Incremental Updating of 3D Topological Maps to Describe Videos

Guillaume Damiand, Sylvain Brandel, Donatello Conte

Incremental Updating of 3D Topological Maps to Describe Videos

Guillaume Damiand[1], Sylvain Brandel[1], and Donatello Conte[2]

[1] Université de Lyon, CNRS, LIRIS, UMR5205, F-69622 France
{guillaume.damiand|sylvain.brandel}@liris.cnrs.fr
[2] Université François-Rabelais de Tours, LI EA 6300, F-37200 France
donatello.conte@univ-tours.fr

Abstract. A topological map is an efficient mathematical model for representing an image subdivision where all cells and adjacency relations between elements are represented. It has been proved to be a very good tool for video processing when video is seen as a 3D image. However the construction of a topological map for representing a video needs the availability of the complete image sequence. In this paper we propose a procedure for online updating a topological map in order to build it as the video is produced, allowing to use it in real time.

Keywords: 3D Topological Maps; Video processing; Combinatorial Maps.

1 Introduction

Many works have studied models representing partitions of an image. Topological data structures describe images as a set of elements and their adjacency relations. The most famous example is the Region Adjacency Graph (RAG) [Ros74] which represents each region by a vertex, and where neighboring regions are connected by an edge. But the RAG suffers from several drawbacks as it does not represent multiple adjacency or makes no differences between enclosure and adjacency relations. Topological maps [BDF01] have been used to solve these issues. A topological map is a mathematical model that represents an image subdivision. It aims to allow the use of topological and geometrical features of the subdivision in image processing. This kind of features have been used effectively for image segmentation [DR03] and more recently for video denoising [CD14].

Approaches for processing and understanding video data are mostly based by statistical representations; they can be broadly divided in two categories: some methods represent videos in a classical way that is an images sequence in which each image is represented by its pixels [Jai89,KC01], other methods represent videos based on "mid-level" vision, that is using some features to represent and to analyze data [WA94]. In the last category, motion features, moving objects and trajectories are widely used as representation of video data [KZK03,LZ09]. Structural representation is rarely used for representing videos. In fact, if video metadata are stably represented by structured data [GNHY98,HA,NMZ05], structural

representation of video content is still an open problem [MOF13], and to the best of our knowledge graph representation was used only for coding videos and not for processing or understanding them.

In our approach a video is seen as a 3D image: a temporal sequence of 2D images is considered as a 3D image in which each voxel is considered as a temporal pixel, described by three coordinates (x, y, t), (x, y) being the spatial coordinates and t being the temporal coordinate. The 3D image is thus represented by a topological map. Then, topological features are proved to be very useful for video analysis and processing [CD14].

However the main problem in representing a video as a topological map is that for building this kind of data structure the complete video sequence has to be available (to represent it as a 3D image). Therefore algorithms using topological map are not available for real time processing.

The aim of this paper is to propose a method for online updating a topological map as the images of the sequence are produced. We propose two algorithms. The first one allows to remove k slices from the beginning of a given topological map. The second one allows to add k slices after the end of a given topological map. Combining these two operations allows to move a slicing window through a whole video.

The remainder of the paper is organized as follows: in Sect. 2 we introduce preliminary notions on topological maps that will be used afterwards. Section 3 describes the proposed operations by giving the algorithms and illustrating them on examples. In Sect. 4, some results show the complexity of our operations in the context of objects detection for video surveillance applications. Lastly, we conclude and give some perspectives in Sect. 5.

2 Preliminary Notions

A 3D topological map is an extension of a combinatorial map used to represent a 3D image partition. Therefore we first recall the notions on combinatorial maps and then we introduce notions on topological maps that are used in this work.

A combinatorial map is a mathematical model of representation of a space subdivision in any dimension. This is done using abstract elements, called *darts*, and applications defined on these darts, called β_i. The formal definition [Lie94,DL14] is the following:

Definition 1 (Combinatorial Map). *Let $n \geq 0$. An n dimensional combinatorial map (or n-map) is an algebra $C = (D, \beta_1, \ldots, \beta_n)$ where:*

1. *D is a finite set of* darts*;*
2. *β_1 is a* permutation *on D;*
3. *$\forall i$, $2 \leq i \leq n$, β_i is an* involution *on D;*
4. *$\forall i$, $1 \leq i \leq n-2$, $\forall j$, $i+2 \leq j \leq n$, $\beta_i \circ \beta_j$ is an* involution.

When two darts are linked with β_i, we say that they are i-sewed. An n-map represents each cell of the space subdivision implicitly by a set of darts. Moreover,

adjacency and incidence relation between these cells are encoded through β_i operators (see [Lie94,DL14] for more details).

A 3D topological map is a mathematical model which represents a 3D labeled image. This is a specialization of a 3-map in order to take into account the specific properties of a 3D image regarding any possible 3D subdivision. We can see in Fig. 1 an example of a 3D topological map.

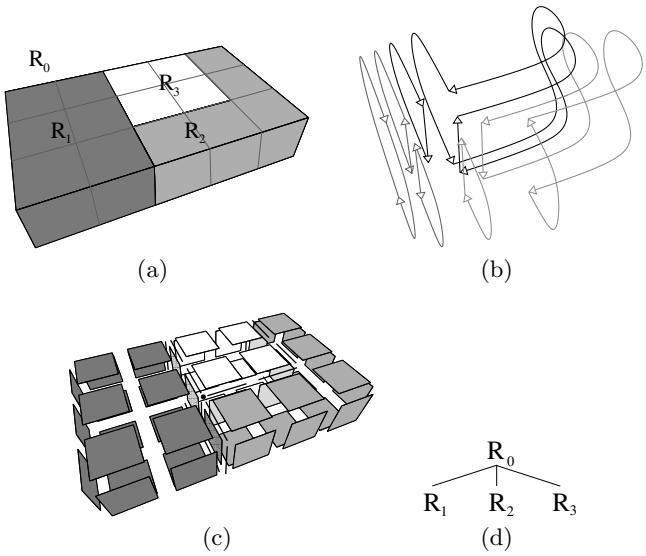

Fig. 1. Example of a 3D topological map. (a) A 3D labeled image. (b) The minimal 3-map. (c) The intervoxel matrix. (d) The region enclosure tree.

A 3D topological map is composed of three parts (see [Dam08] for details):

- a minimal 3-map representing the topology of the image (Fig. 1(b));
- an intervoxel matrix used to retrieve geometrical information associated to the 3-map. The intervoxel matrix is called the embedding of the combinatorial map (Fig. 1(c));
- an enclosure tree of regions (Fig. 1(d)).

The 3-map (Fig. 1(b)) represents each region of the image by describing its surfaces. There is always one external surface, and possibly some internal surfaces, one for each cavity in the region. These surfaces are described in the 3-map through their minimal form (minimal in number of cells, i.e. we cannot remove any cell without changing the topology of the 3-map). This minimality gives interesting properties to the 3-map that will be useful in algorithms:

- each face of the 3-map describes exactly a maximal contact surface between two regions;

– each edge of the 3-map describes exactly a maximal contact curve between several faces.

A last interesting property of the 3-map is to capture the topology of each region. We can thus compute for example Euler characteristics or Betti number of regions and use these characteristics in image and video processing algorithms, as done for example in [CD14].

The intervoxel matrix is the embedding of the 3-map describing the geometry of the regions in the image (Fig. 1(c)). Each cell of the map is associated with intervoxel elements representing geometrical information of the cell. A face in the combinatorial map is embedded by a set of surfels separating voxels of the two incident regions. The edges, which are the border of faces, are associated to sets of linels. The vertices, which are the border of edges, are embedded by pointels.

The enclosure tree of regions (Fig. 1(d)) represents the enclosure relations. Each region in the topological map is associated to a node in the enclosure tree. The nodes are linked together by the enclosure relation (a region R_i is enclosed in R_j if R_i is completely surrounded by region R_j [DD08]). To link this tree with the 3-map, each dart d of the map knows its belonging region (called *region(d)*). Each region R knows one of its dart called representative dart (called *rep(R)*).

A 3D topological map can directly represent a 2D video simply by considering the temporal sequence of 2D images as a 3D image: each voxel corresponds to a temporal pixel, described by three coordinates (x, y, t), (x, y) being the spatial coordinates and t being the temporal coordinate.

3 Incremental Updating of 3D Topological Maps

In order to incrementally update a 3D topological map, we introduce in this paper two operations. The first one, given in Algo. 1, allows to remove the k first slices of a given topological map. The second one, given in Algo. 2, allows to add k slices after a given topological map. Combining these two operations allows to move a slicing window through a whole video.

3.1 Remove Slices

Algorithm 1 presents the method allowing to remove the first `k` slices of a given topological map `T`.

– The first step consists here to split in surfels the surfaces of all the regions intersecting the slices to remove. We use for that the split operation defined in [Dup09]. The test if a region is intersected by the cutting plane is achieved thanks to the minimum and maximum voxel of the region (computed during the extraction of the 3D topological map).
– In the second step, all the faces describing the cutting plane are created, and then 2-sewn to darts around them in the existing 3-map. To retrieve the pair of darts to 2-sew, we use the geometry associated with each dart: two darts must be 2-sewn if they represent the same linel in reverse orientation.

Algorithm 1: Remove slices at the beginning of a 3D topological map

Input: `T`: A 3D topological map;
`k`: A number of slices.
Result: `T` is modified to remove its `k` first slices.

foreach *region* R *in* $\mathcal{T}$ **do**
  if *the minimum z value in* $R \leq k$ **and** *the maximum z value in* $R \geq k-1$ **then**
    split the surface of R in surfels;
Create faces `F` to describe the cutting plane;
2-sewn darts of `F` with darts of `T`;
foreach *dart* d *in* $\mathcal{T}$ **do**
  if *the z value of the triplet of* $d < k$ **then**
    remove the face containing d;
Simplify `T`;
Recompute the region enclosure tree;

- The third step consists in removing all the faces having one dart belonging to the first `k` slices. Thanks to the two previous steps, we are sure that these faces totally belong to the slices to remove.
- The last step consists in simplifying the combinatorial map in order to retrieve the minimality property and to recompute the region enclosure tree. We use for that the same algorithms that the ones used for the 3D topological map extraction [Dam08].

We can prove that this algorithm produces the 3D topological map describing the initial 3D image where we have removed its `k` first slices. Thus this proves directly the consistency of the operation: the modified 3D topological map represents the topology of the updated 3D image.

The remove slices operation is illustrated in Fig. 3. In this example, we consider a 2D topological map since it is really hard to visualize drawings of 3D topological maps, but things are similar in both dimensions by replacing *voxels* by *pixels* and *face* by *edge*. We consider the initial 2D topological map depicted in Fig. 2(b) where we want to remove the three first slices (thus `k` = 3). In the first step of Algo. 1, the borders of the two regions R_1 and R_2 are split in linels since some of their pixels belong to the three first slices. We obtain the 2-map shown in Fig. 3(a).

The second step inserts all the red edges in Fig. 3(a) which are new edges describing the cutting plane. These edges are sewn with their neighbor existing edges in the 2-map (illustrated by blue arcs in Fig. 3(b)). The next step (not drawn) consists in removing all the green edges from the 2-map. Indeed, all these edges belong to the first three slices. The last step of the algorithm simplifies the map. For that, each degree two vertex is removed. This produces the combinatorial map shown in Fig. 3(c) which is the 2D topological map of the image

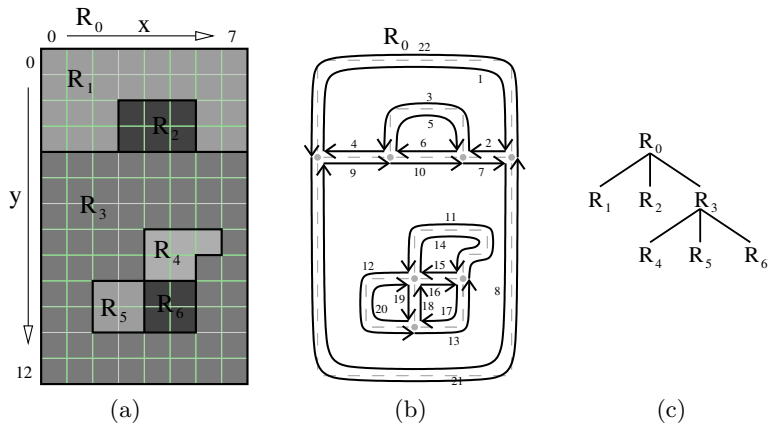

Fig. 2. Example of a 2D topological map. (a) A labeled image. (b) The minimal 2-map. (c) The region enclosure tree.

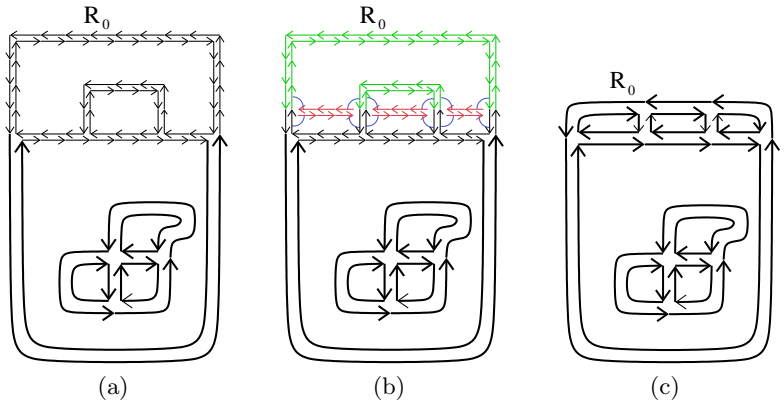

Fig. 3. Illustration of the removing of the three first slices of the 2D topological map depicted in Fig. 2(b). (a) The 2-map obtained after the split in linels of the borders of regions R_1 and R_2. (b) The 2-map obtained after the insertion of edges describing the cutting plane (in red). Then all the green edges are removed since they belong to the removed slices. (c) The final topological map obtained after the simplification of the previous 2-map.

obtained from the initial labeled image (given in Fig. 2(a)) where the three first slices were removed.

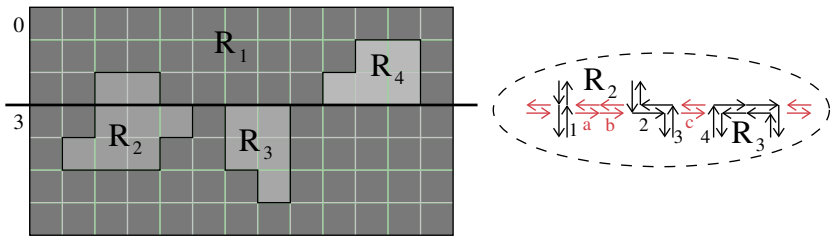

Fig. 4. Illustration of the three possible configurations of a region around the cutting plane $y = 3$. R_1 and R_2 are cut by the plane, i.e. they have pixels before and after it. R_3 touches the plane and have all its pixels after it. R_4 touches the plane and have all its pixels before it.

Figure 4 illustrates the three possible configurations of a region around the cutting plane. In these three cases, the surface of the region must be split in surfels in order to be correctly 2-sewn with the new faces describing the cutting plane. Other regions are either totally smaller or greater than the cutting plane. Smaller regions are totally removed by the cutting, while greater ones are totally kept. In both cases, there is no need to subdivide their faces. The example of region R_4 illustrates the -1 used in Algo. 1. Indeed in such a case, the maximal z value of the region is equal to the value of the plane minus 1, and such a region must have its border split in linels to be correctly sewn with the new edges.

The right part of Fig. 4 illustrates the method used to 2-sew correctly the new edges describing the cutting plane with the existing edges. The principle consists in turning around the vertex to sew until finding an existing dart. For dart a, we found dart 1. For dart b, we found dart 2 (because here we 2-sew the extremity of dart b and not their origin). For the origin of dart c we found dart 3 and for its extremity we found dart 4. Note that the 2-sew process is done for one dart per edge since at each step we 2-sew a dart and its β_3.

3.2 Add Slices

In order to add slices at the end of a 3D topological map `T`, the first step of Algo. 2 consists in splitting the external surface of `T` in surfels. This step allows the identification of the new slices with the external boundary of `T`. Indeed, only faces having the same topology can be identified in combinatorial maps (the two faces must be isomorphic i.e. they must be composed by the same number of darts), and it is simpler to split in surfels the external faces to identify instead of searching to modify each pair of faces in order to have the same topology.

In the second step, we compute the topological map $\mathtt{T}'$ describing the slices to add. During this computation, we do not simplify the external surface of $\mathtt{T}'$. Thus the external boundary of $\mathtt{T}'$ is composed by faces describing surfels.

This allows to directly identify in the third step the right faces of $\mathtt{T}$ with the left faces of $\mathtt{T}'$. Retrieving the faces to identify is done by using the geometry of the faces: both faces must have the same geometry but with reverse orientation. Testing if two faces have the same geometry is done by iterating simultaneously through the two cycles of darts, in reverse direction, while comparing the coordinates of the pairs of pointels associated with darts of the first face and darts of the second face.

Then it is enough to merge regions having same labels by removing the face separating them (step four) and to simplify the resulting 3-map (step five) in order to obtain the topological map describing the initial partition added with the $\mathtt{k}$ new slices.

Algorithm 2: Add slices at the end of a 3D topological map

Input: $\mathtt{T}$: A 3D topological map;
$\mathtt{I}'$: A 3D image containing $\mathtt{k}$ slices.
Result: $\mathtt{T}$ is modified to add the slices in $\mathtt{I}'$ at its end.

```
foreach face f of the infinite region of T do
    split f in surfels;
T'← compute the topological map of I' without simplifying its external surface;
Identify the right surfels of T with the left surfels of T';
foreach face f to the left of T' do
    if f separates two regions with the same label then
        remove f;
Simplify T;
```

As for the remove slices operation, we can prove that this algorithm produces the 3D topological map describing the initial 3D image where we have added the $\mathtt{k}$ new slices. Thus this proves directly the consistency of the operation: the modified 3D topological map represents the topology of the updated 3D image.

The add slices operation is illustrated in Fig. 5, once again in 2D in order to simplify the visualisation of combinatorial maps. The topological map depicted in Fig. 3(c) is considered where we want to add the three slices described by the 2D labeled image shown in Fig. 5(b). The first step of the algorithm consists in splitting the infinite face of the topological map to modify. We obtain the 2-map shown in Fig. 5(a). In the second step, the topological map of the new slices is computed while keeping its infinite face subdivided in linels (see Fig. 5(c)).

Now it is possible to identify the bottom linels of Fig. 5(a) and the top linels of Fig. 5(c). Note that during these identifications, the darts of the two identified parts belonging to the infinite regions are removed. Figure 6(a) shows the obtained 2-map. During the next step, all the adjacent regions having the

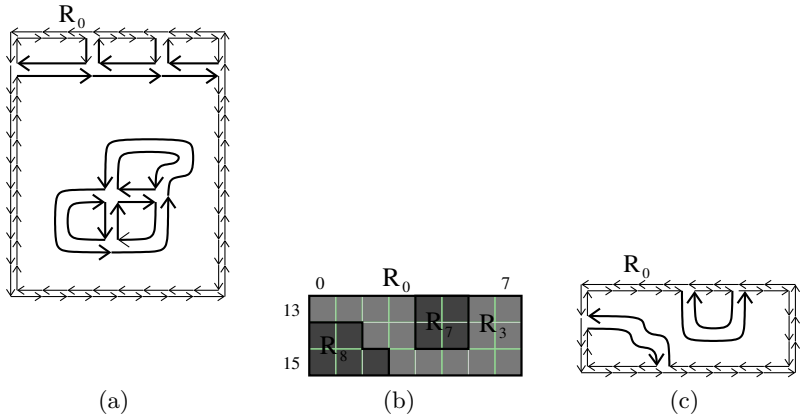

Fig. 5. Illustration of the adding of three new slices at the end of the 2D topological map depicted in Fig. 3(c). (a) The combinatorial map obtained from the initial topological map after the split of its infinite region in linels. (b) The labeled image corresponding to the three new slices. (c) Its corresponding topological map, having its external face subdivided.

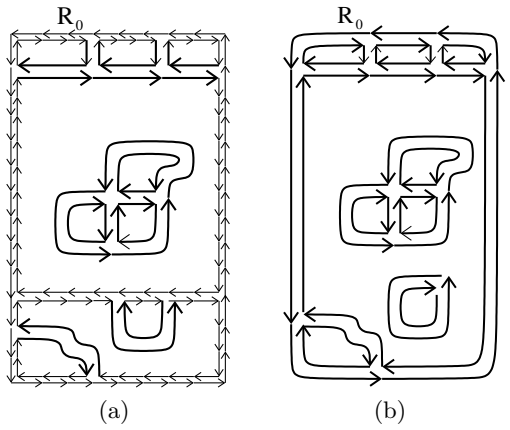

Fig. 6. Illustration of the adding of three new slices (cont). (a) The 2-map obtained after the identification of the bottom linels of Fig. 5(a) and the top linels of Fig. 5(c). (b) The final 2D topological map obtained after the merging of regions with same labels and the simplification step.

same label are merged by removing the edges separating them. Now it is enough to simplify the map in order to obtain the topological map representing the initial labeled image added by the new slices (shown in Fig. 6(b)).

Note that allowing these two operations to deal with k slices is an optimization comparing to the application of k successive operations for 1 slice. Indeed, each operation make some splits of existing cells following by some merges in order to retrieve the minimality property. Dealing with k slices allows to make only one split and one merge instead of k splits and k merges.

The add slices and the remove slices operations can directly be combined in order to propose a sliding window method which allows to iterate through all the frames of a given video. It is enough to call `remove_slices(1)` then `add_slices(1)`. However, in this case, we can avoid the simplification step and the region enclosure tree computation at the end of the `remove_slices` algorithm which will be done only once at the end of the `add_slices` operation.

Lastly, note that both proposed algorithms can be improved by avoiding some useless splits. For example for the add slice operation, we can only split in surfels the bottom boundary of the initial topological map and not all the infinite face. For the remove slice, we can split in surfels only the part of the concerned regions belonging to the removed slices instead of its entire boundaries. These optimizations will improve the speed of the operations since less cells will be split and then simplified, but the algorithm will become more complex since they are more different cases to consider.

4 Experiments

We have developed both operations `add_slices` and `remove_slices` in `3DTopoMap` [3], a software allowing to compute a 3D topological map from a 3D image.

Fig. 7. An image extracted from a video detection masks from the PETS 2010 database (768×576 pixels).

We used the PETS 2010 Dataset [pet] which is a well known database containing video sequences of moving people, used for the performance evaluation of tracking and surveillance systems. We use the video of detection masks: white pixels correspond to moving objects and black pixels to background (see Fig. 7).

From this database, we extracted a first 3D topological map containing 65 slices, where each slice corresponds to a frame in the video stream, and each frame has 768×576 pixels. We evaluated the time required by the three operations `add_slices`, `remove_slices` and `add_and_remove_slices` for different k (number of slices to add and/or to remove) starting from 1 and going to 64.

Results are shown in Fig. 8. We can first verify that the time processing for `add_slices` and `add_and_remove_slices` operations is linear regarding the number of added slices. More interestingly, we can see that the time of the `remove_slices` operation decreases linearly regarding the number of removed slices. This can be explained by the number of darts of the obtained 3-map which is smaller after having removed a bigger number of slices.

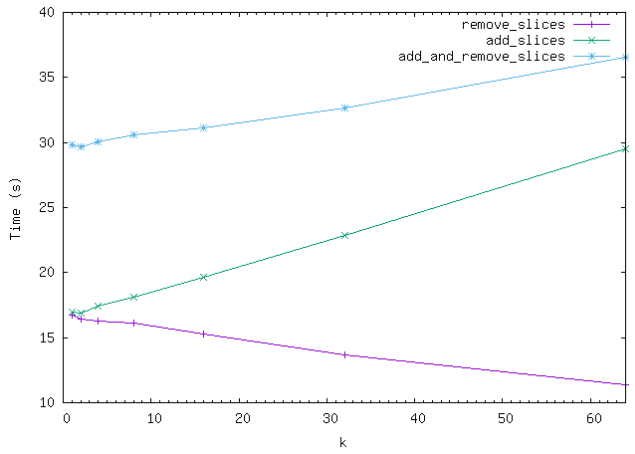

Fig. 8. Times evaluation (in seconds) of `remove_slices`, `add_slices` and `add_and_remove_slices` operations for $k \in \{1, 2, 4, 8, 16, 32, 64\}$ for an initial video sequence containing 65 slices. Each slice corresponds to a frame in the video stream. The resolution of each frame is 768×576 pixels.

5 Conclusion

In this paper, we have presented two operations defined on 3D topological maps. The first operation allows to remove the k first slices of an existing topological map. The second operation allows to add new slices at the end of an existing topological map. By combining these two operations, it is possible to define a

sliding window method which allows to iterate through all the frames of a given video.

Now we are working on the use of the new proposed operations in order to define efficient video processing. Our first goal is to update our previous denoising algorithm [CD14] in order to benefit from the sliding window method. This will allow us to improve the method by combining all the information coming from the different superposed windows. Moreover this will allow us to consider streamed videos.

Another future work is the optimization of our method in order to be able to propose real time processes. For that, we have started to integrate some parallelism in our algorithm. Indeed, several parts can be made in parallel, leading to speed up our method. A second possible optimization is to modify the two algorithms in order to avoid to split faces in surfels. Indeed, this step takes an important time and requires to simplify the map at the end of the algorithms. Modifying this step will give more complex but more efficient algorithms. With these two optimizations we are confident to obtain a method allowing to achieve real-time performance.

Lastly, we plan to propose other video processing algorithms based on 3D topological map and the new sliding window method. Following our first previous results [CD14], we think that integrating some topological information provided by the 3D topological map can improve several existing video processing algorithms.

Acknowledgment: This work has been partially supported by the French National Agency (ANR), project SoLStiCe ANR-13-BS02-0002-01.

References

[3DT] `3DTopoMap`: Topological 3d image processing software. `http://liris.cnrs.fr/guillaume.damiand/carte-topo3D.php?lang=en`.

[BDF01] Y. Bertrand, G. Damiand, and C. Fiorio. Topological map: Minimal encoding of 3D segmented images. In *Proc. of International Workshop on Graph-Based Representations in Pattern Recognition*, pages 64–73, Ischia, Italy, May 2001.

[CD14] D. Conte and G. Damiand. Remove noise in video with 3d topological maps. In *Proc. of Structural, Syntactic, and Statistical Pattern Recognition*, volume 8621 of *LNCS*, pages 213–222. Springer Berlin Heidelberg, 2014.

[Dam08] G. Damiand. Topological model for 3d image representation: Definition and incremental extraction algorithm. *Computer Vision and Image Understanding*, 109(3):260–289, March 2008.

[DD08] A. Dupas and G. Damiand. First results for 3D image segmentation with topological map. In *Proc. of International Conference on Discrete Geometry for Computer Imagery*, volume 4992 of *LNCS*, pages 507–518, Lyon, France, April 2008. Springer Berlin/Heidelberg.

[DL14] G Damiand and P. Lienhardt. *Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing*. A K Peters/CRC Press, 2014.

[DR03] G. Damiand and P. Resch. Split and merge algorithms defined on topological maps for 3D image segmentation. *Graphical Models*, 65(1-3):149–167, May 2003.

[Dup09] A. Dupas. *Opérations et Algorithmes pour la Segmentation Topologique d'Images 3D*. Thèse de doctorat, Université de Poitiers, Novembre 2009.

[GNHY98] Y. Gonno, F. Nishio, K. Haraoka, and Y. Yamagishi. Metadata structuring of audiovisual data streams on mpeg-2 system. In *Metastructures*, August 1998.

[HA] J. Hunter and L. Armstrong. A comparison of schemas for video metadata representation.

[Jai89] Anil K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, 1989.

[KC01] I. Koprinska and S. Carrato. Temporal video segmentation: A survey. *Signal Processing: Image Communication*, 16:477–500, 2001.

[KZK03] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis. A survey of video processing techniques for traffic applications. *Image and Vision Computing*, 21:359–381, 2003.

[Lie94] P. Lienhardt. N-Dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry and Applications*, 4(3):275–324, 1994.

[LZ09] X. Li and Y. Zheng. Patch-based video processing: a variational bayesian approach. *IEEE Transactions on Circuits and Systems for Video Technologies*, 19-1:27–40, 2009.

[MOF13] T. Maugey, A. Ortega, and P. Frossard. Graph-based representation and coding of multiview geometry. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1325–1329, 2013.

[NMZ05] C.W. Ngo, Y.F. Ma, and H.J. Zhang. Video summarization and scene detection by graph modeling. *IEEE Transactions on Circuits and Systems for Video Technologies*, 15:296–305, 2005.

[pet] Pets2001 dataset. http://www.cvg.rdg.ac.uk/pets2001/.

[Ros74] A. Rosenfeld. Adjacency in digital pictures. *Information and Control*, 26-1:24–33, 1974.

[WA94] J. Wang and E. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3-5:625–638, 1994.