ON MINIMUM SPANNING TREE STREAMING FOR IMAGE ANALYSIS

Leonardo Gigli, Santiago Velasco-Forero, Beatriz Marcotegui

ON MINIMUM SPANNING TREE STREAMING FOR IMAGE ANALYSIS

Leonardo Gigli, Santiago Velasco-Forero and Beatriz Marcotegui

MINES ParisTech, PSL Research University, CMM - Center of Mathematical Morphology
{leonardo.gigli,santiago.velasco,beatriz.marcotegui@mines-paristech.fr}

ABSTRACT

This work addresses minimum spanning tree (MST) construction in streaming for images. We study the problem of computation a MST on streaming in which image columns from a continuous stream are processed in blocks of a given size. The correctness of proposed algorithm is proved and confirmed in the case of morphological segmentation of remote sensing images.

Index Terms— Minimum Spanning Tree, Streaming Processing, Hierarchical Segmentation

1. INTRODUCTION

The computation of minimum spanning tree (MST) on the gradient of an image is one of the main ingredients for the characterization of structures in different image processing algorithms including registration [1], anisotropic filtering [2], salience detection [3], hierarchical segmentation [4], marker-based segmentation [5], prior-based segmentation [6], spatiotemporal background subtraction [7], stereo matching [8] among others. More particullary, in Remote Sensing (RS) applications the computation of MST is one step to the retrieval of important objects from different scales [9]. However, recent advances in RS and computer techniques give birth to the explosive growth of remote sensing data [10]. The application of classical image processing technique needs to wait until all the data are known and this is an issue for streaming applications. Accordingly, streaming spatial algorithms has been identified as one of the main research challenges in RS[11].

Namely, in this paper we address the problem of computation a MST on streaming in which image columns from a continuous stream are processed in blocks of a given size. In a nutshell, the highlights of this paper are shown as follows:

- Methods to produce MST on streaming are introduced and their correctness proved.
- We illustrated, how can algorithms based on MST computation be adjusted to exploit the streaming nature that is becoming increasingly common in remotely sensed data.

The rest of the paper is organized as follows: In Section 2, we introduce the notation and the proposed algorithms. In Section 3, we validate our procedure using various experiments on image. Conclusion is presented in Section 4

2. ALGORITHMS

First, we introduce the notation that we use in this paper. Let $\mathbf{I}_t$ be an image streaming during time, in which new pixels come from one side of the image. Let $t = 0, \ldots, N$ be different intervals of time, and let I_t be the new pixels that arrive at time t, that is $\mathbf{I}_t = \mathbf{I}_{t-1} \cup I_t$. To simplify our problem, we assume that the new image I_t shares a column with $\mathbf{I}_{t-1}$: the last column of $\mathbf{I}_{t-1}$ is the first column of I_t (see Figure 1).

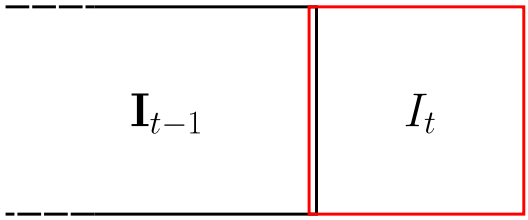

Fig. 1. Representation of image $\mathbf{I}_t$ streaming as the union of the two images $\mathbf{I}_{t-1}$ and I_t. We do not consider the two images as disjoint, but instead, they share one column of pixels.

An image can be considered as a standard 4-connected, undirected graph, with nodes being all the image pixels and edges between neighboring pixels being weighted by color/intensity differences. Using Kruskal's algorithm, a minimum spanning tree (MST) can be constructed in average $O(|\mathbf{E}| \log(|\mathbf{V}|))$ time, where $|\mathbf{E}|$ is the number of edges in the graph and $|\mathbf{V}|$ is the number of vertices. In our case, since we construct 4-connected graphs from images, we have that $|\mathbf{E}| \approx 2|\mathbf{V}|$, so Kruskal's algorithms takes $O(|\mathbf{V}| \log(|\mathbf{V}|))$. Nevertheless, in case of graphs associated to images, Bao et Al. [2] developed an algorithm that builds a MST for a 8-bit depth image in $O(|\mathbf{V}|)$ time.
Before introducing our work, we need to recall some operations between graphs that we use in our algorithms.

Definition 1 (Graph Union)**.** *Given two weighted graphs*

$\mathcal{G}_1 = (\mathbf{V}_1, \mathbf{E}_1, \mathbf{W}_1)$ *and* $\mathcal{G}_2 = (\mathbf{V}_2, \mathbf{E}_2, \mathbf{W}_2)$ *such that*

$$\mathbf{W}_1\big|_{\mathbf{E}_1 \cap \mathbf{E}_2} \equiv \mathbf{W}_2\big|_{\mathbf{E}_1 \cap \mathbf{E}_2},$$

we call $\mathcal{G}_1 \cup \mathcal{G}_2$ *the graph* $\mathcal{G} = (\mathbf{V}, \mathbf{E}, \mathbf{W})$*, with* $\mathbf{V} = \mathbf{V}_1 \cup \mathbf{V}_2$*,* $\mathbf{E} = \mathbf{E}_1 \cup \mathbf{E}_2$ *and*

$$\mathbf{W}(e) = \begin{cases} \mathbf{W}_1(e) & \text{if } e \in \mathbf{E}_1, \\ \mathbf{W}_2(e) & \text{if } e \in \mathbf{E}_2, \end{cases}$$

for all $e \in \mathbf{E}$*.*

In Figure 2 we report an example of union of two MSTs.

Definition 2. *Given a weighted graph* $\mathcal{G} = (\mathbf{V}, \mathbf{E}, \mathbf{W})$ *and a subset of the edges* $\mathbf{E}_1$*, we call* $\mathcal{G} - \mathbf{E}_1$ *the graph* $(\mathbf{V}, \mathbf{E} \setminus \mathbf{E}_1, \mathbf{W})$ *obtained by removing the edges in* $\mathbf{E}_1$ *from* $\mathcal{G}$*.*

Finally, we indicate $\mathcal{MST}(\cdot)$ the function that takes a graph $\mathcal{G}$ and returns a minimum spanning tree of $\mathcal{G}$, and with $E(\mathcal{G})$ the set of all edges in a $\mathcal{G}$. From now on, we are going to refer to $\mathbf{I}_t$ and I_t, both as images and graphs.
As said in the introduction, in this paper we address the problem of updating the minimum spanning tree of the image $\mathbf{I}_{t-1}$ each time that a new bunch of pixels I_t arrives. Three methods are compared to perform this task:

- **Method 1** (Naive MST): At the step t, compute from scratch the MST of the image $\mathbf{I}_t$.
- **Method 2** (Union MST): At the step t, compute the MST of the new image I_t, and then compute $\mathcal{MST}(\mathcal{MST}(\mathbf{I}_{t-1}) \cup \mathcal{MST}(I_t))$.
- **Method 3** (Proposed Method): At the step t, the graph $\mathcal{G}$ made by the union of $\mathcal{MST}(\mathbf{I}_{t-1})$ and $\mathcal{MST}(I_t)$ may contain cycles. In fact, observe that each time the two trees do not share an edge in the joining column of pixels, that generate a cycle in the resulting graph, caused by the fact that there are two different ways, one in each tree, to go from one pixel to its neighbor (see Figure 2). So, the method, first finds all the candidate edges to form those cycles on the MST union, and then computes the MST on those edges. We call $E_{\mathbf{I}_{t-1}}$, the candidate edges in $\mathcal{MST}(\mathbf{I}_{t-1})$ to form cycles on $\mathcal{G}$. Similarly, we call E_{I_t}, the candidate edges in $\mathcal{MST}(I_t)$ to form cycles on $\mathcal{G}$. The method computes $\mathcal{MST}(E_{\mathbf{I}_{t-1}} \cup E_{I_t})$, and finally it returns $\mathcal{MST}(\mathbf{I}_t) = (\mathcal{MST}(\mathbf{I}_{t-1}) - E_{\mathbf{I}_{t-1}}) \cup (\mathcal{MST}(I_t) - E_{I_t}) \cup (\mathcal{MST}(E_{\mathbf{I}_{t-1}} \cup E_{I_t}))$

The underlying idea of method 3 is that only a little part of the edges in the two MSTs need to be removed after merging. In fact, the number of edges to remove is equal to the number of cycles in the graph $\mathcal{MST}(\mathbf{I}_{t-1}) \cup \mathcal{MST}(I_t)$, and in the following proposition we prove that the number of cycles in this last graph limited by the number of common pixels between the two images $\mathbf{I}_{t-1}$ and I_t.

Proposition 1. *Given* $\mathbf{I}_{t-1}$ *and* I_t *two images of dimension* $n \times k$ *and* $m \times k$*, and let* $\mathcal{MST}(\mathbf{I}_{t-1})$ *and* $\mathcal{MST}(I_t)$ *minimum spanning trees respectively of* $\mathbf{I}_{t-1}$ *and* I_t*. Finally, let*

$$\mathcal{G} = \mathcal{MST}(\mathbf{I}_{t-1}) \cup \mathcal{MST}(I_t)$$

the union of the two trees. The number of cycles in $\mathcal{G}$ *is* $k - h - 1$*, where* h *is the number of edges in common between the two trees.*

Proof. The number of edges in $\mathcal{MST}(\mathbf{I}_{t-1})$ is $nk - 1$, while the number of edges in $\mathcal{MST}(I_t)$ is $mk - 1$. Thus the number of edges in $\mathcal{G}$ is:

$$nk - 1 + mk - 1 - h = (n + m)k - h - 2.$$

We can observe that $\mathcal{G}$ contains a spanning tree for the image $\mathbf{I}_t$, which has $(n + m - 1)k - 1$ edges. So the number of cycles in $\mathcal{G}$ is

$$(n + m)k - h - 2 - (n + m - 1)k + 1 = k - h - 1.$$

□

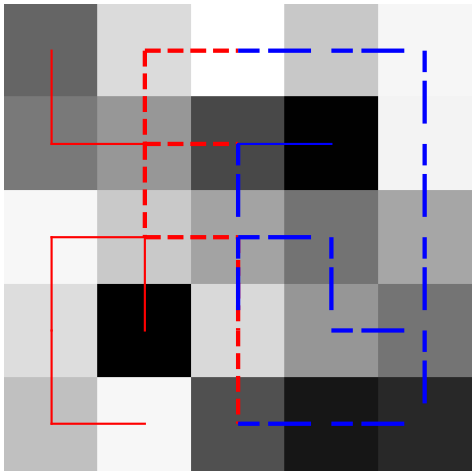

Fig. 2. $\mathcal{MST}(\mathbf{I}_0)$ in red and $\mathcal{MST}(I_1)$ in blue. $E_{\mathbf{I}_0}$ and E_{I_1} in bold and dashed, edges linking common pixels and candidate to form cycles on the union of the two MST.

In order to compute the edges in $E_{\mathbf{I}_{t-1}}$ (respectively in E_{I_t}), we compute all the paths in $\mathcal{MST}(\mathbf{I}_{t-1})$ (respectively $\mathcal{MST}(I_t)$) from the sharing pixels between $\mathbf{I}_{t-1}$ and I_t, to the upper-right pixel in $\mathbf{I}_{t-1}$ (respectively upper-left pixel in I_t). We called this function *edges_in_cycles*.
In Procedure 1 we report the pseudo code for method 3.

We conclude this section proving that all the three methods generate a minimum spanning tree for the image $\mathbf{I}_t$.

Theorem 1. *All the proposed methods return a minimum spanning tree for the image* $\mathbf{I}_t$*, for each* t*.*

Procedure 1 Method 3

Input: A streaming image $\mathbf{I}_t$

Output: A minimum spanning tree MST for the image $\mathbf{I}_t$

procedure METHOD 3
  $T_0 \leftarrow \mathcal{MST}(I_0)$
  $E_{\mathbf{I}_0} = \textit{edges\_in\_cycles}(T_0)$
  $\mathbf{T}_0 \leftarrow T_0 - E_{\mathbf{I}_0}$
  $\mathcal{MST}(\mathbf{I}_0) = \mathbf{T}_0 \cup E_{\mathbf{I}_0}$
  while a new image I_t arrives **do**:
    $T_i \leftarrow \mathcal{MST}(I_t)$
    $E_{I_t} \leftarrow \textit{edges\_in\_cycles}(T_i)$
    $\mathbf{T}_i \leftarrow T_i - E_{I_t}$
    $T \leftarrow \mathcal{MST}(E_{\mathbf{I}_{t-1}} \cup E_{I_t})$
    $\mathcal{MST}(\mathbf{I}_t) \leftarrow \cup_{i=0}^{t} \mathbf{T}_i \cup T$
    // Fetching edges in cycles for the next iteration
    $E_{\mathbf{I}_t} \leftarrow \textit{edges\_in\_cycles}(\mathbf{T}_i)$
    $\mathbf{T}_i \leftarrow \mathbf{T}_i - E_{\mathbf{I}_t}$

Proof. The proof for the first method is straightforward since for each t it returns $\mathcal{MST}(\mathbf{I}_t)$. Concerning the other two methods, we are going to prove that the resulting graphs $\mathcal{G}$ respect the following properties:

- $\mathcal{G}$ is connected,
- $\mathcal{G}$ is a spanning tree,
- the sum of all weights of edges in the graph is minimal.

Method 2 Let $\mathcal{G}_2 = \mathcal{MST}(\mathcal{MST}(\mathbf{I}_{t-1}) \cup \mathcal{MST}(I_t))$ be the result of the second algorithm for the image $\mathbf{I}_t$. It is easy to check the first two properties since $\mathcal{G}_2$ is a MST on a graph containing all the vertices in $\mathbf{I}_t$. Thus we only need to prove that $\mathcal{G}_2$ is minimal. By contradiction, suppose that $\mathcal{G}_2$ is not minimal. Thus, there exists an edge $e = (u, v) \in E(\mathbf{I}_t)$, such that $\mathcal{T} \subseteq \mathcal{G}_2 \cup e$ is a spanning tree which sum of edges is smaller than the sum of the edges in $\mathcal{G}_2$. Since $e \in \mathbf{I}_t$ then either $e \in \mathbf{I}_{t-1}$ or $e \in I_t$. Without loss of generality, let suppose that $e \in I_t$, then $\mathcal{MST}(I_t) \cup e$ contains a smaller spanning tree than $\mathcal{MST}(I_t)$, and this is a contradiction.

Method 3 Let $\mathcal{G}_3 = (\mathcal{MST}(\mathbf{I}_{t-1}) - E_{\mathbf{I}_{t-1}}) \cup (\mathcal{MST}(I_t) - E_{I_t}) \cup (\mathcal{MST}(E_{\mathbf{I}_{t-1}} \cup E_{I_t}))$ be the graph obtained using method 3 on image $\mathbf{I}_t$. First of all, we prove that $\mathcal{G}_3$ is connected. Let $u, v \in \mathbf{I}_t$, and $\mathcal{G} = \mathcal{MST}(\mathbf{I}_{t-1}) \cup \mathcal{MST}(I_t)$. Since $\mathcal{G}$ is connected, it is possible to find a path $\pi = \{v = v_0, \ldots, v_n = u\}$ contained in $\mathcal{G}$. Using the fact that $\mathcal{G}_3 \subseteq \mathcal{G}$, we can conclude that either $\pi \subseteq \mathcal{G}_3$ or it exists an edge $e = (v_i, v_{i+1})$ such that $e \notin E(\mathcal{G}_3)$. In this last case, by construction $e \in E_{\mathbf{I}_{t-1}} \cup E_{I_t}$, but $e \notin \mathcal{MST}(E_{\mathbf{I}_{t-1}} \cup E_{I_t})$. However, since $\mathcal{MST}(E_{\mathbf{I}_{t-1}} \cup E_{I_t})$ is a connected tree we can find another path $\pi_1 = \{v_i = w_0, \ldots, w_m = v_j\} \subseteq \mathcal{MST}(E_{\mathbf{I}_{t-1}} \cup E_{I_t})$ and thus in $\mathcal{G}_3$. We just proved that for each couple of vertices u, v in $\mathbf{I}_t$, we can construct a path that joins u and v entirely contained in $\mathcal{G}_3$ starting from a path $\pi \subseteq \mathcal{G}$, so $\mathcal{G}_3$ is connected. Secondly, the reason why $\mathcal{G}_3$ is a spanning tree comes directly from Proposition 1. In fact $\mathcal{G}_3$ is obtained from $\mathcal{G}$, removing all the cycles that this last graph contains, and using the fact that $\mathcal{G}_3$ connects all the vertices in $\mathbf{I}_t$, then we can say that $\mathcal{G}_3$ is a spanning tree. Finally, the same argument used for Method 2 can be applied to prove that the sum of weights of edges in $\mathcal{G}_3$ is minimal. □

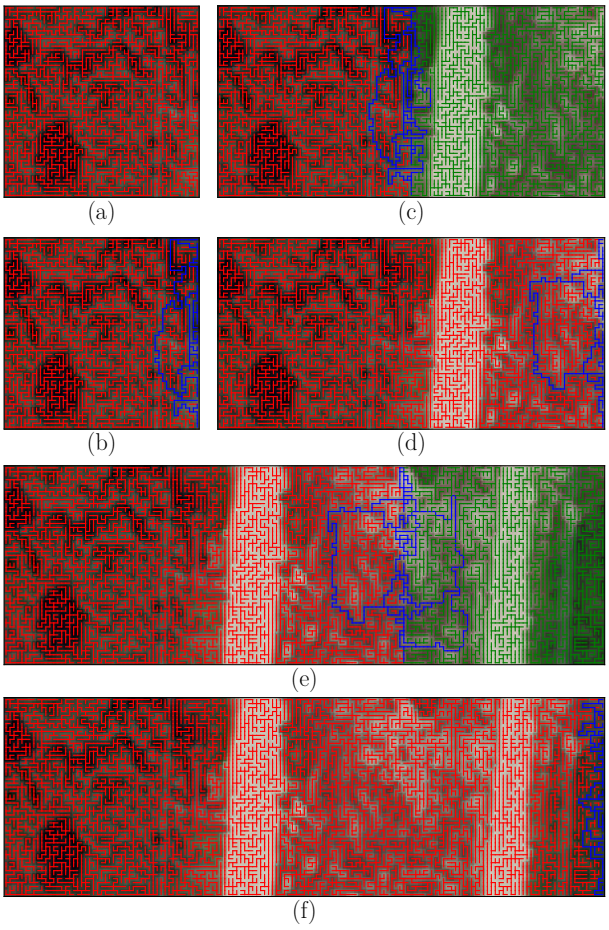

Fig. 3. Three iterations of the proposed method. (a) The MST of the image $\mathbf{I}_0$, while in (b) we report in blue the edges in $E_{\mathbf{I}_0}$ and in red the edges in $\mathbf{T}_0$. In (c) a new image I_1 arrives. First the MST of I_1 is constructed, and all the candidate edges that form cycles in the union between $\mathcal{MST}(\mathbf{I}_0) \cup \mathcal{MST}(I_1)$ are found. In red the edges in $\mathbf{T}_0$, in green $\mathcal{MST}(I_1) - E_{I_1}$, and in blue the edges in $E_{\mathbf{I}_0} \cup E_{I_1}$. Note that only blue graph contains cycles. (d) The MST of I_1 is $\mathcal{MST}(\mathbf{I}_1) = (\mathcal{MST}(\mathbf{I}_0) - E_{\mathbf{I}_0}) \cup (\mathcal{MST}(I_1) - E_{I_1}) \cup (\mathcal{MST}(E_{\mathbf{I}_0} \cup E_{I_1}))$. Like in (b), in blue $E_{\mathbf{I}_1} = \textit{edges\_in\_cyles}(\mathcal{MST}(\mathbf{I}_1))$ and in red $\mathbf{T}_0 \cup \mathbf{T}_1$. Again, figures (e) and (f) show the results obtained after that the image I_2 arrives.

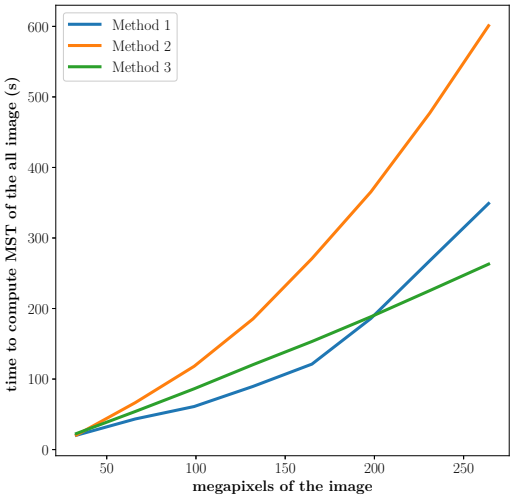

Fig. 4. Time comparison between three methods described in Section 2. We tested the three methods on images of different sizes. Remark that the execution times of our proposed method is linear in the size of images.

3. EXPERIMENTAL RESULTS AND DISCUSSIONS

We conducted our experiments on a satellite high-resolution image of San Diego, that is a 33-mega-pixels image (7038×4723 in RGB channels). The CPU of the machine used for the tests is Intel 3.00 GHz Xeon with 32GB RAM.

In the first experiment, we compared the execution times of our methods on images of different sizes all generated starting from the image described above and repeating it several times along the vertical axis. The goal was to test how our algorithms perform on really big images. We run our methods on each of the generated images and we measured how long each method took to build the MST for the complete image given as input. While Method 1 charges each image directly in memory and then compute the MST, in method 2 and 3 we divided each image in several blocks $I_0, \ldots, I_n$ each of size (7038×4723). Since we repeated the images along the vertical axis, we made stream those blocks from the bottom side of the image instead of the right side. In Figure 4 is possible to see that the execution times of our proposed increase linearly with the size of the images. We could not experiment Method 1 with bigger images because it allocates all the system memory and the program crashes.

Second, we compute in streaming a level of the *quasi-flat zones* hierarchy [12][13]. It is obtained by computing the strongly connected component of a threshold on edge-weighted graph, the weight being a gradient of intensity [13]. Equivalently, these connected component can be obtained directly from the MST. The corresponding connected components are also called *lambda-flat zones* [14][15]. Using the proposed method it is possible to find on the fly some of the *lambda-flat zones* in the image $\mathbf{I}_t$. In fact, our method returns the MST of $\mathbf{I}_t$ as the sum of two components. The first is $\mathbf{T} = \cup_i \mathbf{T}_i$ that is made by edges that we do not need to touch any more, while the second is $E_{\mathbf{I}_t}$. So, thresholding edges in $\mathbf{T}$ and removing all the connected components that contains pixels adjacent to edges in $E_{\mathbf{I}_t}$, we obtain *lambda-flat zones* that will be in any $\mathbf{I}_s$, for any $s \geq t$. In Figure 5 we report the results obtained on our test image. Pixels in black, (b) and (c), are those for which we cannot assign a label at the time we process image I_t.

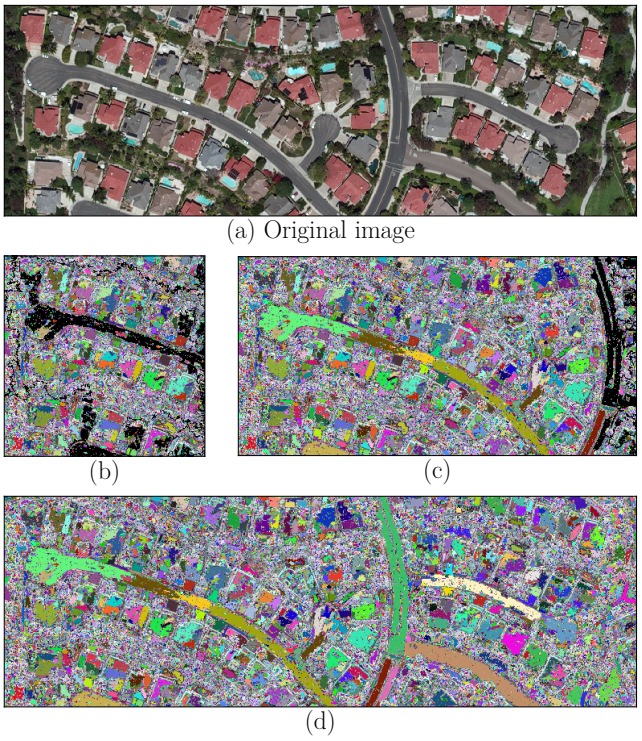

Fig. 5. (b) and (c) Lambda-flat zones ($\lambda = 5$) computed on streaming by the proposed algorithm on MST. Pixels in black are those that are waiting to a definite label in the streaming process. (d) Final result on the complete image.

4. CONCLUSION

This paper studies the problem of minimum spanning tree streaming on images. We proposed new algorithms for this task and a detail analysis of the solution is provided. Our initial experiments in morphological segmentation provide supporting evidence for using our proposed MST streaming for more difficult task as anisotropic filtering [2] or spatiotemporal background subtraction [7].

5. REFERENCES

[1] B. Ma, A. Hero, J. Gorman, and O. Michel, "Image registration with minimum spanning tree algorithm," in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 1. IEEE, 2000, pp. 481–484.

[2] L. Bao, Y. Song, Q. Yang, H. Yuan, and G. Wang, "Tree filtering: Efficient structure-preserving smoothing with a minimum spanning tree," *IEEE Trans. Im. Proc.*, vol. 23, no. 2, pp. 555–569, 2014.

[3] W.-C. Tu, S. He, Q. Yang, and S.-Y. Chien, "Real-time salient object detection with a minimum spanning tree," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2334–2342.

[4] F. Meyer, "Hierarchies of partitions and morphological segmentation," in *International Conference on Scale-Space Theories in Computer Vision*. Springer, 2001, pp. 161–182.

[5] C. Couprie, L. Grady, L. Najman, and H. Talbot, "Power watershed: A unifying graph-based optimization framework," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 7, pp. 1384–1399, 2011.

[6] A. Fehri, S. Velasco-Forero, and F. Meyer, "Prior-based hierarchical segmentation highlighting structures of interest," in *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, 2017, pp. 146–158.

[7] M. Chen, Q. Yang, Q. Li, G. Wang, and M.-H. Yang, "Spatiotemporal background subtraction using minimum spanning tree and optical flow," in *European Conference on Computer Vision*. Springer, 2014, pp. 521–534.

[8] Q. Yang, "A non-local cost aggregation method for stereo matching," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1402–1409.

[9] L. Gueguen, "Classifying compound structures in satellite images: A compressed representation for fast queries," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 4, pp. 1803–1818, 2015.

[10] Y. Ma, H. Wu, L. Wang, B. Huang, R. Ranjan, A. Zomaya, and W. Jie, "Remote sensing big data computing: Challenges and opportunities," *Future Generation Computer Systems*, vol. 51, pp. 47–60, 2015.

[11] S. Li, S. Dragicevic, F. A. Castro, M. Sester, S. Winter, A. Coltekin, C. Pettit, B. Jiang, J. Haworth, A. Stein *et al.*, "Geospatial big data handling theory and methods: A review and research challenges," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 115, pp. 119–133, 2016.

[12] F. Zanoguera and F. Meyer, "On the implementation of non-separable vector levelings," in *in (H. Talbot and R. Beare Eds.) Mathematical Morphology, Proc. of ISMM2002*. CSIRO Publishing, 2002.

[13] L. Najman, J. Cousty, and B. Perret, "Playing with kruskal: Algorithms for morphological trees in edge-weighted graphs," in *ISMM*, 2013, pp. 135–146.

[14] P. Soille, "Constrained connectivity for hierarchical image partitioning and simplification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 7, pp. 1132–1145, Jul. 2008.

[15] L. Gueguen, S. Velasco-Forero, and P. Soille, "Local mutual information for dissimilarity-based image segmentation," *Journal of mathematical imaging and vision*, vol. 48, no. 3, pp. 625–644, 2014.