Faster α-expansion via dynamic programming and image partitioning

Jefferson Fontinele[1], Marcelo Mendonça[2], Marco Ruiz[3], Joao Papa[4] and Luciano Oliveira[5]
[1,2,5] *Intelligent Vision Research Lab, Federal University of Bahia, Bahia, Brazil*
[3] *VORTEX-CoLab*
[4] *São Paulo State University, Bauru, Brazil*
[1,2,5] {jeffersonfs, marceloms, lrebouca}@ufba.br
[3] marcoantonio.ruizrueda@altran.com
[4] joao.papa@unesp.br

***Abstract*—Image segmentation is the task of assigning a label to each image pixel. When the number of labels is greater than two (multi-label) the segmentation can be modelled as a multi-cut problem in graphs. In the general case, finding the minimum cut in a graph is an NP-hard problem, in which improving the results concerning time and quality is a major challenge. This paper addresses the multi-label problem applied in interactive image segmentation. The proposed approach makes use of dynamic programming to initialize an α-expansion, thus reducing its runtime, while keeping the Dice-score measure in an interactive segmentation task. Over BSDS data set, the proposed algorithm was approximately 51.2% faster than its standard counterpart, 36.2% faster than Fast Primal-Dual (FastPD) and 10.5 times faster than quadratic pseudo-boolean optimization (QBPO) optimizers, while preserving the same segmentation quality.**

***Index Terms*—α-expansion, dynamic programming, multi-label, image segmentation.**

I. Introduction

In a nutshell, the goal of multi-labeling is to determine a label for each pixel in the image, which is encoded in a graph node. Multi-label assignment in graphs is commonly applied in computer vision tasks such as image restoration [1–3] and segmentation [3–8]. Assigning multi-labels to images is usually an NP-hard problem [9], where the runtime may increase regarding the number of image pixels.

One of the main algorithms for multi-labeling problems is the so-called α-expansion [9], which presents linear complexity with respect to the number of pixels, and has shown to be very efficient in practical applications [2]. α-expansion algorithm works iteratively, performing an expansion move for each label, *i.e.*, expansion goes toward a local optimum by employing a maximum-flow strategy that takes $\mathcal{O}(n^2)$ operations, where n is the number of pixels.

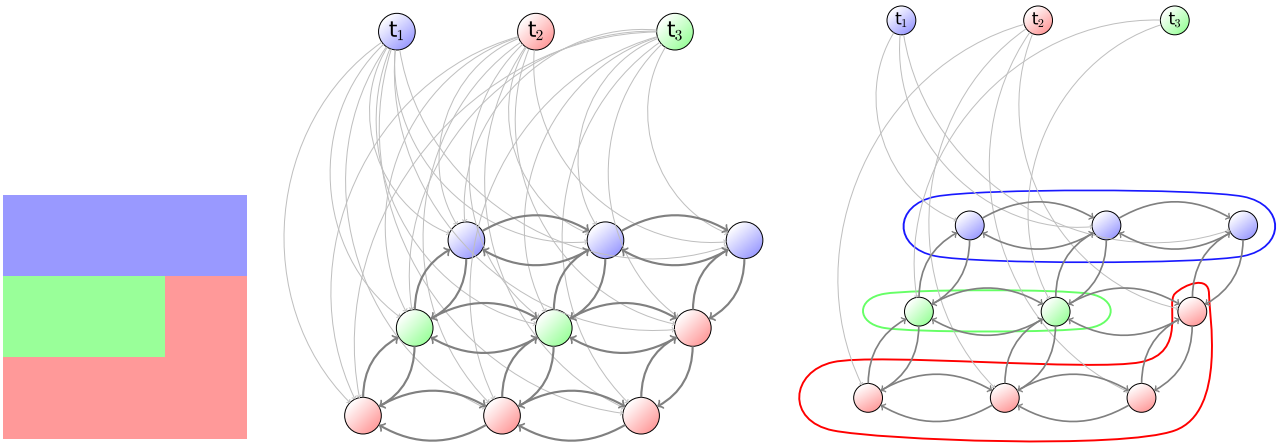
For α-expansion-based algorithms, the closeness to the optimal solution is given by the distance $2kE(f^*)$, where f^* is the global minimum, k is a constant that depends on pixel values, and $E(f^*)$ is the minimum energy of the graph [9]. From this rule, α-expansion strongly depends on the initial conditions, which allows for improving the algorithm by changing its initialization. For instance, Lempitsky *et al.* [10] consider different ways to initialize the α-expansion. The goal is to generate suboptimal solutions, which are later combined through fusion moves in order to obtain a labeling procedure with lower energy. Felzenszwalb and Veksler [11] use an expansion algorithm to optimize a specific energy function. Depending on the strategy adopted for initialization (if starting from a constant or a random labeling), the expansion algorithm eventually falls into a local minima.

Other extensions of α-expansion algorithm have been proposed [12–14]. Veksler [12] applies dynamic programming as part of an α-expansion strategy, thus replacing the very used maximum flow algorithm. Dynamic programming is used to solve multi-label problems through geometric constraints as an optimization algorithm. Khatab *et al.* [13] modify the GrabCut [15] algorithm in order to initialize segmentation without human intervention for multi-label problems. Similarly to α-expansion, the method addresses multi-label segmentation as a set of binary problems by iteratively minimizing an energy function. Isack *et al.* [14] tackle a multi-label segmentation problem by using hedgehog shapes to constraint an interactive segmentation. Final optimization is done by a modified α-expansion algorithm based on the shape prior.

A. Contributions

Here we propose an approximate algorithm by exploiting dynamic programming and partitioned images. The goal is to define the granularity of label assignment in each iteration of the algorithm. Differently from [11] and [12], our work proposes to initialize α-expansion via dynamic programming, rather than replacing its standard optimization algorithm. To make the runtime execution of dynamic programming viable, we adopt an image partitioning strategy. By doing so, we introduce an extension of the α-expansion algorithm that improves the runtime of iterative multi-label segmentation, while a close approximate solution is provided. To achieve this goal, we exploit a new way to initialize an α-expansion via dynamic programming and local clustering of pixels to reduce the runtime of α-expansion moves. Our method, called iterative dynamic programming expansion (IDP-Expansion), was evaluated for the task of interactive image segmentation, which consists of labeling each pixel of an image, starting from hand-crafted annotations (seeds) of a dataset. IDP-Expansion

978-1-7281-6926-2/20/$31.00 ©2020 IEEE



(a) Array of pixels. (b) (c)

Figure 1. Representation of a pixel array (a) in a graph, where the energy function is minimized to obtain each label (b). Each terminal t_n represents a label after a cut in the graph, where the image is segmented (c).

was 51.2% faster than the standard α-expansion, 36.2% faster than Fast Primal-Dual (FastPD) [16] and 10.5 times faster than quadratic pseudo-boolean optimization (QBPO) [17], over BSDS data set [18].

II. Background

Greig *et al* [1] demonstrated that a cut on a graph could be used for binary image restoration, thus introducing the graph-cut for image processing. This problem was modelled by using the maximum estimate *a posteriori* of a Markov Random Field (MRF) to minimize an energy function, defined as

$$E\left(I\right) = -\lambda \sum_{p \in P} \ln Pr\left(I_p | I^o\right) + \sum_{p,q \in \mathcal{N}} f(p,q)\,, \qquad (1)$$

where $\mathcal{P}$ is set of pixels, $\mathcal{N}$ is the set of neighboring pixels, and λ is a real-valued constant. The function $Pr(I_p|I^o)$ represents the conditional probability given a pixel intensity value, I_p belongs to one of the classes in the binary image segmentation problem, I^o denotes label intensity, and the function, $f(.,.)$, is given by

$$f(p,q) = \begin{cases} 1 & \text{if } I_p \neq I_q \\ 0 & \text{if } I_p = I_q \end{cases}\,, \qquad (2)$$

and represents a relation of the neighborhood between the pixels p and q in the image grid.

The pixels clustering problem can be modelled as a minimal cut problem, whereas a minimum cut problem with two labels can be modelled, in turn, as a maximum flow [19]. The solution to this problem can be obtained in polynomial time by maximum flow algorithms, *e.g.*, the algorithms proposed in [19], [4], and the push-relabel algorithm [20].

A. Image multi-labeling

Image multi-labeling consists in defining labels for each pixel in such a way that minimizes an energy function. There are a set of $\mathcal{P}$ image pixels and a set of $\mathcal{L}$ labels, where each pixel $p \in \mathcal{P}$ is assigned to a label $l \in \mathcal{L}$. If $|\mathcal{L}| = k$, for $k = 2$, we have a binary problem. An energy function can be defined as

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{(r,q) \in \mathcal{N}} V_{rq}(f_r, f_q)\,, \qquad (3)$$

where f_p denotes the relation between a label and each image pixel. $D_p(f_p)$ is the data function that depends on the difference between I_p (pixel value p) and I_l (label pixel value). $V_{rq}(f_r, f_q)$ is the attenuation function that depends on the relationship between neighbors in $\mathcal{N}$, and must satisfy the constraints:

$$V_{rq}(0,0) + V_{rq}(1,1) \leq V_{rq}(0,1) + V_{rq}(1,0)\,. \qquad (4)$$

From $D_p(f_p)$ and $V_{rq}(f_r, f_q)$, Potts Models [2] are used following [1]. Boykov *et al.* [9] show that it is possible to transform a multi-label problem into a multiple-cut problem (NP-hard) [21], so that approximate algorithms are required to solve this task.

B. Graph representation

Let be $G = (\mathcal{V}, \mathcal{E})$ a graph, where $\mathcal{V}$ defines the set of vertices and $\mathcal{E}$ the set of edges. The vertices are divided into two groups: Type-n that represents each pixel $p \in P$, and type-t that represents each label $l \in \mathcal{L}$. Each edge is comprised of two sets – *t-links* and *n-links*. *t-links* are the set of edges that defines the connection of each type-n vertex for each type-t vertex. *n-links* represent the relationship between the neighbors of vertices type-n. The neighborhood $\mathcal{N}$, in case of 2D images, may have 4 or 8 neighbors. Regarding the energy function E, *n-links* depict the attenuation function, while *t-links* represent the data function. Figure 1(a) shows an example of a 3×3 array of RGB pixels, considering three segmentation classes, and Figure 1(b) depicts its graph representation. In Figure 1(b), the *n-links* are the strong lines between unmarked vertices (type-n), and the *t-links* are edges with edges leaving

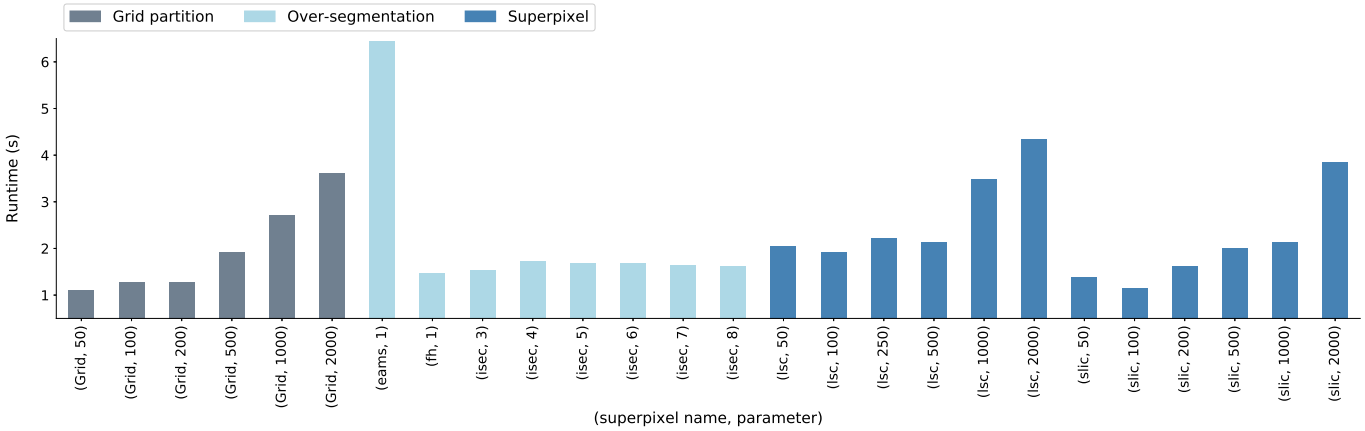


Figure 2. Ablation of the partitioning method to be used in IDP-Expansion.

Algorithm 1 α-expansion

Set $\mathbf{m}$ arbitrary labels for each $l \in \mathcal{L}$
repeat
  $mp = E(\mathbf{m})$ ▷ Calculate the energy for the current state
  for all $\alpha \in \mathcal{L}$ **do**
    Calculate a minimum α-expansion $\hat{\mathbf{m}}$ via Graph-cut
    $\alpha m = E(\hat{\mathbf{m}})$
    if $\alpha m < mp$ **then**
      mp = αm
    end if
  end for
until $\alpha m > mp$

in t_1, t_2 and t_3 (type-t) and arriving in the type-n vertices. Figure 1(c) depicts how the nodes are labeled as segments after optimization.

III. Interactive segmentation with IDP-Expansion

A. α-expansion

The original α-expansion algorithm solves sub-problems with $k = 2$, by successively using a maximum-flow optimizer. The algorithm converges, when the energy of the current iteration is higher than the one in the previous iteration. For each iteration, the algorithm divides the set of pixels into α and non-α, labeling several vertices in each iteration (Alg. 1). A remarkable characteristic of this algorithm is its sensitivity to initialization [2]. Different ways of assigning arbitrary labels before the iterations are prone to change the final result, *i.e.*, the obtained energy and consequently the runtime. We exploit such a condition as a pathway to reduce the overall runtime of an α-expansion-based segmentation. For that, we apply dynamic programming as a way to initialize the α-expansion algorithm.

Algorithm 2 IDP-Expansion

Get superpixels $\mathcal{S}$ of the image I
DP-TopDown($|\mathcal{S}|$, $\mathcal{S}$, 0, 0) ▷ See Alg. 3
Calculate an α-expansion move

Algorithm 3 DP-TopDown

function DP-TopDown($ns, \mathcal{S}, v, level$)
  if Solution $r \exists t$ for the sub-problem $v, level$ **then**
    return r ▷ r is a sub-solution in the table t
  else
    for all $l \in \mathcal{L}$ **do**
      u Label $S[ns]$ as the label l
      Calculate $e = D(u)$ + DP-TopDown($|\mathcal{S}| - 1$, $\mathcal{S}$, level + 1)
      if e is the minimum energy **then**
        $min = e$
        $minl = l$
      end if
    end for
    Label $S[ns]$ with the label $minl$
    Save in t: ns, $level$, and e
  end if
end function

B. IDP-expansion

Our IDP-expansion method modifies the original α-expansion in order to make viable the initialization via dynamic programming (see Alg. 2). From the experiments, we observed that application of dynamic programming to label the pixels directly is impractical in terms of runtime. We solved this issue by labeling image partitions instead of pixels, *i.e.*, the vertices of the graph are assigned to groups of pixels instead of each pixel individually.

*a) **Image partitioning**:* In principle, there are no constraints about which strategy could be adopted to provide image partitioning. This is because the final result of the segmentation via IDP-expansion is not affected by the initial

(a)

(b)

(c)

Figure 3. Examples of image partitioning: (a) grid, (b) over-segmentation (ISEC [22]) and (c) superpixel (LSC [23]).

conditions, but only the time spent by the algorithm to converge. Bare that in mind, we evaluated different strategies of image partitioning in order to identify the one that provides the best gain regarding computational efficiency. Tests included three main strategies for image partitioning: **Over-segmentation** methods divide the image into segments that are quite variable in both size and shape, depending on the image content. Usually, the generated segments present boundaries with high correspondence to the contours of the objects in the image, and the number of segments is highly variable. The following methods were evaluated: EAMS [24], ISEC [22] and FH [25]. On the other hand, **superpixel** methods provide strict control over the number of generated segments. The results generally pursue a compromise between correspondence to the object contours and uniformity in shape and size of the segments. The evaluated methods in this partitioning class were: SLIC [26] and LSC [23]. Lastly, an image content-agnostic, **grid partition** strategy was included. The goal was to verify if the additional computational effort required by image content-aware partitioning methods, like over-segmentation and superpixel, justify their usage, or if just using the simplest possible partitioning method would be more beneficial.

Figure 2 shows an ablation study over the overall runtime spent by IDP-expansion, when using the different methods for image partitioning (including tests for different hyperparameters, when available). As can be seen, although at first glance it could be expected that using content-aware methods would be favorable, the results reveal that the grid-like partitioning with 50 partitions (grid 50, in Fig. 2) is the best option. Hence we followed with this solution in IDP-expansion. Figure 3 illustrates examples of each one of the aforementioned image partitioning strategies.

*b) **Dynamic programming for vertice labeling optimization**:* IDP-expansion uses only the data function to get an initial definition of the labels and to initialize the α-expansion. Once the image partitioning is generated, the minimum energy is computed by means of a dynamic programming algorithm that we call DP-TopDown (see Alg. 3). This algorithm works by assigning a label to each image partition and checking the new energy function value for that solution. If the value of the energy function is less than the lowest one found so far, the value of the energy function is updated, and the assignment move performed is stored in a table. The algorithm ends, when it identifies that a new move will not lead to an energy reduction. The purpose of DP-TopDown is to construct a search tree with possible solutions to the multi-label problem. This way, we avoid recalculations of the energy at each move.

*c) **Complexity**:* The complexity of the function in line 2 of Alg. 2 is $\mathcal{O}(nl)$, where n is the number of pixels and l is the number of labels. DP-TopDown is performed by evaluating all possibilities of label assignments for each image partition. Next a move of α-expansion is performed. IDP-Expansion maintains linear complexity, depending on the number of pixels. Since IDP-Expansion changes the initialization of an α-expansion move, the obtained solution, f, and at most $2f*$ are optimal solutions.

*d) **Energy function**:* To define the t-link weights, a function, D, proposed by Nieuwenhuis2013 *et al.* [2] is used. The function considers both the location of the pixels and the color value of each pixel, achieving a tradeoff between the distance for scribbles and the similarity in color. D is given by

$$D_p(f_p) = -logP(I(x), x|u(x) = p)\,, \tag{5}$$

where the joint probability distribution P is defined as

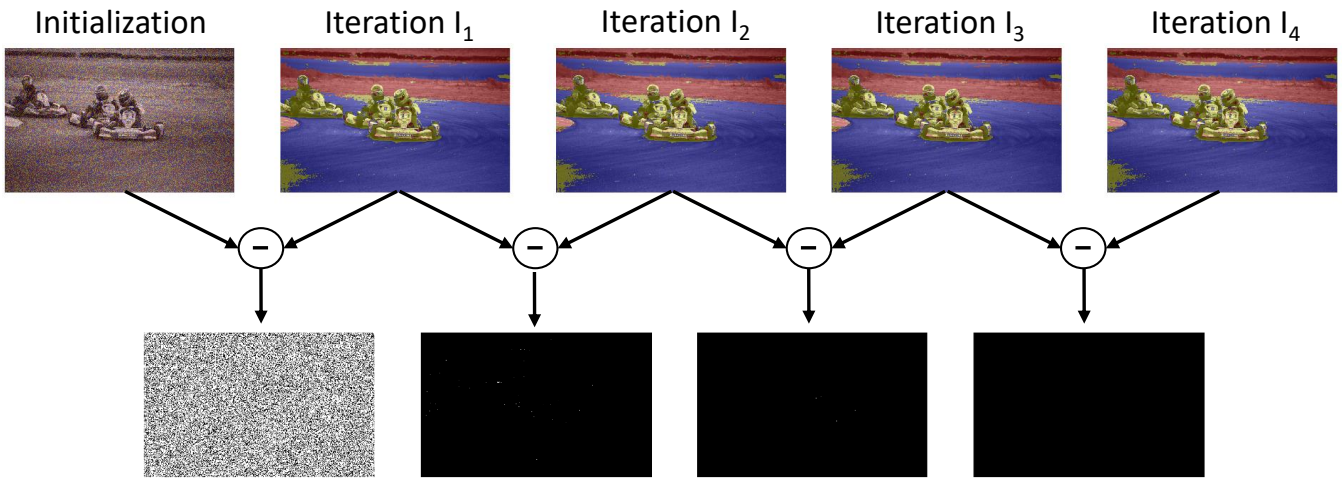


Figure 4. IDP-Expansion on each iteration.

Figure 5. IDP-Expansion work on each iteration: the amount of energy ($E \times 10^6$) per iteration.

$$P(I(x), x|u(x) = p) = \frac{1}{m_p} \sum_{j=1}^{m_p} k_\sigma(I - I_{pj}) k_{i_p}(x - x_{pj}), \tag{6}$$

where $k_\sigma(I - I_{pj})$ is a function, which represents the probability of a pixel to be similar to scribble-selected colors, defining how much the colors contribute to the overall likelihood. The term $k_{i_p}(x - x_{pj})$ represents the probability that a pixel belongs to a class in relation to the location, weighing the contribution of the location in function P.

$V_{(r,q)}$ can now be defined as a Gaussian kernel used as an attenuation function:

$$V_{(r,q)} = \begin{cases} \exp\left(\dfrac{-\|\mathcal{I}_r - \mathcal{I}_q\|^2}{2\sigma^2}\right) & \text{if } I_r \neq I_q \\ 0 & \text{if } I_r = I_q \end{cases}. \tag{7}$$

Figure 6. Example of available images in the data set [18].

IV. Experiments and Results

To evaluate IDP-Expansion performance, we used the BSDS data set [18] in an interactive segmentation task. Over that data set, the runtime and the segmentation quality provided by the Dice-score metric of IDP-Expansion were compared with the methods: A randomly initialized α-expansion, the Fast-PD [16] and the QPBO [17].

Figure 7. Top-down view: considering seeds provided on the original images (first line) and ground truths (second line), some intermediary results of the function DP-TopDown (third line; also see Alg. 3) and IDP-Expansion (last line).

Table I
LABEL DISTRIBUTION IN THE DATA SET FOUND IN [18]

Labels	2	3	4	5	6	7	8	9	10
Frequency	66	104	58	18	11	2	2	1	1

The benchmark also includes our method and α-expansion, both set to stop the segmentation with one iteration. In fact, this choice is based on the results found in Figs. 4 and 5. Figure 4 illustrates a resulting image in each iteration over an image sample from BSDS data set. Figure 5 shows the relationship between IDP-Expansion energy function and number of iterations to converge. The quality evaluation of IDP-Expansion on each iteration, as well as the plot reveal that just after one iteration most of the energy is minimized, making the remaining iterations negligible.

A. Data set

The BSDS data set used for the experiments was introduced in [18]. The provided images and annotations were used in order to evaluate an interactive segmentation. Figure 6 illustrates one example from the data set, where the first line shows an original image, and the second one depicts the hand-crafted ground truth annotation: seeds (yellow lines) and semantic segmentation masks (one color for each defined label). The data set contains 158 images with 262 annotations. Table I summarizes the distribution of the labels in the data set.

B. Evaluation metric

Dice-score was used as a metric to evaluate the segmentation quality of the images in the BSDS data set. Dice-score, $Dice(.,.)$, determines the overlap of the area between the automatic segmentation, Ω, performed by the evaluated algorithm and its ground truth annotation, $\bar{\Omega}_i$.

$$Dice(\Omega, \bar{\Omega}) = \frac{1}{n} \sum_{n}^{i=1} \frac{2|\Omega \cap \bar{\Omega}|}{|\Omega_i| + \bar{\Omega}_i|}, \tag{8}$$

where n as the number of segments.

Additionally, we also calculated the average runtime of each algorithm over the BSDS data set.

C. Result analysis

Table II summarizes the results of the benchmarked methods, considering the average Dice-score and runtime for all images on BSDS data set. According to the results, IDP-Expansion achieved the same segmentation quality as the original α-expansion and FASTPD [16], and is only 0.02 lower than QBPO [17]. Considering the computational time,

Table II
BENCHMARK CONSIDERING PIXEL-WISE SEGMENTATION AND RUNTIME.

Algorithms	**Dice-score**	**Runtime (s)**
α-expansion	0.88	1.62
α-expansion (one iteration)	0.88	0.82
QBPO	0.90	9.11
FastPD	0.88	1.24
IDP-Expansion	0.88	**0.79**
IDP-Expansion (one iteration)	0.88	**0.74**

our method was 51.2% faster compared to the original α-expansion, and 3.7% faster than one iteration of the original α-expansion. Considering IDP-Expansion with just one iteration, our method achieved still faster results (6% faster than the full version of IDP-Expansion), while maintaining the very same segmentation quality. This is can be explained due to the fact that the initialization provided by IDP-expansion reduces the amount of pixel labeling in comparison with a random initialization of the original α-expansion.

Figure 7 depicts some examples of segmentation results of IDP-Expansion. From a top-down view, the last two lines shows the intermediary segmentation of the DP-TopDown function and the final result of the IDP-Expansion segmentation, respectively. It is noteworthy that the DP-TopDown function makes the coarser work in the segmentation, while the rest of the IDP-Expansion method refines the previous results until reaching the final image segmentation.

V. CONCLUSIONS

IDP-Expansion achieves comparable results with the original α-expansion algorithm, while speeding up the time of convergence in 50.6%. In fact, the definition of a function that adequately initialize the α-expansion substantially reduced the original algorithm. Particularly considering our proposed method, the use of a dynamic programming strategy decreased the search to minimize the term D in the energy function E. The remarkable runtime reduction was mainly obtained by exploiting the sub-problems redundancy in the search space, as well as by doing image partitioning.

Differently from α-expansion, IDP-Expansion can be only applied in 2D images and is not expansible to problems described by other types of graphs. This way, exploring the adequate initialization for α-expansion in other contexts, such as 3D image segmentation, should be the straightforward way to future works. The development of other initialization strategies that exploit GPU to reduce computational time is another alternative for investigation.

ACKNOWLEDGEMENTS

This work was supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), which supported Luciano Oliveira under grant 307550/2018-4 and João Paulo Papa under grant 307066/2017-7. Besides, this work was also support by São Paulo Research Foundation (Fapesp) under the grants 2013/07375-0, 2014/12236-1, and 2017/25908-6 (Microsoft/Fapesp).

REFERENCES

[1] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact maximum a posteriori estimation for binary images," *Journal of the Royal Statistical Society. Series*, vol. 51, no. 2, pp. 271–279, 1989.

[2] C. Nieuwenhuis and D. Cremers, "Spatially varying color distributions for interactive multilabel segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 5, pp. 1234–1247, 2013.

[3] L. Gorelick, Y. Boykov, and O. Veksler, "Adaptive and move making auxiliary cuts for binary pairwise energies," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4922–4930.

[4] Y. Boykov and V. Kolmogorov, "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision," *IEEE Transactions on PAMI*, vol. 261, no. 92, pp. 1124–1137, 2004.

[5] D. Khattab, H. M. Ebied, A. S. Hussein, and M. F. Tolba, "Multi-label automatic grabcut for image segmentation," in *2014 14th International Conference on Hybrid Intelligent Systems*. IEEE, 2014, pp. 152–157.

[6] J. S. Baxter, M. Rajchl, A. J. McLeod, J. Yuan, and T. M. Peters, "Directed acyclic graph continuous max-flow image segmentation for unconstrained label orderings," *International Journal of Computer Vision*, vol. 123, no. 3, pp. 415–434, 2017.

[7] H. Isack, L. Gorelick, K. Ng, O. Veksler, and Y. Boykov, "K-convexity shape priors for segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 36–51.

[8] J. Liu, P. L. Rosin, X. Sun, J. Xiao, and Z. Lian, "Image-driven unsupervised 3d model co-segmentation," *The Visual Computer*, vol. 35, no. 6-8, pp. 909–920, 2019.

[9] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[10] V. Lempitsky and S. Roth, "Fusion Moves for Markov Random Field Optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 8, pp. 1–14, 2010.

[11] P. F. Felzenszwalb and O. Veksler, "Tiered scene labeling with dynamic programming," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3097–3104.

[12] O. Veksler, "Dynamic Programming for Approximate Expansion Algorithm," in *Lecture Notes in Computer Science*, 1st ed. Springer Berlin Heidelberg, 2012, ch. Computer V, pp. 850–863.

[13] D. Khattab, H. M. Ebied, A. S. Hussein, and M. F. Tolba, "Multi-label automatic GrabCut for image segmentation," in *2014 14th International Conference on Hybrid Intelligent Systems*. IEEE, dec 2014, pp. 152–157.

[14] H. Isack, "Hedgehog Shape Priors for Multi-object Segmentation," *Cvpr*, pp. 2434–2442, 2016.

[15] C. Rother, V. Kolmogorov, and A. Blake, "" grabcut" interactive foreground extraction using iterated graph cuts," *ACM transactions on graphics (TOG)*, vol. 23, no. 3, pp. 309–314, 2004.

[16] N. Komodakis, G. Tziritas, and N. Paragios, "Performance vs computational efficiency for optimizing single and dynamic mrfs: Setting the state of the art with primal-dual strategies," *Computer Vision and Image Understanding*, vol. 112, no. 1, pp. 14–29, 2008.

[17] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer, "Optimizing binary mrfs via extended roof duality," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8.

[18] J. Santner, T. Pock, and H. Bischof, "Interactive multi-label segmentation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6492 LNCS, no. PART 1, 2011, pp. 397–410.

[19] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Journal canadien de mathématiques*, vol. 8, pp. 399–404, 1956.

[20] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum-flow problem," *Journal of the ACM*, vol. 35, no. 4, pp. 921–940, 1988.

[21] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis, "The complexity of multiterminal cuts," *SIAM Journal on Computing*, vol. 23, no. 4, pp. 864–894, 1994.

[22] M. Mendonça and L. Oliveira, "Isec: Iterative over-segmentation via edge clustering," *Image and Vision Computing*, vol. 80, pp. 45 – 57, 2018.

[23] Z. Li and J. Chen, "Superpixel segmentation using linear spectral clustering," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1356–1363.

[24] P. Meer and B. Georgescu, "Edge detection with embedded confidence," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 23, pp. 1351–1365, 2001.

[25] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision (IJCV)*, vol. 59, no. 2, pp. 167–181, 2004.

[26] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 34, no. 11, pp. 2274–2282, 2012.