Computer Vision Winter Workshop 2010, Libor Špaček, Vojtěch Franc (eds.)
Nové Hrady, Czech Republic, February 3–5
Czech Pattern Recognition Society

Representing the Surface of Objects by Combinatorial Pyramids*

Esther Antúnez[1], Helena Molina-Abril[1,2], and Walter G. Kropatsch[1]

[1]Pattern Recognition and Image Processing Group
Vienna University of Technology
{eortiz,habril,krw}@prip.tuwien.ac.at

[2]Computational Topology and Applied Mathematics Group
University of Seville
habril@us.es

Abstract This paper introduces a new method to represent the surface of objects using two dimensional combinatorial maps. The classical definition of two dimensional combinatorial maps is extended here by adding a "back face" that corresponds to the non–visible part of the object. As a first step, every object in the scene is extracted in one image pyramid, where levels are related by means of coordinates. Finally, an algorithm to complete the description of the surface is presented. Such representation is translation, rotation and scale invariant. It will allow to update information about movements and parts of the object that become visible, reducing the complexity and the computational time.

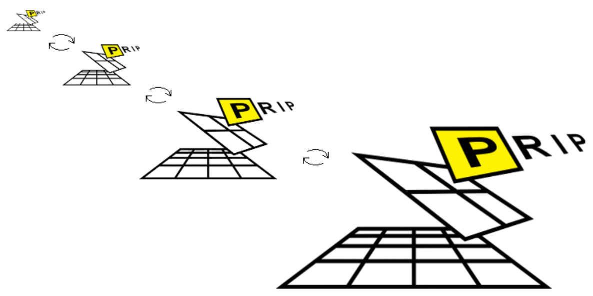


Figure 1: Image pyramid

1 Introduction

A large amount of computational resources is required to deal with image and video data. The main advantage of hierarchical structures is the rapid computation of a global information in a recursive manner. Thus, an hierarchical structure (like image pyramids) might be the answer to the time and space complexity in computer vision systems.

From a single image frame, the visible front surface of an object can be represented in terms of combinatorial maps or graphs. This extracted map (or graph) embeds the topological structure of the object, and describes in an efficient way its visible features. In order to efficiently represent the object in a hierarchical manner, a combinatorial pyramid [2] (or a dual graph pyramid [13]) can be constructed above the initial map (or graph).

The benefit of using combinatorial maps, is that they combine the advantages of dual graphs with an explicit orientation of the boundary segments of the embedded object. Moreover, the combinatorial map formalism is defined for any dimensions.

Nevertheless the use of such a complex structure implies some drawbacks. The extension of combinatorial maps to higher dimensions ($3D, 4D, \ldots, nD$) is feasible, but at the price of much higher memory costs [7].

In addition, in many applications in computer vision due to the fact that inner parts are usually not visible, what we really need to deal with is the surface of the object.

The goal of this paper is to extend the classical two dimensional combinatorial maps, in such a way that they represent two dimensional manifolds. This new representation is translation, rotation and scale invariant.

The inclusion of this new structure in the environment of image pyramids will be very useful when for example a three dimensional object is moving in a video sequence, and at each moment, some parts are occluded or partially visible.

The paper is structured as follows: In Section 2 we introduce the concepts of combinatorial maps and combinatorial pyramids. In Section 3 the description of an object of the initial image using a single pyramid is outlined. In Section 4 we present the procedure in which the invisible part is added to the previous representation in order to complete the object surface. The paper concludes in Section 5.

*Partially supported by the Austrian Science Fund under grants P20134-N13 and P18716-N13, the "Computational Topology and Applied Mathematics" PAICYT research project FQM–296, the "Andalusian research projects" PO6-TIC-02268 and P07-TIC-03106 and the "Science and Innovation Ministry" project MTM2009-12716

2 Recall on combinatorial pyramids

Image pyramids are a stack of images with decreasing resolutions [4] (see Figure 1). Such pyramids present many interesting properties within the Image Processing and Analysis framework such as [1]: Reducing the influence of noise by eliminating less important details in lower-resolution versions of the image, making the processing independent of the resolution of the regions of interest in the image, converting global features to local ones, reducing computational costs for many computations, etc.

The construction of the pyramid hierarchy follows the philosophy to reduce the amount of data at each higher level of the hierarchy by a reduction factor $\lambda > 1$ while preserving important topological properties like connectivity and inclusion. Every cell in level k is linked with cells on level directly below $k-1$. Those cells are called its children and cells on the level directly above $k+1$ its parent(s). The highest level of the pyramid is called its apex.

There exist different topological representations for structured objects that can be used in the hierarchical framework of image pyramids. These representations are dual graphs, combinatorial maps and generalized maps [11]. Image pyramids that contain these structures are called topological pyramids [12].

Combinatorial maps define a general framework, which allows to encode any subdivision of nD topological spaces orientable or non-orientable with or without boundaries. They were introduced in [6], at first as a planar graph model, and extended in [14] in dimension n to represent orientable or not-orientable quasi-manifolds.

Informally speaking, a combinatorial map is a mathematical model describing the subdivision of a space, and encoding all the cells of the subdivision and all the incidence and adjacency relations between the different cells. In this way, the topology of the space is fully described.

A more formal definition, describes a n dimensional combinatorial map as a $(n+1)$-tuple $M = (D, \beta_1, \beta_2, ..., \beta_n)$ such that D is the set of abstract elements called darts, β_1 is a permutation on D and the other β_i are involutions on D. An involution is a permutation whose cycle has the length of two or less. In the case of 2D, combinatorial maps may be defined with the triplet $G = (D, \alpha, \sigma)$, where D is the set of darts, σ is a permutation in D encoding the set of darts encountered when turning (counter) clockwise around a vertex, and α is an involution in D connecting two darts belonging to the same edge:

$$\forall d \in D, \alpha^2(d) = d \qquad (1)$$

Figure 2 shows an example of a combinatorial map. In Table 1 the values of α and σ for such a combinatorial map can be found. In this example counter clockwise (CCW) orientation has been chosen for σ. Note that every border cell is adjacent to the background/infinite region (Bg).

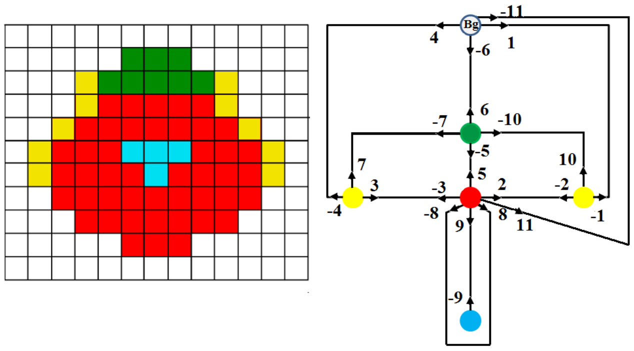


Figure 2: Example of a digital image and a combinatorial map encoding the partition

A combinatorial pyramid is a hierarchical stack of combinatorial maps successively reduced by a sequence of contraction or removal operations [3]. Combinatorial pyramids combine the advantages of dual graph pyramids with an explicit orientation of the boundary segments of the embedded object thanks to the permutation σ [2]. Moreover, using combinatorial maps, the dual graph is both implicitly encoded and updated. Finally, the combinatorial map formalism is defined in any dimensions.

3 Extracting objects from a combinatorial pyramid

Given an image pyramid, we would like to describe a single object contained on it. Let us recall that our aim is to obtain a complete representation of the object's surface. This representation must be translation, rotation and scale invariant.

The first step will consist in adding coordinates to the structure. The second step is to extract the pyramid that describes the object from the pyramid of the complete picture.

3.1 Relation between pyramid levels

In order to add coordinates to the structure, each cell (faces in case we are considering the dual map, or vertices in case we consider the primal) in the base level of the pyramid is attributed with the coordinates of the corresponding image pixels. For higher levels the coordinates of each cell are computed by inheritance from the surviving child to the parent. But storing all this information along the structure, will cause a big amount of redundant data. To avoid this fact, we compute and store at each level only the difference between the coordinates in one level with the coordinates in the level above, following the idea of the Laplacian pyramid [5]. This difference is called the child's correction vector:

$$d(c) = p(parent(c)) - p(c), \qquad (2)$$

where $d(c)$ is the child's correction vector, $p(c)$ are the child's coordinates and $p(parent(c))$ are the parent's

dart	1	-1	2	-2	3	-3	4	-4	5	-5	6	-6	7	-7	8	-8	9	-9	10	-10	11	-11
α	-1	1	-2	2	-3	3	-4	4	-5	5	-6	6	-7	7	-8	8	-9	9	-10	10	-11	11
σ	-11	10	5	-1	7	-8	-6	3	-3	-10	-7	1	-4	-5	11	9	8	-9	-2	6	2	4

Table 1: Values of α and σ for the combinatorial map in Figure 2

coordinates.

The original position of each cell can be reconstructed accurately by adding all the correction vectors from the apex. When the object is translated, such translation is applied to the coordinates of the apex. After that, the coordinates of the rest of the cells are reconstructed in the new position using the correction vectors in a top-down process, making the object representation invariant to translation.

In order to make our representation rotation and scale invariant, the orientation of the object is added as an attribute in the apex cells [9] as well as a scale factor. Such information replaces the previous one stored in the correction vectors. The orientation correction vector is called $\theta(c)$.

$$\theta(c) = o(parent(c)) - o(c), \tag{3}$$

where $\theta(c)$ is the child's orientation correction vector, $o(c)$ is the child's orientation and $o(parent(c))$ is the parent's orientation.
Once we have performed this process, all the information about position and orientation of the object is concentrated in the apex. Given the position and orientation in the top level the coordinates of each cell can be obtained by means of the following formulas:

$$\begin{aligned} x_c &= x_p + \lambda * r(c) * cos(o_p + \theta(c)) \qquad (4)\\ y_c &= y_p + \lambda * r(c) * sin(o_p + \theta(c)), \end{aligned}$$

being x_c, y_c, x_p, y_p the coordinates of the child and the parent respectively, λ the scale factor, $r(c)$ the child-parent euclidean distance, o_p the parent orientation and $\theta(c)$ the child's orientation correction vector.

One of the advantages of this representation is that for example in the analysis of video sequences it is not necessary to compute one pyramid for each frame; it is enough to apply all the transformations in the apex and then the computation of the correction vectors allows an accurate reconstruction of the whole pyramid in a top-down process.

3.2 Cutting out objects

In order to separate the data structure for each (moving) object, we would like to have one pyramid per object in the image and one for the background, keeping the connection between them by the coordinates previously added. This representation will allow to update movements and properties in an independent way.

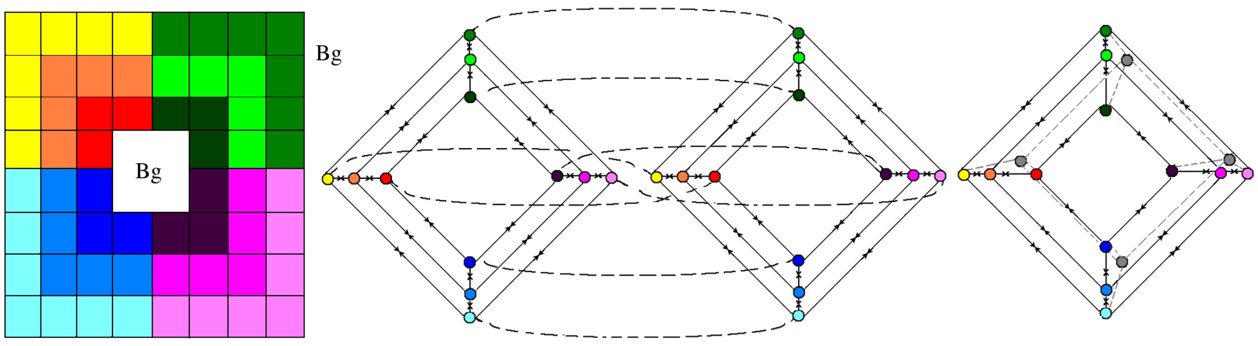
In a sequence of video frames it may happen that some objects in the scene move and others remain in the same position. Having one pyramid per object allows to update only the information corresponding to the moving objects. This fact decreases the number of updates in the structure from frame to frame and the processing time with the advantage that the connectivity between the objects and the background is always preserved by means of the stored coordinates in the apex of each pyramid.

In order to extract the object from the image pyramid, we first need to identify it. Several segmentation methods can be used for that purpose [10, 15], allowing to distinguish cells that belong to the object. Once those cells have been determined, the corresponding combinatorial map that represents the desired object at each pyramid level is extracted. Every cell that is not part of the object, will be considered as part of the background. Consequently, the border cells of the object will be adjacent to the background. Removal operations need to be performed on the initial map in order to delete parts that do not belong to the object. At each level a new combinatorial map that only corresponds to the desired object is obtained by adding to a removal kernel [3] the darts that do not belong to the object. After that we also have to update the information that relates two levels in the pyramid.

Our method may also be useful for articulated objects since it is possible to concentrate the information of each rigid part in the apex. The transformations can be applied to the apex of the moving part, and at the end only the part of the structure that has moved will be updated.

3.3 Preservation of connectivity

A significant advantage of using topological pyramids is that the connectivity is always preserved. This is not always true for other image processing tools (e.g. Photoshop). For instance, when some rotations are applied, these tools have to use some kind of interpolation or re-sampling when the coordinates are not integers. Figure 5 shows an example where a thin line is rotated. Figure 5b contains the result obtained using Photoshop. The red stars mark the new rounded coordinates of each point and the black squares are the position of the points estimated by the processing tool. Note that in that case the rotation results in a disconnected line (or in a thicker line if bilinear interpolation or anti-aliasing is applied) (see Figure 5b). Using our approach, although the position of the points change with the rotation, the relation between the cells in the graph remains the same since the edges of the graph always connect the same vertices. The black stars in Figure 5c point out that there is a connection between those points in the graph. In our case it is possible to apply again the same rotation in the other direction

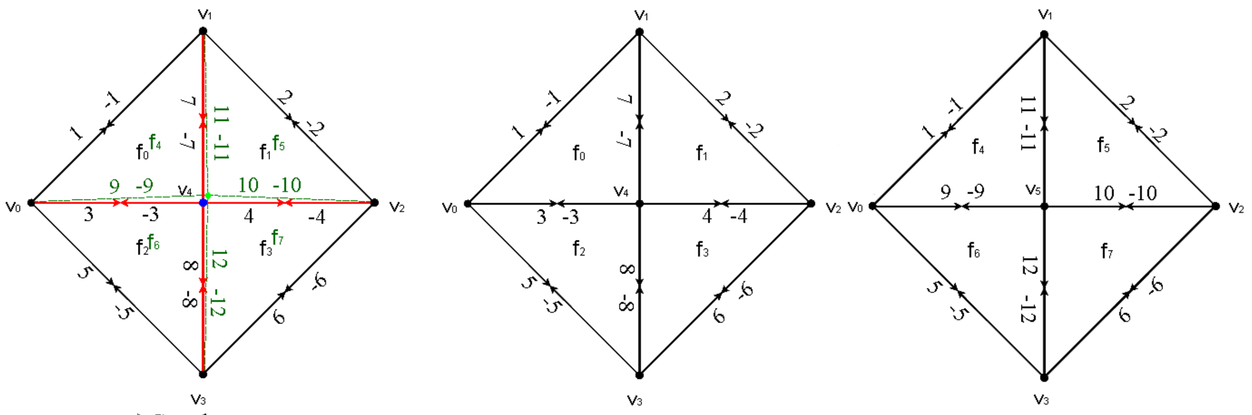


a) Digital object

b) Combinatorial maps showing
the correspondence between front and back

c) Complete map

Figure 3: Object surface completion process



a) Complete map

a) Front Map

b) Back Map

Figure 4: Object surface completion example

dart	1	-1	2	-2	3	-3	4	-4	5	-5	6	-6	7	-7	8	-8	9	-9	10	-10	11	-11	12	-12
σ_{step1}	5	7	-1	-4	1	8	-7	-6	3	6	-8	-2	2	-3	4	-5	5	-7	8	-2	-1	4	-3	6
σ_{step2}	9	7	11	-4	1	8	-7	-6	3	-12	-8	-10	2	-3	4	-5	5	-11	12	-2	-1	10	-9	6

Table 2: Values of σ and α of the complete combinatorial map in Figure 4

and obtain the original line while in other cases it is impossible since the line has been destroyed.

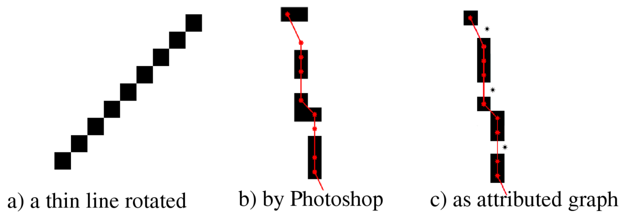


Figure 5: 50° rotation of a thin line

4 Object surface completion

Once we have the visible part (front face) of the object described using a combinatorial map, we would like to complete it with the non visible part of the object surface. To do this, we use one or more invisible surface patches in order to completely cover the surface of the volumetric object.

The main idea is to create a combinatorial map that is identical to the initial one, and that will conform the back face. These two maps (front and back) will be glued together, sharing the border cells. Therefore, only darts that are not adjacent to the background, are duplicated and conform the back face. Darts that are adjacent to the background (border of the object) will be common to the front and back faces (see Figure 3). In order to correctly cover every face of the object surface, we must use counter orientation for front and back maps. Thus, in case we use for example σ_{ccw} for the front maps, the back ones will be defined in terms of σ_{cw}.

The pseudo-code of the algorithm applied to every level of the pyramid is presented below. In this pseudo-code Bg represents the background face. The function B relates every dart with its corresponding one at the back face and vice versa.

```
Given a combinatorial map $G = (D, \alpha, \sigma_{ccw})$, where
$G$ contains $n$ edges
Step 1:
for every edge of the map, formed by the darts $d$ and
$-d$ (where $d = \alpha(-d)$) do
  if $d$ is not in $Bg$ and $-d$ is not in $Bg$ then
    Add two new back darts $b$ and $-b$ to $D$
    $\alpha(b) = -b$ ,       $\alpha(-b) = b$
    $\sigma(b) = \sigma_{cw}(d)$ , $\sigma(-b) = \sigma_{cw}(-d)$
    $B(d) = b$ ,       $B(-d) = -b$
  else
    $B(d) = d$ ,  $B(-d) = -d$
  end if
end for
Step 2:
for every dart $t \in D$ do
  if $B(t) = t$ and $B(\sigma(t)) = \sigma(t)$ then
    $\sigma(t) = B(\sigma_{cw}(t))$
  end if
  if $t$ is a back dart and $B(\sigma(t)) \neq \sigma(t)$ then
    $\sigma(t) = B(\sigma_{cw}(t))$
  end if
end for
```

In the first step of the algorithm, we include the darts that conform the back face, and initialize their σ and α values (see Figure 4). These darts are related with their correspondent front darts by the function B. For instance, in Figure 4 $B(3) = 9$ and $B(1) = 1$. In the second step, σ must be updated, in order to be consistent with the newly added darts. In Table 2, the values of σ that have to be updated are marked in red.

After applying this algorithm each level of the pyramid is a map, presenting the closed surface of the object in multiple resolutions. The pyramid representation can cope with this structure and the same operations can be applied as in the case of an image. In order not to store so much information, the back face can be contracted to a single cell.

As new visible parts of the surface would reveal previously invisible parts, the object representation will be incrementally updated automatically from observing the target object in a video sequence. This requires the registration of the visible parts and the replacement of some invisible patches. For this purpose, some of the existing methods dealing with multi-view integration might be useful [8, 17].

When some hidden parts appear, the new topological structure will be added into the previous 2D manifold to obtain the updated object representation [9]. The new view could imply more complexity on the topology of the object (like when the handle of a cup becomes visible) or could also imply the simplification of the former topological structure (for example a twisted torus, can be perceived from some view points as a double torus, see Figure 6).

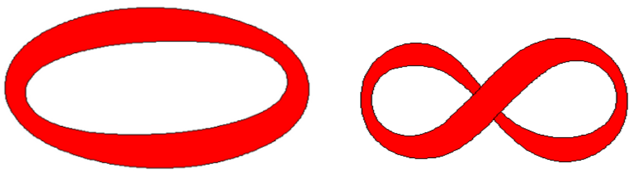


Figure 6: Two different views of a twisted torus

When all the observable parts of the object have been integrated in the object model, the topological structure of the target object is complete. This is the process defined as topological completion [16]. Due to the fact that only visible parts of the object are considered, parts that never become visible will not be included on its representation. Therefore the word "complete" is applied here in a non rigorous sense.

5 Conclusion

A new representation of object surfaces using two dimensional combinatorial maps has been presented in this paper. This representation is translation, rotation and scale invariant and preserves the topological properties of the visible parts of the object. In this way we reduce the complexity and the amount of stored data, that would be higher in case of using pure three dimensional structures.

Another major advantage is that the computation time in the analysis of video sequences is considerably reduced due to the reduction of updates from frame to frame since the movement of the objects modifies only one cell in the structure (the apex). This last advantage allows to deal with real time processing requirements.

As future work we plan to develop efficient methods to update this structure as new visible parts of the object become visible, and make experimentations and comparisons to other methods.

References

[1] M. Bisterand, J. Cornelis, and A. Rosenfeld. A critical view of pyramid segmentation algorithms. PRL, pages 605–617, 1990.

[2] L. Brun and W.G. Kropatsch. Introduction to combinatorial pyramids. Digital and Image Geometry, Lecture Notes in Computer Science, pages 108–128, 2001.

[3] L. Brun and W. G. Kropatsch. Construction of combinatorial pyramids. In E.R. Hancock and M. Vento, editors, GbRPR, volume 2726 of Lecture Notes in Computer Science, pages 1–12. Springer, 2003.

[4] P. Burt, T-H. Hong, and A. Rosenfeld. Segmentation and estimation of image region properties through cooperative hierarchial computation. IEEE Transactions on Systems, Man and Cybernetics, pages 802–809, December 1981.

[5] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. IEEE Transactions on Communications, COM-31,4:532–540, 1983.

[6] J. Edmonds. A combinatorial representation of polyhedral surfaces. Notices of the American Mathematical Society, page 646, 1960.

[7] Thomas Illetschko. Minimal combinatorial maps for analyzing 3d data. Technical Report PRIP-TR-110, PRIP, TU Wien, 2006.

[8] K. Kolev and D. Cremers. Integration of multiview stereo and silhouettes via convex functionals on convex domains. In ECCV '08: Proceedings of the 10th European Conference on Computer Vision, pages 752–765, Berlin, Heidelberg, 2008. Springer-Verlag.

[9] W. G. Kropatsch. When pyramids learned walking. In E. Bayro-Corrochano and J-O. Eklundh, editors, CIARP, volume 5856 of Lecture Notes in Computer Science, pages 397–414. Springer, 2009.

[10] W. G. Kropatsch, Y. Haxhimusa, and A. Ion. Multiresolution image segmentations in graph pyramids. In A. Kandel, H. Bunke, and M. Last, editors, Applied Graph Theory in Computer Vision and Pattern Recognition, volume 52 of Studies in Computational Intelligence, pages 3–41. Springer, 2007.

[11] W. G. Kropatsch, Y. Haxhimusa, and P. Lienhardt. Hierarchies relating topology and geometry. Cognitive Vision Systems, pages 199–220, October 2003.

[12] W. G. Kropatsch and H. Molina-Abril. Open issues and chances of topological pyramids. In A. Ion and W. G. Kropatsch, editors, Computer Vision Winter Workshop, pages 121–126, 2009.

[13] W. G. Kropatsch, C. Reither, D. Willersinn, and G. Wlaschitz. The dual irregular pyramid. In D. Chetverikov and W. G. Kropatsch, editors, CAIP, volume 719 of Lecture Notes in Computer Science, pages 31–40. Springer, 1993.

[14] P. Lienhardt. Subdivisions of n -dimensional spaces and n-dimensional generalized maps. Symposium on Computational Geometry, pages 228–236, 1989.

[15] R. Marfil, L. Molina-Tanco, A. Bandera, J.A. Rodríguez, and F. Sandoval. Pyramid segmentation algorithms revisited. Pattern Recogn., 39(8):1430–1451, 2006.

[16] L. A. Mateos and W. G. Kropatsch. Multi-view integration for a rotating 3d object. Submitted to the Computer Vision Winter Workshop 2010, Nove Hrady, Czech Republic, February 2010.

[17] J. Maver, A. Leonardis, and F. Solina. Planning the optimal set of views using the max-min principle. In Jan-Olof Eklundh, editor, ECCV (2), volume 801 of Lecture Notes in Computer Science, pages 117–122. Springer, 1994.