoupling methods for higher resolution images. Although some regional boundary misplacement can be visually verified for the B-DSC (especially in the lower resolutions), the segmentation quality is still superior to the state-of-the-art methods (see Figure8). This also endorses one of the main hypothesis of this work, which is that different decoupling methods employed in the first stage of the proposed method, for recoupling the compressed sub-graphs C_p into a single graph $\mathcal{C}$, and then re-compressing this graph into the globally compressed graph $\mathcal{F}$, produce segmentations that may differ, but have similar quality.

On the other hand, Fig. 5 shows a comparison of segmentation quality and cost for different sub-sampling rate κ values. In this figure only the B-DSC is evaluated, but the same trends are verifiable for the other variants of the proposed method in Figs. 6 and 7. As we can see in the top and middle charts of Fig. 5, the average segmentation quality tends to be higher for larger images, but it is approximately the same for all tested κ for a given image size. The bottom chart shows that although the observed difference may not be significant at lower resolutions, there is a considerable reduction of computation time for larger images when κ is larger. This observations supports one of the main hypothesis of this work, that in the later stages of the segmentation of high resolution images, the regular elimination of texture feature samples does not

(a) C-DSC; (b) WP-DSC; (c) SP-DSC; (d) STRM [14]; (e) HCD [15]; (f) FBS [7].

Figure 8: Visual comparison of the proposed algorithm with the state-of-the-art texture segmentation methods. form left to right, the proposed methods: (a) C-DSC, (b) WP-DSC, (c) SP-DSC, and the state-of-the-art methods: (d) STRM [14], (e) HCD [15], (f) FBS [7].

jeopardize the result significantly, but allows a reduction in the algorithm cost, as discussed in Sec. 3, which is critical for larger images.

Furthermore, Fig. 8 presents a visual comparison of the proposed DSC variants and some state-of-the-art segmentation techniques. Analogously to the quantitative experimental results described above, the visual comparison also demonstrates that the DSC strategy produces image segmentations of superior quality. We also verify that while different decoupling methods will produce very distinct segmentations, they yield similar levels of image segmentation quality. Other interesting finding in this visual analysis is that although the C-DSC numerically presents slightly better boundaries (F-measure) than other methods, it is not always reflected into a visually better image segmentation. It shall be observed that the Stochastic Texture Representation Models (STRM) [14] and the Hierarchical Contour Detection (HCD) [15] methods are prone to oversegment the image, while the Factorization-Based Segmentation (FBS) [7] tends to produce undersegmentation. On the other hand, the proposed DSC methods and its variant algorithms, are able to avoid these errors in most cases.

5. Conclusion

This paper proposed a novel segmentation strategy based on graph compression that not only produces better quality results than the compared state of the art methods, but also consumed considerably less resources (in terms of time and memory) to do so. The proposed method represents the image as a dense graph, which is decoupled to allow efficient processing at a fine scale. Finally, the compressed sub-graphs are recombined into a single connected graph, that yields the segmentation of the image. In order to understand the efficiency of the proposed method, we formulate and discuss the computational complexity in terms of time and memory, showing the improvement of decoupling the image prior to compression the graph.

To evaluate the proposed segmentation strategy experiments were performed in a dataset containing large images (1000×1500), that were collected and la-

beled specifically for this task (i.e. for evaluating the performance of segmentation methods). These experiments compared some state-of-the-art methods with 4 variants of the proposed method, which use distinct ways to decouple the initial graph (Blocks, low-resolution graph compression, waterpixels and superpixels).

The numeric and visual data collected from these tests indicate that, although there may be variations on the image segmentations produced by different decoupling methods, all DSC variants perform better than the compared state of the art methods (PRI = 0.782 and F = 0.543), at a reasonable computing cost. By these experiments, we were also able to confirm two important assumptions of the proposed method. The first one is that, the local optimization of the sub-graphs (with specialized dictionaries) lead to better segmentation than a single global compression, regardless of the decoupling method utilized, at a lower computational cost. Second, using the sub-graphs of regular shape, with approximately the same size, will improve the segmentation efficiency, as the compression complexity grows quadratically on the number of pixel interactions. Consequently, it is viable to exploit the high resolution of large images (i.e., their local redundancy), to improve computational efficiency without a significant loss of the segmentation quality.

As future work, we plan to further investigate parallel implementations of the proposed algorithm, as well as employing the proposed decoupling strategy to improve other image processing algorithms.

[1] J. Scharcanski, Bringing vision-based measurements into our daily life: A grand challenge for computer vision systems, Frontiers in ICT 3 (2016) 3. doi:10.3389/fict.2016.00003.
URL http://journal.frontiersin.org/article/10.3389/fict.2016.00003

[2] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (11) (2001) 1222–1239. doi:10.1109/34.969114.

[3] X. Yuan, F. Xiang, Z. Wang, Semi-automatic object segmentation using colour invariance and graph cuts, Img. Proc., IET 8 (2) (2014) 69–77. doi:10.1049/iet-ipr.2013.0154.

[4] M. Khokher, A. Ghafoor, A. Siddiqui, Image segmentation using multilevel graph cuts and graph development using fuzzy rule-based system, Img. Proc., IET 7 (3) (2013) 201–211. doi:10.1049/iet-ipr.2012.0082.

[5] A. Heimowitz, Y. Keller, Image segmentation via probabilistic graph matching, IEEE Transactions on Image Processing 25 (10) (2016) 4743–4752. doi:10.1109/TIP.2016.2590832.

[6] M. Mignotte, A label field fusion bayesian model and its penalized maximum rand estimator for image segmentation, IEEE Trans. Img. Process. 19 (6) (2010) 1610–1624. doi:10.1109/TIP.2010.2044965.

[7] J. Yuan, D. Wang, A. M. Cheriyadat, Factorization-Based Texture Segmentation., IEEE Trans. Img. Proc. 24 (11) (2015) 3488–97. doi:10.1109/TIP.2015.2446948.

[8] M. Pereyra, S. McLaughlin, Fast unsupervised bayesian image segmentation with adaptive spatial regularisation, IEEE Transactions on Image Processing 26 (6) (2017) 2577–2587. doi:10.1109/TIP.2017.2675165.

[9] Z. jie Zhu, Y. er Wang, G. yi Jiang, Unsupervised segmentation of natural images based on statistical modeling, Neurocomputing 252 (2017) 95 – 101, sI: Dynamic Depth Field. doi:http://dx.doi.org/10.1016/j.neucom.2016.03.117.
URL http://www.sciencedirect.com/science/article/pii/S0925231217306549

[10] D. Dera, N. Bouaynaya, R. Polikar, H. M. Fathallah-Shaykh, Non-negative matrix factorization for non-parametric and unsupervised image clustering and segmentation, in: 2016 International Joint Conference on Neural Networks (IJCNN), 2016, pp. 3068–3075. doi:10.1109/IJCNN.2016.7727589.

[11] A. Wong, J. Scharcanski, P. Fieguth, Automatic skin lesion segmentation via iterative stochastic region merging, IEEE Trans. Inf. Technol. Biomed. 15 (6) (2011) 929 –936. doi:10.1109/TITB.2011.2157829.

[12] D. E. Ilea, P. F. Whelan, Ctex - an adaptive unsupervised segmentation algorithm based on color-texture coherence, IEEE Trans. Img. Process. 17 (10) (2008) 1926–1939. doi:10.1109/TIP.2008.2001047.

[13] M. Krinidis, I. Pitas, Color texture segmentation based on the modal energy of deformable surfaces, IEEE Trans. Img. Process. 18 (7) (2009) 1613–1622. doi:10.1109/TIP.2009.2018002.

[14] R. Medeiros, J. Scharcanski, A. Wong, Image segmentation via multi-scale stochastic regional texture appearance models, Comput. Vision and Img. Understanding (2015) –doi:http://dx.doi.org/10.1016/j.cviu.2015.06.001.

[15] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, Contour detection and hierarchical image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 33 (5) (2011) 898–916. doi:10.1109/TPAMI.2010.161.

[16] H. Min, W. Jia, X.-F. Wang, Y. Zhao, R.-X. Hu, Y.-T. Luo, F. Xue, J.-T. Lu, An Intensity-Texture model based level set method for image segmentation, Pattern Recognition 48 (4) (2015) 1547–1562. doi:10.1016/j.patcog.2014.10.018.

[17] Q. Wu, Y. Gan, B. Lin, Q. Zhang, H. Chang, An active contour model based on fused texture features for image segmentation, Neurocomputing 151 (2015) 1133–1141. doi:10.1016/j.neucom.2014.04.085.

[18] S. Alpert, M. Galun, A. Brandt, R. Basri, Image segmentation by probabilistic bottom-up aggregation and cue integration, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (2) (2012) 315–327. doi:10.1109/TPAMI.2011.130.

[19] T. H. Kim, K. M. Lee, Learning full pairwise affinities for spectral segmentation, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, 2010, pp. 2101–2108. doi:10.1109/CVPR.2010.5539888.

[20] Z. Li, X. M. Wu, S. F. Chang, Segmentation using superpixels: A bipartite graph partitioning approach, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, 2012, pp. 789–796. doi:10.1109/CVPR.2012.6247750.

[21] R. Hettiarachchi, J. Peters, Voronoï region-based adaptive unsupervised color image segmentation, Pattern Recognition 65 (2017) 119 – 135. doi:http://dx.doi.org/10.1016/j.patcog.2016.12.011.
URL http://www.sciencedirect.com/science/article/pii/S0031320316303983

[22] S. Yin, Y. Qian, M. Gong, Unsupervised hierarchical image segmentation through fuzzy entropy maximization, Pattern Recognition 68 (2017) 245 – 259. doi:http://dx.doi.org/10.1016/j.patcog.2017.03.012.
URL http://www.sciencedirect.com/science/article/pii/S0031320317301115

[23] A. Li, X. Wang, K. Yan, C. Li, D. Feng, Multilevel affinity graph for unsupervised image segmentation, in: 2016 IEEE International Conference on Image Processing (ICIP), 2016, pp. 1264–1268. doi:10.1109/ICIP.2016.7532561.

[24] G. Mori, X. Ren, A. Efros, J. Malik, Recovering human body configurations: combining segmentation and recognition, in: Proc. 2004 IEEE Computer Society Conf. Computer Vision and Pattern Recognition, Vol. 2, 2004, pp. II–326 – II–333 Vol.2. doi:10.1109/CVPR.2004.1315182.

[25] V. Machairas, M. Faessel, D. C.-P. na, T. Chabardes, T. Walter, E. Decencière, Waterpixels, IEEE Transactions on Image Processing 24 (11) (2015) 3707–3716. doi:10.1109/TIP.2015.2451011.

[26] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Süsstrunk, Slic superpixels compared to state-of-the-art superpixel methods, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (11) (2012) 2274–2282. doi:10.1109/TPAMI.2012.120.

[27] S. Z. Li, Markov Random Field Modeling in Image Analysis, 3rd Edition, Springer Publishing Company, Incorporated, 2009.

[28] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905. doi:10.1109/34.868688.
URL http://dx.doi.org/10.1109/34.868688

[29] P. Soille, Morphological Image Analysis: Principles and Applications, 2nd Edition, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[30] R. Medeiros, A. Wong, J. Scharcanski, Scalable texture segmentation via decoupled sub-graph compression, Pre-printPreprint.

[31] J. Malik, S. Belongie, T. Leung, J. Shi, Contour and texture analysis for image segmentation, Int. J. Comput. Vision 43 (1) (2001) 7–27. doi:10.1023/A:1011174803800.
URL http://dx.doi.org/10.1023/A:1011174803800

[32] M. Varma, A. Zisserman, A statistical approach to material classification using image patch exemplars, IEEE Trans. Pattern Analysis and Machine Intelligence 31 (11) (2009) 2032 –2047. doi:10.1109/TPAMI.2008.182.

[33] P. K. Harmer, M. A. Temple, M. A. Buckner, E. Farquahar, Using differential evolution to optimize 'learning from signals' and enhance network security, in: Proc. Annu. Conf. on Genetic and Evolutionary Computation, ACM, New York, NY, USA, 2011, pp. 1811–1818. doi:10.1145/2001576.2001819.
URL http://doi.acm.org/10.1145/2001576.2001819

[34] F. J. Aherne, N. A. Thacker, P. I. Rockett, The bhattacharyya metric as an absolute similarity measure for frequency coded data, Kybernetika 32 (1998) 1–7.

[35] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, Third Edition, 3rd Edition, The MIT Press, 2009.

Rafael Sachett Medeiros received the bachelor degree in computer science from the Federal University of Pelotas in 2011, and the M.Sc. degree in computer science from the Federal University of Rio Grande do Sul (UFRGS) in 2013. He is currently pusuing his PhD in computer science at UFRGS. His research interest are computer vision, feature engineering and unsupervised learning.

Alexander Wong, P.Eng., is currently the Canada Research Chair in Medical Imaging Systems, co-director of the Vision and Image Processing Research Group, and an associate professor in the Department of Systems Design Engineering at the University of Waterloo. He had previously received the B.A.Sc. degree in Computer Engineering from the University of Waterloo, Waterloo, ON, Canada, in 2005, the M.A.Sc. degree in Electrical and Computer Engineering from the University of Waterloo, Waterloo, ON, Canada, in 2007, and the Ph.D. degree in Systems Design Engineering from the University of Waterloo, ON, Canada, in 2010. He has published over 300 refereed journal and conference papers, as well as patents, in various fields such as computational imaging, artificial intelligence, computer vision, and multimedia systems.

Jacob Scharcanski is a Professor in Computer Science at the Federal University of Rio Grande do Sul (UFRGS), Brasil. He holds a cross appointment with the Department of Electrical Engineering at UFRGS, and also is an Adjunct Professor with the Department of Systems Design Engineering, University of Waterloo, Ontario, Canada. He has authored and co-authored over 150 refereed journal and conference papers, books and book chapters on imaging and measurements. In addition to his academic publications, he has several technology transfers to the private sector. Presently, he serves as an Associate Editor for two journals, and has served on dozens of International Conference Committees. Professor Scharcanski is a licensed Professional Engineer, Senior Member of the IEEE, IEEE IMS Distinguished Lecturer (2015-2017). His areas of expertise are Image Processing, Pattern recognition, Imaging Measurements and their applications.