Graph-based image gradients aggregated with random forests

Raquel Almeida, Ewa Kijak, Simon Malinowski, Zenilton K.G. Patrocínio Jr, Arnaldo Araújo, Silvio J.F. Guimarães

Graph-based image gradients aggregated with random forests

Raquel Almeida[(a)] Ewa Kijak[(a)] Simon Malinowski[(a)]
Zenilton K. G. Patrocínio Jr[(b)] Arnaldo A. Araújo[(c)] Silvio J. F. Guimarães[(b)]

[(a)] *Linkmedia, IRISA, Univ. Rennes, France*

[(b)] *Image and Multimedia Data Science Laboratory, Pontifícia Universidade Católica de Minas Gerais, Brazil*

[(c)] *Computer Science Department, Universidade Federal de Minas Gerais, Brazil*

Abstract

Gradient methods subject images to a series of operations to enhance some characteristics and facilitate image analysis, usually the contours of large objects. We argue that a gradient must show other characteristics, such as minor components and large uniform regions, particularly for the image segmentation task where subjective concepts such as region coherence and similarity are hard to interpret from the pixel information. This work extends the formalism of a previously proposed graph-based image gradient method that uses edge-weighted graphs aggregated with Random Forest (RF) to create descriptive gradients. We aim to explore more extensive input image areas and make changes driven by the RF mechanics. We evaluated the proposals on the edge and segmentation tasks, analyzing the gradient characteristics that most impacted the final segmentation. The experiments indicated that sharp thick contours are crucial, whereas fuzzy maps yielded the worst results even when created from deep methods with more precise edge maps. Also, we analyzed how uniform regions and small details impacted the final segmentation. Statistical analysis on the segmentation task demonstrated that the gradients created by the proposed are significantly better than most of the best edge maps methods and validated our original choices of attributes.

1 Introduction

Image segmentation is a computer vision task aiming to group pixels into regions. It is a complex task based on abstract concepts such as region coherence, perceptual similarity, and composition of objects and scenes by aggregated parts. We could define the region coherence in terms of uniformity, continuity, and contrasts that create well-defined boundaries between two adjacent regions [1]. Perceptual similarity and composition are subjective, but ideally, they would reflect our perception by hierarchically recognizing parts, objects, and concepts [2].

This ideal could be why hierarchical methods for segmentation have remained popular since their creation [3, 4, 5, 6]. Hierarchical methods build in their structure different detail levels, providing easy navigation and merging operations to build more semantically significant objects from lower-level instances. The most well-known hierarchical method is the hierarchical watershed [3], which extends the morphological watershed [7] and creates a sequence of segmentations from image magnitudes.

Traditionally, the magnitudes are extracted by measuring dissimilarities in the color space (*e.g.*, RGB, Lab) [8, 9] or in the gray-scale representation of the images [10, 11]. However, these approaches may produce a significant variation of absolute values, making it hard to determine which values compose a coherent region [12]. An alternative is to preprocess the images by applying gradient operators to emphasize desirable characteristics, such as the object's contours. Well-known gradient operators based on kernel filters for local variation, such as Laplacian and Sobel, used to be the preferable operators [13, 14]. However, learned edge maps, as in the Structured Edge Detection (SED) method [15], were proven to produce better results [8] and became more popular [16, 17].

Our previous work [18] argued that although the contours are essential for contrasting adjacent regions, other characteristics reflecting uniformity and continuity are desirable for the segmentation task. We proposed a graph-based gradient operator that produced gradients with firm contours of the objects and other characteristics such as minor components, textures, and large uniform regions. We demonstrated that these gradients used as input for the watershed method produced better segmentation results than Laplace, Sobel, and SED.

Our graph-based image gradient (GIG) method uses edge-weighted graphs aggregated with Random Forest (RF) [19] as an image gradient operator trained on the edge detection task. The motivation behind this

approach is four-fold: (i) **exploit the spatial image domain preserved in the form of a grid graph:** typically, we define graphs on the image domain with a structured grid adjacency relation, and the set of elemental graph components, called vertices, represents the pixels of the image; (ii) **strengthen the representation with the graph relational features:** we represent the adjacency relation by edges connecting neighboring vertices, and a weighting function could be associated to represent local variation. Therefore, the graph operator contains the structured spatial information and measurements of relational dissimilarities; (iii) **suppress noise and encourage large consistent regions with RF:** the RF algorithm, through attribute selection and implicit regularization [20], can reinforce desirable characteristics and mitigate some strong responses created by the local difference measurements on the graph operator; and (iv) **maintain the analyses on the discrete space:** each vertex description represents an entire neighboring region associated with a single label; therefore, we avoid training in a complex structured output space, like in [15].

Our motivation goes beyond a good performance in a task. It also relies on a proposal of a machine learning framework operating on graphs, a topic of interest due to its capacity to represent multivariate information and the possibility of multiple applications, such as classification or clustering. A significant challenge is that most machine learning algorithms require regular inputs to operate, including RF. This requirement is inherently opposed to the unconstrained nature of graphs (*e.g.*, no clear beginning or end, connected vertices are not necessarily close). The GIG representation considered the type of graph, its proximity to the original data, and the expected results, allowing us to process the graphs as regular data with a fast machine learning algorithm and avoid the long computations of graph networks.

In this work, we extend the formalism to exploit better the relationships modeled by graphs, mainly focusing on the RF mechanics and limitations. Namely, we propose:

1. **Region adjacency graphs:** Extends the formalism from the bijective correspondence of vertices and pixels in GIG to vertices and a set of regions produced by an initial segmentation into image super-pixels. This approach could reduce the number of data points during training and impact the gradient and the computational cost;
2. **Positional features:** Vertices corresponding to the pixels on the image's border have an incomplete set of neighbors. In GIG, they received padding values that disregarded the missing value's position on the regular representation. In the positional feature approach, we only take the vertices with a complete set of adjacent vertices to avoid changing the feature connotation during training;
3. **Unique paths:** The regular representation of the grid graph in GIG is redundant, meaning not all values are unique as the vertices on the grid path share some neighbors. Our unique path approach considers only the first instance of a neighboring vertex in a region, reducing the representation's size and allowing the region's expansion.

Besides the extended formalism, we add evaluations on edge detection and comparisons with deep learning approaches [21, 22] on edge detection and region segmentation. We aim to advocate for our assertion that a good gradient for image segmentation should present more than precise object contours by comparing it with even more accurate edge maps.

We organized this work as follows: Section 2 presents the theoretical background and Section 3 the methodology. The experimental setup and results are in Section 4, followed by a discussion in Section 5 and the conclusions in Section 6.

2 Theoretical background

An undirected *graph* $G = (V, E)$ consists of a finite non-empty set of *vertices*, denoted by V, and a finite set of *edges* $E = \{\{u, v\} \mid u, v \in V\} \subseteq V \times V$. The set E induces a unique *adjacency* relation Γ on V, which associates $u \in V$ with $\Gamma(u) = \{v \in V | (u, v) \in E\}$. In the image graph, the set of vertices represents the pixels of the image, and the edges represent the connections between them. Usually, we define the adjacency relation Γ by a structured component in a grid form, such as 4- or 8-adjacency. We define *vertex attributes* as vertex functions $f : V \to \mathbb{R}$ that map low-level image features to a vertex v.

An *edge-weighted* graph is denoted by $(V, E, \mathcal{F})$, in which $\mathcal{F} : V \times V \to \mathbb{R}$ is a dissimilarity function that weights the edges of G. We could assign multiple functions to the set of edges $\mathcal{F}$ and vertices f. This work uses $\mathcal{F}_{\mathbf{euc}}(u, v) = \sqrt{(f_g(u) - f_g(v))^2}$, where the vertex function f_g maps the gray-scale magnitudes. Also, we apply the descriptor proposed in [23] as a vertex function f to map the pixel's color space and color gradients.

The edge weights may represent the local variation around a vertex and serve as an image gradient operator bounded by the adjacency relation. The function defined for $\mathcal{F}_{\mathbf{euc}}$ may have a strong response to noise, as in the case of many spatial filters based on local dif-

Algorithm 1: Regular GIG and regular GIG-RAG

Input : $G = (V, E, \mathcal{F})$: an edge-weighted graph, a flag *isTrainSet* indicating if G is a train instance and the expected representation size p.

Output: $\mathcal{X}_G = \{(\mathbf{X}_1, \mathbf{Y}_1), \ldots, (\mathbf{X}_{|V|}, \mathbf{Y}_{|V|})\}$: set of regular representations of the graph vertices attributes $\mathbf{X}_v \in \mathbb{R}^p$ and its associated labels $\mathbf{Y}_v$ in case when G is a train instance.

Function *getAttributes(v)*:

if *isTrainSet* **then**
  $\mathbf{X} = \mathbf{ones}[p+1]$
  $\mathbf{Y} = \mathbf{getLabel}(v)$
  $\mathbf{X}[p+1] \leftarrow \mathbf{Y}$ // at $p+1$ position
else $\mathbf{X} = \mathbf{ones}[p]$
$colorFeatures \leftarrow \mathbf{getDescriptors}(v)$
$\mathbf{X} \leftarrow colorFeatures$
$firstNeighbors \leftarrow \{u \mid \forall u \in \Gamma(v)\}$
$secondNeighbors \leftarrow \{q \mid \forall q \in \Gamma(u)$ **and** $\forall u \in firstNeighbors\}$
$\mathbf{X} \leftarrow \{\mathcal{F}(u, v) \mid \forall u \in firstNeighbors\}$
$\mathbf{X} \leftarrow \{\mathcal{F}(q, u) \mid \forall q \in secondNeighbors$ **and** $\forall u \in firstNeighbors\}$
return X

Main:

for *vertex v* ***in*** V **do**
  $\mathbf{X}_v = getAttributes(v)$
  $\mathcal{X} \leftarrow append(\mathbf{X}_v)$
end
return $\mathcal{X}$

ferences. The RF acting as a regularizer can diminish the noise, mitigate any eventual poor topology choice and accentuate strong connections.

RF [19] is a non-parametric ensemble method for supervised classification and regression that relies on randomizing selected data and features and has extensive practical uses in many domains. Some authors [20, 24] believe that randomness performs as an implicit regularization process, promoting consistency [24] and noise suppression [20]. The RF model consists of randomized independent trees, each trained with bootstrap samples of the input labeled data $\mathcal{D}$. Applying RF to edge-weighted graph representation implies deriving regular inputs from the graph, which we will detail in Sec. 3.

3 Graph-based image gradients

The main challenge in this framework concerns the strategy to parse the data and create the RF's regular input without losing too much information. In our previously proposed representation, GIG [18], the strategy depicted each vertex $v \in V$ of the edge-weighted graph $G = (V, E, \mathcal{F})$ as a vector $\mathbf{X}_v$ with dimension $p = |\mathbf{G_{att}}|$, where $\mathbf{G_{att}}$ is a set of selected attributes. The selection belonged to two categories: (i) vertex attributes ($\mathbf{X}_f$), representing the vertices functions; and (ii) edge weights ($\mathbf{X}_\mathcal{F}$), representing the weight value in every edge v and any adjacent $u \in \Gamma(v)$. Thus, $\mathbf{G_{att}} = \{\mathbf{X}_f \mathbf{X}_\mathcal{F}\}$.

Algorithm 1 presents the steps to create the regular GIG representation $\mathcal{X}_G$ for an edge-weighted graph G. We repeat the procedure for all graphs in the training set and concatenate the $\mathcal{X}_G$ outputs to make the RF training input $\mathcal{D}$. Therefore, $\mathcal{D} = ((\mathbf{X}_1, \mathbf{Y}_1), \ldots, (\mathbf{X}_M, \mathbf{Y}_M))$, where M is the total number of vertices in the training set. We train the RF on the edge detection task: $\mathbf{Y} \in \{0, 1\}$, and all M entries have a unique discrete label. In the test step, we make the regular representation for each graph in the validation/test set and individually subject them to the estimations of the RF. The final estimated values are mapped back to the image coordinates as gradients.

We now extend the GIG formalism to propose strategies to explore more extensive input image areas without adding redundancy and make changes driven by the RF mechanics.

3.1 GIG defined on region adjacency graphs

The proposed region adjacency graph approach (GIG-RAG) reduces the number of vertices by presenting regions of grouped pixels in the set of vertices instead of a single pixel as in GIG. The GIG-RAG requires a strategy to group the pixels and considerations for edges, weights, and label attribution.

Given an edge-weighted grid graph $G_I = (V, E, \mathcal{F})$ and a list of grouped pixels into regions $S_I = \{r^1, \ldots, r^R\}$ for an image I, the *edge-weighted RAG graph* G_{rag} has one vertex for each labeled region in S_I, thus $V_{rag} = \{v_l \,|\, l \in \{1, R\}\}$. There is an edge between two vertices in V_{rag} if an edge connects two vertices in the original grid graph G_I, hence: $E_{rag} = \{\{v_p, v_q\} \,|\, v_p, v_q \in V_{rag} \wedge p \neq q \wedge \exists \{u, v\} \in E \,|\, u \in r^p \wedge v \in r^q\}$. The set E_{rag} induces a unique adjacency relation on V_{rag}, which associates $u_q \in V_{rag}$ with $\Gamma(u_q) = \{v_p \in V_{rag} | (v_p, u_q) \in E_{rag}\}$. For the weighting function, we average the edges' weights in G_I, $\mathcal{F}_{rag}(u_q, v_q) = \mathbf{mean}\{\mathcal{F}(u, v) \,|\, \forall \{u, v\} \in E \,|\, u \in r^p \wedge v \in r^q\}$.

Finally, for label attribution in GIG-RAG, we take a majority vote of the pixels within a region to determine the region label. Also, because we have multiple vertices within each region, we do not use the vertex attributes $\mathbf{X}_f$; thus, GIG-RAG attributes are only the edge weights $\mathbf{X}_{\mathcal{F}_{\mathbf{rag}}}$ of a regular number of closest adjacent regions.

3.2 Positional features

RF is an ensemble of multiple decision trees in which each independent tree takes local decisions to split the

data considering a combination of features. The position of the features is an essential factor, as the model would assume that any subsequent data in that specific position would represent the same feature. In GIG, we added padding values to the vertices that do not have all neighbors in the grid path, disregarding the position that the missing value would assume if present. In GIG-Positional, we will take only the vertices with a complete set of adjacent vertices, creating a more regular representation for training.

3.3 Unique path

We tackle the redundancy created when we transpose the edge-weighted graph to a regular representation for our final proposal. In Algorithm 1, when we take the first and second adjacent neighbors, we inevitably cast repeated values to the regular representation as many vertices share some neighbors within the grid path. While this redundancy is not necessarily a problem, out of the 64 values obtained with two levels of neighbors with an 8−adjacency relation, only 24 of these values are unique. In the GIG-Unique approach, we will consider only the first instance of a neighboring vertex within a region. Removing the redundant values allows the expansion of the region of analysis while maintaining a similar-sized representation.

We propose three variations with an 8−adjacency relation: (i) two levels of neighbors, a 5×5 grid on the original image, 24 values instead of 64; (ii) three levels of neighbors, a 7×7 grid, 48 values instead of 512; and (iii) four levels of neighbors, a 9×9 grid, 80 values instead of 4096. For all variations, the regular representation has the same attributes as GIG, $\mathbf{G_{att}} = \{\mathbf{X}_V \mathbf{X}_\mathcal{F}\}$, but different dimensions for the edge's attributes.

4 Experiments

We performed the experiments in three stages (illustrated in Fig. 1): (i) create the edge-weighted graph gradient operator from the input image; (ii) train the RF on the edge detection task to obtain the gradients; and (iii) evaluate the quality of the gradients on the segmentation task.

We chose the Berkeley Segmentation Dataset and Benchmark (BSDS500) [25] because it proposes edge detection and segmentation labels. It contains 500 RGB images (200 train, 100 validation, and 200 test) of the same size. Each image has multiple labels performed by different annotators; thus, we applied a majority vote on the edge task to get a single label.

We evaluated GIG[1], GIG-RAG, GIG-Positional, and GIG-Unique, all with an 8-adjacency relation and without a combination of methods. The low-level descriptor discussed in Sec. 3 has 13 dimensions ($|\mathbf{X}_V| = 13$), therefore, for GIG and GIG-Positional $p = 77$. For the GIG-Unique representation, we use the terminology GIG-$\mathcal{X}$-Unique, with $\mathcal{X}$ in $\{24, 80\}$ indicating the number of unique values (thus $|\mathbf{X}_\mathcal{F}|$), leading therefore to $p = \{37, 93\}$, respectively. For GIG-RAG, $p = 64$ (no vertex attributes), and we use the terminology GIG-RAG-$\mathcal{R}$ to indicate the number of desired regions of grouped pixels, where $\mathcal{R}$ is in $\{1\text{k}, 5\text{k}, 10\text{k}, 50\text{k}\}$. For completeness, we also evaluate GIG-Edge, the GIG variant considering only edge attributes, with $p = 64$.

We propose an initial segmentation into image super-pixels to group the pixels in the GIG-RAG strategy. We tested with a well-consolidated super-pixel method called *simple linear iterative clustering* (SLIC) [26]. SLIC is an iterative method, which clusters the pixels with the closest center, initially distributed in a regular grid, evaluates the similarity within a cluster, and recalculates the centers until convergence. SLIC execution is fast, easy to set the parameters and the number of produced regions is the closest approximation to the number of desired regions passed as a parameter.

In order to evaluate the impact of the super-pixel quality in the GIG-RAG strategy, we added some comparisons with a more modern super-pixel method: the *superpixel segmentation with fully convolutional networks* (SpixelFCN) [27]. The SpixelFCN is a deep network trained to assign each pixel of an image, initially partitioned into a regular grid, to one of its neighboring grids. The network is an auto-encoder, in which the encoder learns the features, and the decoder aims to group pixels with similar features and enforce compactness. SpixelFCN is easy to set and overall produces better regions, but as in many deep network methods, it is limited in the number of created regions since the regular grid has a fixed size. To increase the number of super-pixels, one should increase the scale of the input image.

For the RF, we used the Random Forest Regressor included in the *scikit-learn* Python package [28], which provides a parallelized implementation over the trees. For the parameters, we performed a grid search on the number of trees, the bootstrap sample size, and the number of sampled features for the split, using the validation set and the original GIG representation. The selected parameters after our search are $\#trees = 500$,

[1] Gradient computation code and model available at: `https://github.com/RaquelAlmeida/GIG.git`

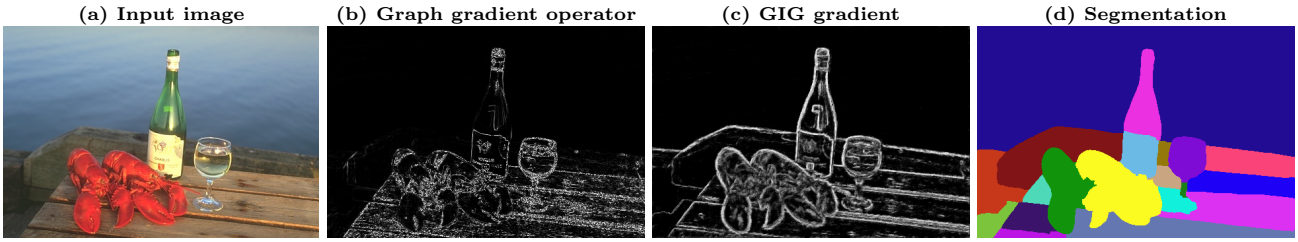

Figure 1: The pipeline illustrated with images created at each step. We present a challenging input image (a) with transparent materials, a body of water, object occlusion, and different objects with similar colors and patterns. The graph gradient operator (b) is an image projection of the graph. The gradient in (c) is the RF predictions for the GIG method, and the segmented image (d) is the Hierarchical watershed output with 20 regions. Best viewed in color.

$samples = 25\%$, $\#features = \log_2(p)$.

The final step in our experimental pipeline is to evaluate the quality of the gradients. We propose to apply the gradients as input to the hierarchical watershed method (HWS) [29], whose performance is dependent on the gradient input (discussion in Sec. 5). Because the watershed is hierarchical, we must pass the desired number of regions to obtain the final segmentation. We used the *Higra* library [30] for the watershed implementation.

We present our quantitative analysis for edge detection and segmentation tasks, using the F-measure for the precision-recall metrics for boundaries and regions. We also show the Probabilistic Rand Index (PRI) [31] measure, a metric that ponders the multiple ground-truths for the segmentation interpretation. Results are presented in terms of the optimal dataset scale (ODS), optimal image scale (OIS), and average precision (AP) through all scales. The scales for the boundaries are different thresholds applied to the edge maps to create a binary image and, for the regions, the desired number of segmented regions.

4.1 Results on edge detection

First, we want to evaluate each proposed strategy's impact on the representation performance regarding the quality of the edge maps (illustrated in Fig. 2(f)-(k)). Table 1 presents the relevant results in the validation set for the edge detection task. We omitted some similar values in Tables 1-3 to avoid repetition. All variations of the GIG-X-Unique presented similar gradients (thin contours and discreet textures) and results (less than 1.5% difference in all metrics). Moreover, all the GIG-Unique strategies had similar score metrics to the original GIG, indicating that there is not much gain in expanding the region of analysis and that the GIG redundancy does not compromise the RF generalization. The GIG-Positional results indicated that the feature position is, in fact, an essential factor during training,

Table 1: F-score for boundaries in terms of optimal dataset scale (ODS), optimal image scale (OIS) and average precision (AP) through all scales (perfect scores=1). Executed on the validation set.

Method	ODS	OIS	AP
GIG	0.623	0.651	0.619
GIG-RAG-600 (SLIC)	0.441	0.471	0.461
GIG-RAG-1k (SLIC)	0.472	0.502	0.463
GIG-RAG-5k (SLIC)	0.522	0.546	0.505
GIG-RAG-10k (SLIC)	0.542	0.566	0.541
GIG-RAG-50k (SLIC)	0.593	0.623	0.587
GIG-RAG-600 (SpixelFCN)	0.498	0.525	0.448
GIG-RAG-1k (SpixelFCN)	0.509	0.538	0.474
GIG-RAG-5k (SpixelFCN)	0.553	0.581	0.562
GIG-RAG-10k (SpixelFCN)	0.546	0.571	0.554
GIG-RAG-50k (SpixelFCN)	-	-	-
GIG-Unique-24	0.618	0.645	0.621
GIG-Unique-80	0.615	0.640	0.619
GIG-Edge	0.605	0.611	0.599
GIG-Positional	**0.712**	**0.727**	**0.729**

and the edge maps created have the best performance on the task among all the compared proposals. GIG-Edge shares the number of regions with GIG while not considering the vertex attributes as GIG-RAG, and the results showed the importance of vertex attributes in the gradients and the score.

We observed that the GIG-RAG strategy considerably reduced the training time proportionally to the number of regions. Nevertheless, it also reduced the performance of the edge detection task. As shown in Table 1, SpixelFCN has a slight advantage over SLIC super-pixels, but we were limited on computational resources to create a larger number of regions. For instance, to make 50k regions with SpixelFCN, one must work with images scaled to ~ 9 times the original size. As illustrated in Fig. 3, the computed gradients vary with the number of regions, fewer regions create larger super-pixels, and the predicted label is applied to a larger area creating gradients of regions instead of contours. Increasing the number of regions creates more

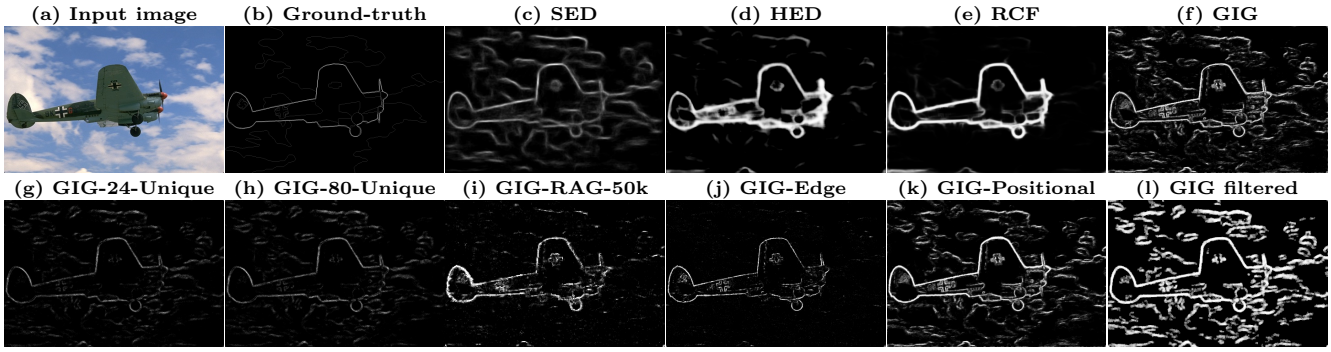

Figure 2: Gradient computations for the input image in (a). SED, HED, and RCF present reinforced contours of the main objects but almost no details; also, most contours are fuzzy, particularly for SED and HED. GIG computes enhanced borders for large and small objects and firm representations of textures. GIG-Unique gradients present thin contours close to the ground-truth and discreet textures (g, h). GIG-RAG gradient with 50k regions computed with SLIC presents a contour-oriented gradient with little texture information (i), the same is true for the GIG-Edge variation (j). GIG-Positional gradients (k) have slightly stronger borders for the main objects than GIG. In (l), the GIG gradient filtered by the morphological opening operation presenting thicker contours.

contour-oriented gradients, which is crucial for the edge detection task.

Table 2 presents the best of our proposed methods, GIG and GIG-Positional. We compared with some leading deep methods on the task: the Holistically-Nested Edge Detection [21] (HED, illustrated in Fig.2(e)) and the Richer Convolutional Features for Edge Detection [22] (RCF, illustrated in Fig.2(f)). Both HED and RCF produced better edge maps (F-measures reported by the authors). Still, they required considerably longer training times (measured using a GPU during $10,000$ iterations on the required augmented dataset, following [21]).

We also included results from the Structured Edge Detection method [15] (SED, illustrated in Fig. 2(d)), which formalism is parallel to ours. Briefly, SED expands the RF formalism for image processing. However, to map the similarity in the structured labels of patches in the image, SED creates an intermediary reduced space, reducing the computational cost by avoiding the calculation of a continuous variance. By contrast, we made structured inputs with the image graphs to attribute a single discrete label. SED, GIG, and GIG-Positional were all trained using the same CPU, with parallelized computation over the trees in 8 CPU cores, wherein the SED's RF is composed of 8 trees and ours of 500. The training time reflects the gain of the structure input on GIG instead of the structured output on SED. Furthermore, the GIG-Positional scores were comparable to SED. For the inference time, all methods took only a fraction of a second for each image, whereas RCF and ours were slightly faster.

4.2 Results on segmentation

We evaluated the segmentation task for all the proposed GIG variants, the deep methods, and SED. In Table 3, we can see that the worst metrics are for the GIG-RAG with a small number of regions (1k and 5k) computed from SLIC. The GIG-RAG gradients computed from SpixelFCN and SLIC with a larger number of regions (10k and 50k) perform like some of the best methods on edge detection (SED and GIG-Positional). HED and GIG-Edge have similar performance on the task, whereas GIG-Edge has an advantage on the PRI metric and the AP. GIG performs better than both in all metrics. GIG-Unique under-performs compared to GIG, except for the F-measure dataset scale, meaning that we could choose a certain number of segmented regions and have more consistent results.

Finally, we have the RCF results that outperformed GIG and the others proposed in all metrics. The gradients produced by GIG and RCF are very different, from the level of details to the thickness of the contours. To investigate the thickness factor, we applied the operation *opening*—a well-known mathematical morphology filtering operation, consisting of one *erosion* to remove small regions followed by one *dilation* to increase object boundaries—in GIG gradients to expand the contours. We used the erosion operation with a kernel 3×3 to avoid enlarging small points followed by a 4×4 dilation kernel. We illustrate the result in Figure 2(l), where the GIG gradient presented thicker contours while retaining most of its details. As shown in Table 3, this operation resulted in better segmentation, indicating that thick contours are crucial to the task. Also, despite the RCF results remaining generally better, the GIG

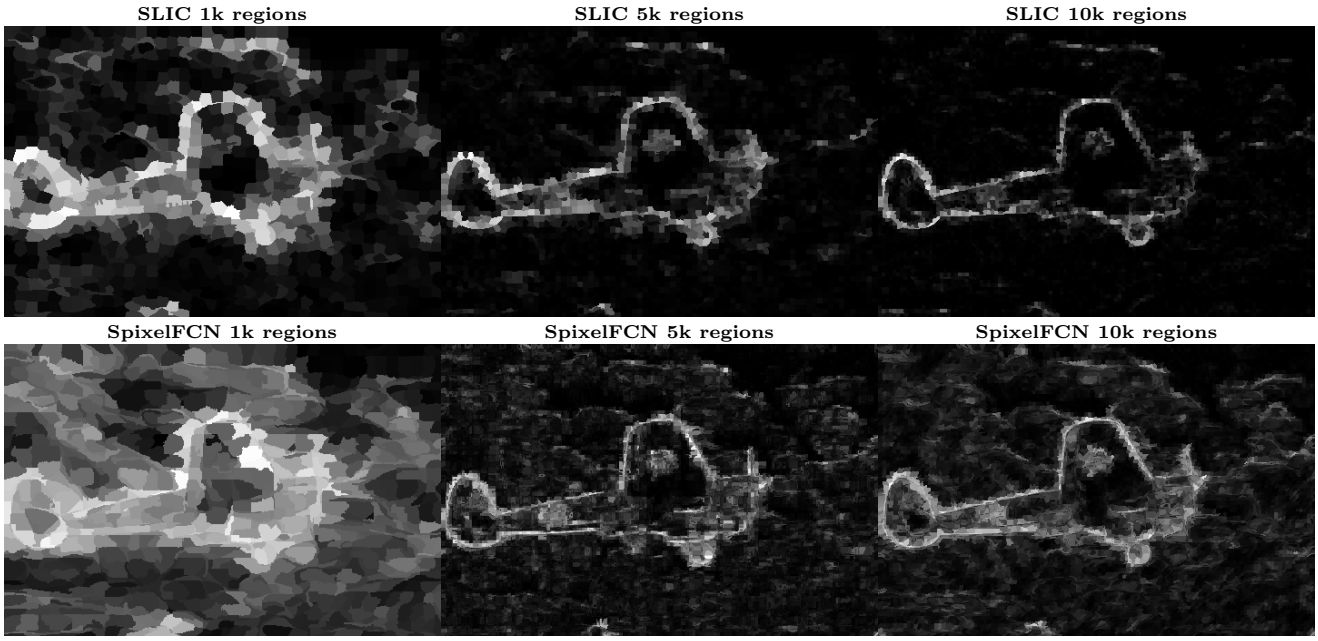

Figure 3: Illustration of the gradient computations for the GIG-RAG strategy comparing the outputs of both tested superpixel algorithms. Gradients from SLIC present more apparent emphasis on the contours, while the ones from SpixelFCN preserve more texture information.

Table 2: Comparison of the best proposed methods on the edge detection task with some of the well-acknowledged methods on the dataset. We present the F-scores for boundaries in terms of optimal dataset scale (ODS), optimal image scale (OIS) and average precision (AP) through all scales, the training and inference time (per image) for all compared methods. Perfect scores=1. Executed on the test set.

	F-measure boundaries			Train time	Inference time
Method	ODS	OIS	AP	(hh:mm:ss)	(s/image)
SED [15]	0.712	0.724	0.750	03:53:18	0.452
HED [21]∗	0.782	0.804	0.833	11:03:42	0.215
RCF [22]∗	**0.811**	**0.830**	**0.947**	10:43:25	**0.141**
GIG	0.635	0.661	0.648	**00:09:18**	0.179
GIG-Positional	0.720	0.748	0.739	00:11:22	0.167

∗ Trained using GPU and F-score as reported by the authors.

filtered representation outperforms RCF in the AP. It is comparable or better in the PRI metrics, which ponder areas without consent among the annotators, such as the small details better captured in GIG. We illustrate the segmentations with highlights on some critical areas in Fig. 4.

We performed statistical analysis for GIG to validate the segmentation results and present in Fig. 5 scatter graphics to illustrate. GIG is better than the best edge maps methods (SED, HED, and GIG-Positional) with statistical significance (p-values $< 10e-17$) and comparable to the GIG-Unique representations (p-values $\sim$0.02, not depicted). GIG is statistically better than GIG-Edge (p-value $< 10e-14$) despite both presenting similar segmentation metrics. We present the RCF comparison with the GIG filtered version, which was still inferior to the RCF (p-value $< 10e-9$) despite the improvement from GIG.

5 Discussion

Edge maps as gradients are commonly used as a preprocessing step in many applications because they are fast to compute and usually facilitate image analysis. Knowing the application and the type of analysis, one should always consider if the contour-oriented image simplification is enough for the task.

Our application is the hierarchical watershed (HWS), which deals with non-linear image analysis based on a hierarchy of partitions. In essence, given

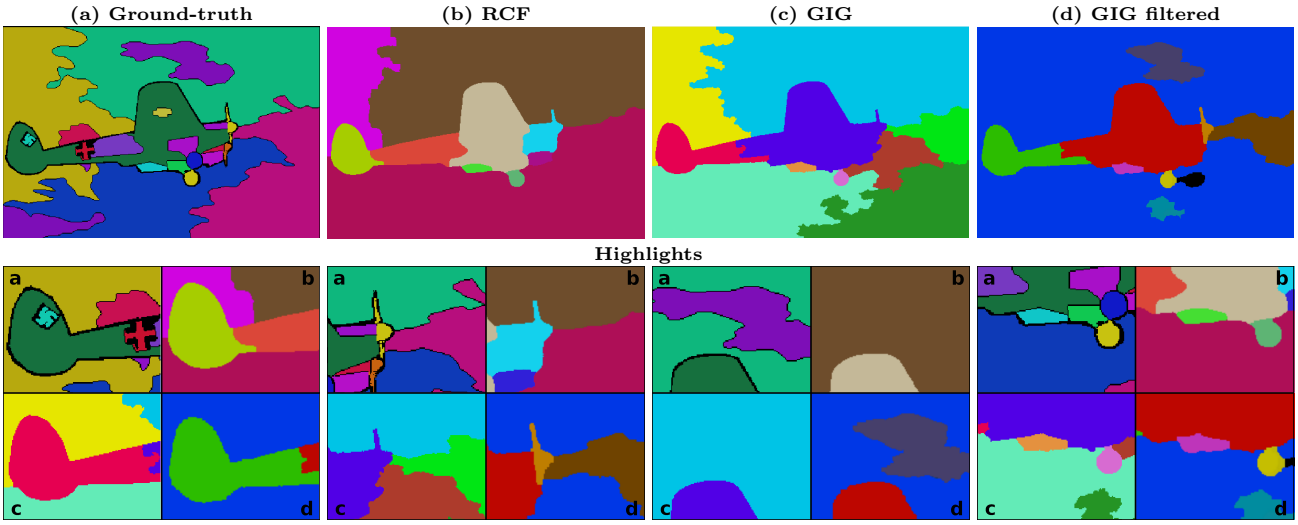

Figure 4: Samples of the segmentations with 10 regions produced by the RCF, GIG and GIG filtered by the morphological operation. The black areas in the ground-truth (a) indicate regions without consent among the annotators. Overall, the RCF (b) boundaries between regions are cleaner. In GIG (c) and its filtered version (d), we have more details from the object along with some background information. Also, the filtered version is more concentrated in certain details and areas of the background. In the second row, we present some region highlights to illustrate our remarks. Best viewed in color.

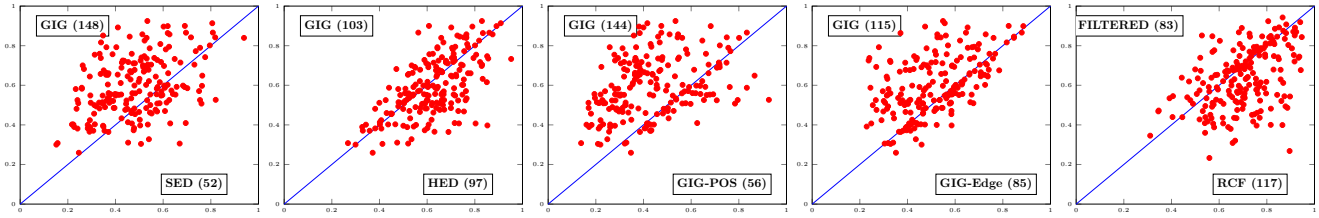

Figure 5: Pair-wise comparison of the F-measure results (red dots) on the best scale for each method. The boxes' values are the number of images that are better for a particular method. For a better visualization, GIG-Positional as GIG-POS and the GIG filtered version as FILTERED. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

an input image (usually a gradient), it creates a multi-scale image representation as an arrangement of coarse to fine partitions as an output. A ranked set of markers is necessary to construct the partitions, usually by identifying local minima based on a given geometric criterion (e.g., volume, area). We could define the hierarchy construction as the minimum spanning problem, aiming to find a solution with the minimum total cost by iteratively merging marked regions linked by the cheapest cost. The solution is optimal and consistent with the original image (details in [29]).

It is important to notice that the core of the HWS resides on the cost value and the markers representing the topology of a region, both extracted from the image magnitudes. From this perspective, one could argue that the success of this method relies on a good gradient image that reflects the distribution of the original image. The usual gradients with well-delineated contours provide clear extreme values for ranking, but we argue that one should consider that these values are constantly contrasted with neighboring regions. Therefore, the depiction of uniformity and small details in conjunction with strong contours could provide additional context.

The results showed that better edge maps do not necessarily translate to better segmentation. For instance, gradients with fuzzy contours like those produced by SED and HED are not the best candidates, despite having good metrics on the edges. The small GIG-RAG representations are primarily gradients of regions instead of edges, yet, their segmentation results are not as bad as one could imagine except for the precision metric.

HED and GIG-Edge have similar segmentation met-

Table 3: Segmentation results for all compared methods when applied as gradient input to the hierarchical watershed method. Results presented as F-scores for boundaries in terms of optimal dataset scale (ODS), optimal image scale (OIS), and average precision (AP) through all scales, and for the Probabilistic Rand Index (PRI). Perfect F-score and PRI=1.

	F-measure regions			PRI	
Gradient	ODS	OIS	AP	ODS	OIS
SED [15]	0.559	0.617	0.477	0.746	0.742
HED [21]	0.616	0.687	0.485	0.747	0.746
RCF [22]	**0.721**	**0.787**	0.548	**0.835**	0.877
GIG-RAG-1k (SLIC)	0.537	0.571	0.420	0.718	0.727
GIG-RAG-5k (SLIC)	0.540	0.580	0.427	0.728	0.729
GIG-RAG-10k (SLIC)	0.549	0.598	0.438	0.737	0.739
GIG-RAG-50k (SLIC)	0.582	0.638	0.482	0.758	0.788
GIG-RAG-1k (SpixelFCN)	0.550	0.607	0.470	0.749	0.780
GIG-RAG-5k (SpixelFCN)	0.558	0.624	0.535	0.758	0.786
GIG-RAG-10k (SpixelFCN)	0.559	0.625	0.518	0.756	0.786
GIG-RAG-50k (SpixelFCN)	-	-	-	-	-
GIG-24-Unique	0.624	0.652	0.456	0.777	0.795
GIG-80-Unique	0.625	0.656	0.449	0.781	0.791
GIG	0.620	0.689	0.508	0.788	0.820
GIG-Edge	0.613	0.674	0.487	0.768	0.798
GIG-Positional	0.599	0.619	0.465	0.742	0.751
GIG (filtered)	0.645	0.715	**0.556**	0.832	**0.885**

rics but distinct gradients: HED has fuzzy, thick contours with little details, while GIG-Edge has thin, detailed contours. GIG performed better than both in all metrics: GIG shares the thick contours but not the fuzziness with HED, and it shares the details with GIG-Edge, plus additional information about patterns. To identify which characteristics command the improvement, we could examine the GIG-Unique methods, which also performed better than HED and GIG-Edge. GIG and GIG-Unique share sharp contours, details, and information patterns, indicating that these are the factors that differ and improve from HED and GIG-Edge. In contrast, GIG performs better than GIG-Unique and is varied by the thicker contours. We could also see the importance of the details and pattern information on the different results obtained with GIG-RAG from SLIC and SpixelFCN. While we could arrive at more contour-oriented gradients with a larger number of super-pixels with SLIC, the pattern information preserved with SpixelFCN gave us superior segmentation metrics and values comparable with SED and HED, even with as little as 1k regions.

Overall, gradients with thick, sharp contours perform better. But as indicated in the qualitative analysis (Fig. 4) and the precision and PRI metrics for the filtered GIG compared to RCF, additional details and information about uniform regions positively contributed to the segmentation results regarding small objects and uniformity.

6 Conclusions

In this work, we presented an extended formalism for the edge-weighted image gradient operator, named GIG, initially proposed in [18]. We proposed three strategies to explore larger image areas and make changes driven by the RF mechanics to achieve a well-considered learning framework operating on graphs. Experiments demonstrated that reducing the number of data points by grouping the image pixels before the graph creation reduced the training time but compromised the performance on the edge detection and segmentation tasks. Also, expanding the analysis region by removing redundancy yielded similar results to the original proposal, indicating that the initial area of study already captured the necessary information, and the redundancy did not diminish the RF generalization in this particular application. Finally, the strategy that considered the position of the features regarding the RF mechanism resulted in better results in the edge detection task.

Furthermore, we extensively analyzed the gradients in the segmentation task, contrasting them with well-known methods for the edge detection task. The results indicated that gradients with fuzzy contours yielded the worst outcomes and that uniform regions and small objects details had a substantial impact but were not as crucial as the sharp thick contours. We performed statistical analysis for GIG on the segmentation task, which is significantly better than most of the best edge maps methods. Moreover, it validated our original selection of attributes, where GIG was superior to the

grouped pixels and the positional strategies and equivalent to the extended region. For future work, we will explore enriched graphs like the ones produced by hierarchical methods.

Acknowledgment

The authors thank the Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq – (PQ 310075/2019-0 and Grant 407242/2021-0), Fundação de Amparo a Pesquisa do Estado de Minas Gerais – FAPEMIG – (Grants PPM-00006-18), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES – (Grant STIC-AmSUD TRANSFORM 88881. 14325 8/2017-01, grant COFECUB 88887-.191730/2018-00 and funding Finance Code 001), PUC Minas and INRIA Associate Team program under the project *Learning on graph-based hierarchical methods for image and multimedia data*.

References

[1] D. Domínguez and R. R. Morales, *Image segmentation: advances*, vol. 1. Magnum Publishing LLC, 2016.

[2] R. Kurzweil, *How to create a mind: The secret of human thought revealed*. Penguin Books, 2013.

[3] S. Beucher, "Watershed, hierarchical segmentation and waterfall algorithm," *Mathematical Morphology and its Applications to Image Processing*, pp. 69–76, 1994.

[4] L. Najman and M. Schmitt, "Geodesic saliency of watershed contours and hierarchical segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 1163–1173, 1996.

[5] P. M. Krishnammal, L. M. Therase, E. A. Devi, and R. M. Joany, "Wavelets and convolutional neural networks-based automatic segmentation and prediction of MRI brain images," *IOT with Smart Systems*, pp. 229–241, 2022.

[6] S. Makrogiannis, N. Annasamudram, Y. Wang, H. Miranda, and K. Zheng, "A system for spatio-temporal cell detection and segmentation in time-lapse microscopy," in *IEEE International Conference on Bioinformatics and Biomedicine*, pp. 2266–2273, 12 2021.

[7] S. Beucher, "Use of watersheds in contour detection," in *International Workshop on Image Processing*, 1979.

[8] B. Perret, J. Cousty, S. J. F. Guimarães, and D. S. Maia, "Evaluation of hierarchical watersheds," *IEEE Trans. on Image Processing*, pp. 1676–1688, 4 2018.

[9] F. Meyer, "Stochastic watershed hierarchies," in *IEEE International Conference on Advances in Pattern Recognition*, pp. 1–8, 1 2015.

[10] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 898–916, 5 2011.

[11] P. Bosilj, E. Kijak, and S. Lefevre, "Partition and inclusion hierarchies of images: A comprehensive survey," *Journal of Imaging*, vol. 4, p. 33, 2 2018.

[12] L. Najman and H. Talbot, *Mathematical morphology: from theory to applications*. John Wiley & Sons, 2013.

[13] K. Haris, S. Efstratiadis, N. Maglaveras, and A. Katsaggelos, "Hybrid image segmentation using watersheds and fast region merging," *IEEE Trans. on Image Processing*, vol. 7, pp. 1684–1699, 1998.

[14] P. Shanthakumar and P. G. Kumar, "Computer aided brain tumor detection system using watershed segmentation techniques," *International Journal of Imaging Systems and Technology*, vol. 25, pp. 297–301, 12 2015.

[15] P. Dóllar and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pp. 1558–1570, 8 2015.

[16] B. Perret, J. Cousty, S. J. F. Guimarães, Y. Kenmochi, and L. Najman, "Removing non-significant regions in hierarchical clustering and segmentation," *Pattern Recognition Letters*, vol. 128, pp. 433–439, 12 2019.

[17] K. Otiniano-Rodríguez, A. de A. Araújo, G. Cámara-Chávez, J. Cousty, S. J. F. Guimarães, and B. Perret, "Hierarchy based salient regions: A region detector based on hierarchies of partitions," in *23rd Iberoamerican Congress on Pattern Recognition*, pp. 444–452, 2018.

[18] R. Almeida, Z. K. G. Patrocínio, A. de A. Araújo, E. Kijak, S. Malinowski, and S. J. F. Guimarães, "Descriptive image gradient from edge-weighted image graph and random forests," in *IEEE Conference on Graphics, Patterns and Images*, pp. 338–345, 10 2021.

[19] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[20] A. J. Wyner, M. Olson, J. Bleich, and D. Mease, "Explaining the success of AdaBoost and random forests as interpolating classifiers," *The Journal of Machine Learning Research*, vol. 18, pp. 1558–1590, 2017.

[21] S. Xie and Z. Tu, "Holistically-nested edge detection," in *IEEE International Conference on Computer Vision*, pp. 1395–1403, 12 2015.

[22] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5872–5881, 7 2017.

[23] P. Dóllar, S. Belongie, and P. Perona, "The fastest pedestrian detector in the west," in *British Machine Vision Conference*, pp. 68.1–68.11, 2010.

[24] G. Biau, L. Devroye, and G. Lugosi, "Consistency of random forests and other averaging classifiers," *Journal of Machine Learning Research*, vol. 9, pp. 2015–2033, 2008.

[25] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *IEEE International Conference on Computer Vision*, pp. 416–423, 2001.

[26] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels," Tech. Rep. 149300, 6 2010.

[27] F. Yang, Q. Sun, H. Jin, and Z. Zhou, "Superpixel segmentation with fully convolutional networks," in *IEEE CVPR*, pp. 13964–13973, 6 2020.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[29] J. Cousty and L. Najman, "Incremental algorithm for hierarchical minimum spanning forests and saliency of watershed cuts," in *Math. Morphology and Its Applications to Image and Signal Processing*, pp. 272–283, Springer, 2011.

[30] B. Perret, G. Chierchia, J. Cousty, S. F. Guimarães, Y. Kenmochi, and L. Najman, "Higra: Hierarchical graph analysis," *SoftwareX*, vol. 10, p. 100335, 2019.

[31] J. Pont-Tuset and F. Marques, "Measures and meta-measures for the supervised evaluation of image segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, p. 2131–2138, 6 2013.