Colorization and Illumination of 2D Animations Based on a Region-tree Representation

Renata Nascimento *, Fabiane Queiroz †, Allan Rocha ‡, Tsang Ing Ren †, Vinicius Mello §, Adelailson Peixoto ¶.
* Department of Mathematics, PUC–Rio, Rio de Janeiro, Brazil.
† Department of Computer Science, UFPE, Pernambuco, Brazil.
‡ Department of Computer Science, PUC–Rio, Rio de Janeiro, Brazil.
§ Institute of Mathematics, UFBA, Salvador, Brazil.¶ Institute of Mathematics, UFAL, Maceió, Brazil.

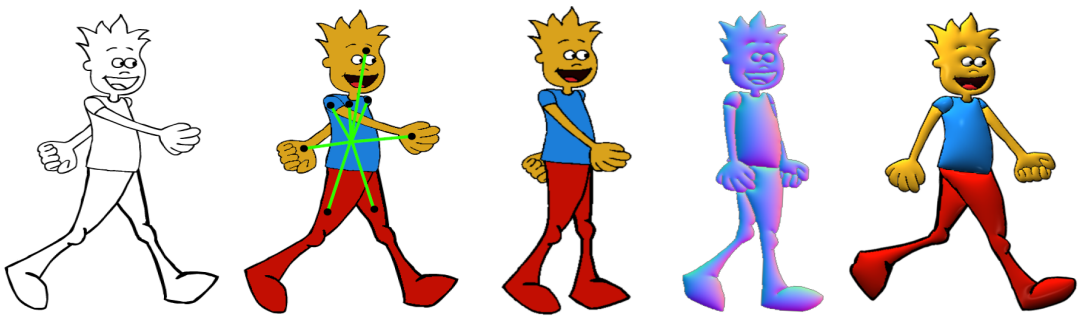

Fig. 1. Colorization and illumination process. From left: original animation frame; graph representation; automatic colorization; normal mapping and shaded animation frame.

***Abstract*—Colorization and illumination are key processes for creating animated cartoons. Computer assisted methods have been incorporated in animation/illustration systems to reduce the artists' effort. This paper presents a new strategy for illumination and colorization of 2D drawings based on a region-tree. Starting from a hand-drawn cartoon, it extracts geometric and topological information and builds a tree structure, ensuring independence among parts of the drawing, such as curves and regions. Based on this structure and its attributes, a colorization method that propagates through consecutive frames of animation is proposed, together with an interpolation method that accurately computes a normal mapping for the illumination process. Different operators on curve and region attributes can be applied independently, obtaining different rendering effects.**

***Keywords*-2D drawing; region-tree; illumination; colorization; animation;**

I. Introduction

Since the 1960's, the computer became part of the tools used in animated films production [1] and it is currently one of the major area of research in computer graphics: Computer Animation.

Conventional animation [2] is based on a frame-by-frame technique representation. Even today, this technique is used in the production of 2D animated cartoons, in which each frame is represented by a free-hand sketch.

In the conventional process of computer assisted cartoon animation, the limitations of the automated solutions and its biggest challenges, such as inbetweening and coloring the frames, were appointed by Catmull [3].

Two important, although tedious, steps in the production of animated cartoons are colorization and illumination. The colorization process transfers the colors from a single frame to the subsequent frames. The most common colorization approaches [4], [5] use structures containing topological information of the drawing, like regions, curves, and graphs, called topological structures.

Although these topological structures work well in the colorization process, they are not commonly exploited in the illumination process of the animation. The illumination process calculates the interaction of the cartoon with the lights present in a 3D environment, and is an important task in the cartoon rendering, since it helps to produce different effects and styles in the animation sequence.

Contributions: This paper presents three main contributions. The first one is a new region-tree structure that explores local spacial information in a single frame. The second contribution is a method based on the region-tree to illuminate the objects, by approximating lighting on 2D drawings using a direct and sphere-preserving interpolation. Third, we propose a recursive tracking method for color transfer based on the region-tree. This new approach improves more effective associations of regions of two consecutive frames that can be retrieved through a recursive analysis of previous frames.

A. Related Work

Regions Representation: An efficient image region representation has been sought in different ways. A common

approach is the use of an hierarchical structure which clusters regions at different scales to obtain the representation of images in various resolutions [6], [7], [8]. These techniques are ideal in processes such as image compression. However, in images like cartoons this structure could be simplified, since only inherent relationship of regions such as inclusion and adjacency relations will be necessary during the cartoon illumination and colorization.

Illumination: In some works [9], [10], automatic normal estimation methods is computed directly from the image. But these techniques calculate the normals from the shading information of the images. They are not suitable for cartoon, because in our input cartoon we have neither normal nor shading. Some techniques to illuminate 2D data provide iterative tools to reconstruct 3D models from available bi-dimensional data [11], [12], followed by traditional 3D illumination [13]. Johnston [14], observes that, even though these methods provide useful results, they may not be suitable to cartoon rendering. Therefore, the author [14] proposes a semi-automatic image-based technique to approximate surface normals directly in 2D drawings while avoiding 3D reconstruction. However, this technique does not enable different kinds of interpolation since it does not have a relationship between curves and regions. Besides it obtains an approximated normal propagation to calculate the normal map. Recently, depth-propagation proposed by Sýcora *et al.* [15] helps to obtain correct initial normal map of cartoons, but the interpolation itself is computed later, by using the method proposed by Johnston [14]. In our work we explicitly calculate the normal vector accurately and guarantees smoothness for a coherent illumination. The region-tree flexibility allows different visual effects since it is possible to choose which curve contributes in the regions interpolation process.

Colorization: Segmentation techniques are generally used to extract topological structures, and employ tracking operations to check the time coherence between consecutive frames, enabling a computer-assisted colorization [4], [16], [17], [18]. Although these techniques provide the colorization to animating cartoons, these approaches do not exploit essential information such as curves and interior regions for the illumination process. Garcia *et al.* [19] propose a strategy for region matching which deals with the curve closeness issue. Since the matching is rather geometric, the method robustness could be affected. Sýcora *et al.* [15] proposes a depth-propagation to handle cartoon animations resulting in automatic colorization. However, this approach does not handle occlusion during tracking. In our work, we propose a topological approach as a way to increase the robustness of region tracking enabling it to handle occlusions.

The remainder of the paper is structured as follows. Section II formulates our proposed structure, the region-tree, that represents cartoon geometric and topological elements. Section III describes our proposed illumination method. Section IV presents a recursive colorization method based on the region-tree. Section V brings some experimental results and Section VI concludes.

II. Region-Tree

This section describes a topological structure, which contains the topological and geometric information of cartoons objects. Similar to several other methods, we first apply a segmentation stage to extract a set of curves and regions [4]. In a later stage, we explore the relationship between regions and curves to construct the data structure *region-tree*.

A. Curves and Regions

Essentially, the traces or outlines of the cartoon define the curves, while the interior of closed curves define the regions. We use the operator sets described by Bezerra *et al.* [4] to construct our topological structure. Those operations includes: *skeletonization* and *region detection*.

Skeletonization or *thinning* is a linear subset of images lines which emphasizes geometrical and topological properties of the shape. The skeletonization algorithm removes redundant points by eroding the image. The final points constitute the skeleton. The main idea is simplify the objects representation without changing its topology. In this work, the Zhang-Suen algorithm was used [20].

Region detection consists on performing the segmentation of the thinning image by identifying each of its curves and regions. Each curve is define by a sequence of pixels from the skeleton. Each region is defined by a boundary (external curve), a set of internal curves and by the set of inside pixels (interior). We first compute the interior of the region by assigning a label to each region by flood-filling the region from a interior point [21].

We need to structure the set of pixels that belongs to the curves. For this, we induce a polygonal representation for the boundary and the internal curves. Although the internal curves are not necessary for a region definition, they are significant part of the drawing, since they provide a variety of details. To obtain a polygonal representation for the curves in each region, we applied the chain-code algorithm [22] with 8-connected neighborhood. After this stage, each region contains: an internal area (set of labeled pixels), a boundary, and a set of internal curves with a polygonal representation (see Fig. 2).

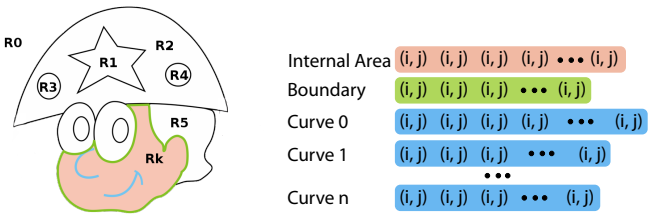

Fig. 2. This figure shows the region structure proposed in this paper. Each region R_k is composed by: a set of internal pixels, labeled by *Internal Area*; a set of boundary points, labeled by *Boundary* and a set of internal curves, labeled by *Curve 0* to *Curve n*.

B. Region-tree Generation

We now describe our proposed region-tree.

Usual topological structures are used in colorization methods to track regions of consecutive frames in the animation process [4], [16], [17], [18]. This structure explores the relationships of adjacency between regions to preserve the temporal coherence. However, they do not explore some local information details in a single frame, such as inclusion relation between regions inside the drawing. The structure proposed in this paper, includes inclusion relation between regions and information from internal curves to the region. This information assists the implementation of new effects in the drawings. We describe in Section III, the use of this structure in the illumination process.

The structure proposed is defined as a tree, where each node contains a region R_i and a set of descendant nodes, representing the internal regions (subsets). Besides being a simple implementation structure, it contains more information than usual topological structures (see Fig. 3), such as the ones described by Sýcora *et al.* [15].

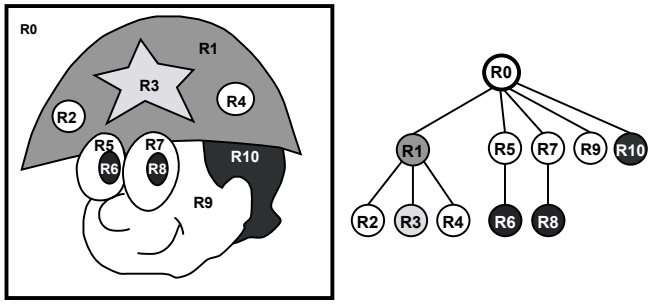

Fig. 3. This figure shows the region-tree structure. Each node represents one region in the image. In this example, the node R_3 is a child node of R_1. So the region R_3 is an internal region of R_1. All regions have the same structured shown in Fig. 2.

The region-tree proposed provides total control in the access of each region: boundary, internal curves, internal pixels, adjacent regions and contained regions. This enables separate attributes for each cartoon region: material properties, color, transparency, texture, and others. Analogously, we can define an attributes to each curve, or group of curves: color, line style, width, and others. Regions attributes can be stored as frame buffers allowing the data to be processed directly on the GPU.

In the context of this work, normals vectors are the most important attribute related to the region or the curves. The next section describes this attribute in detail.

III. Illumination and Normal Mapping

We propose a novel formulation for the normal field inside each region. This normal field is used in a simple Phong model to illuminate the cartoon with 3D impressions. Our normal definition is accurate and sphere-preserving, which leads to smooth and coherent illumination.

For each region, we first compute the normal field of the boundary [4]: given two consecutive points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, we compute the tangent vector $v = (x_2 - x_1, y_2 - y_1)$, and the normal vector at p is given by $n = (y_2 - y_1, x_1 - x_2, 0)$. The coordinate $z = 0$ indicates that the image plane is the projection plane relative to the viewer and the curve belongs to the object's silhouette. Actually, as there are only eight possible directions for the tangent vectors on an 8-connected neighborhood, a smoothing process is applied beforehand. The new tangent vector v_i' is found by the update rule $v_i' = (v_{i-2} + 4v_{i-1} + 6v_i + 4v_{i+1} + v_{i-1})/16$.

The interpolation of the normal field from the boundary to the enclosed region is best described in the continuous domain. Let C be a closed curve parameterized by arc-length, $C(s) = (x(s), y(s))$, and let R be a region delimited by C. The normal vector $n(s)$ at each point of $C(s)$ is given by

$$n(s) = (n_x(s), n_y(s), 0) = (y(s), -x(s), 0).$$

Our task is to compute a 3D normal $n(p) = (n_x(p), n_y(p), n_z(p))$, for each point $p = (x, y, z)$ in R. To compute the components x and y, we integrate the normal's contribution along the curve, using an inverse square distance weight. From the x and y components, we attribute a positive value for the z-component which normalizes the normal vector giving a "lifting" effect to the region.

$$n_x(p) = \frac{\int_C \frac{n_x(s)ds}{|p - C(s)|^2}}{w(p)},$$

$$n_y(p) = \frac{\int_C \frac{n_y(s)ds}{|p - C(s)|^2}}{w(p)},$$

$$n_z(p) = \sqrt{1 - n_x(p)^2 - n_y(p)^2},$$

where

$$w(p) = \int_C \frac{ds}{|p - C(s)|^2}.$$

We now demonstrate that if C is a circle, our interpolation technique provides exactly the normal field of a sphere. In the case that C is a circle, $C(s) = (\cos(s), \sin(s))$. We considered just the case when $p = (x, 0)$, with $-1 < x < 1$, because otherwise we can simply rotate the coordinate system. Applying the symmetry once more, $n_y(p)$ must be zero, since the contribution of $C(s)$ is canceled by the contribution of $C(-s)$. After performing the integration, we conclude:

$$w(p) = \int_0^{2\pi} \frac{ds}{(x - \cos(s))^2 + \sin^2(s)} = \frac{2\pi}{1 - x^2},$$

and

$$n_x(p) = \int_0^{2\pi} \frac{\cos(s)ds}{(x - \cos(s))^2 + \sin^2(s)} = \frac{2\pi x}{1 - x^2}.$$

As $n_y(p) = 0$ by symmetry, $n_z(p) = \sqrt{1 - x^2}$, therefore, the normal field is exactly the same of a sphere of radius 1 centered at the origin.

Unlike Johnston [14], where an approximated normal propagation is obtained to calculate the normal map, our explicit formulation is accurate. Moreover, due to the flexibility of the proposed structure, it is possible to choose which curve contributes in the region interpolation process. This makes it possible to obtain effects as shown in Fig. 4.

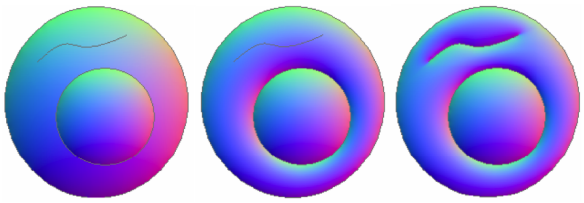

Fig. 4. Interpolation obtained from different curves. In the left one both the interior region and the interior curve does not affect the exterior region normal map. In the middle one, the interior region affects the construction of the exterior region normal map, but the interior curve does not affect. In the right one both the interior region and the internal curve affect the normal map construction.

A. Normal Operators

The proposed structure is also suitable for applying attributes operators to the regions and curves. The user first selects the region or curve, then selects the operator to the desired attribute. Two normal operators were implemented: a *scale operator* and a *depth operator*.

1) Scale Operator: Given a region R, the operator consists in scaling the x, y, or z normal coordinate. If the scale is applied on the z coordinate, the visual effect is to lift the curve when illumination is applied. Fig. 5 shows some z scales applied to normal mapping and level of scale is chosen by the user.

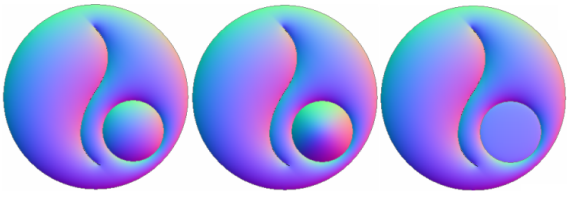

Fig. 5. Scale operator. This operator can be applied only in regions and it is used to scale the x, y and z coordinate. In this figure the operator is applied to the z coordinate. In the left image the normal map is the same obtained by Section III; in the middle one the smaller region was raised and in the left one the smaller region was pushed down.

2) Depth Operator: This operator deals with the idea of rising/sinking the selected curve c located in a region. The user controls the affected area surrounding the curve c and may rise or sink this curve. To do so, we calculate the distance information of each pixel located in a tubular neighborhood of c whose radius is the maximum distance d_{max} chosen by the user. All pixels located in the tubular region will have their normal affected by the depth operator. For each pixel p_i, we obtain its distance $d(p_i)$ to the curve and recalculate its normal $\overline{n}(p_i)$ using the following equation:

$$\overline{n}(p_i) = \left(1 - \varphi\left(\frac{d(p_i)}{d_{max}}\right)\right)(k - n(p_i)) + \left(n(p_i)\varphi\left(\frac{d(p_i)}{d_{max}}\right)\right).$$

Where, $\overline{n}(p_i)$ is the new normal vector of the point p_i, $n(p_i)$ is the normal vector of the point p_i, $k = (0, 0, 1)$ and φ is a function where $\varphi(0) = 0$ and $\varphi(1) = 1$. We used $\varphi(x) = \dfrac{-cos(\pi x) + 1}{2}$.

Fig. 6 shows the depth operator applied to a sphere.

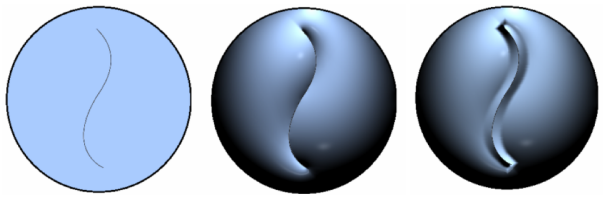

Fig. 6. Depth operator. This figure shows the depth operator applied on curve inside a sphere. The left one is the input image; the middle one is the illumination process from the normal map obtained by Section III and in the right one the curve was raised by the operator. The area surrounding the curve is also affected to create the visual effect.

IV. Automatic Colorization Across a Frame Sequence

As mentioned before, one of the most laborious steps of a cartoon animation is the process of manually coloring its sequence frames.

We will present an automatic recursive colorization technique based on the use of the region-tree representation to each animation frame. This process allows the color information to be propagated automatically from a single frame to the other frames in the same scene of the animation.

A. Regions Tracking

Tracking is used to obtain the best associations between regions in consecutive frames. In general, those frames present small variations throughout the scene. Therefore, the associations can be performed by analyzing the parameters of position, area, shape, and topology [4].

In a given frame, the local area form A associated to each point p_i in the border curve of a region is calculated by

$$A = \frac{1}{2}\sum_{i=0}^{n-1}(x_i \cdot y_{i+1} - x_{i+1} \cdot y_i).$$

The region position is represented by its centroid $C = (c_x, c_y)$, where

$$c_x = \frac{1}{6A}\sum_{i=0}^{n-1}(x_i + x_{i+1})(x_i \cdot y_{i+1} - x_{i+1} \cdot y_i),$$

$$c_y = \frac{1}{6A}\sum_{i=0}^{n-1}(y_i + y_{i+1})(x_i \cdot y_{i+1} - x_{i+1} \cdot y_i).$$

Given a region, we define its contours as the number of points (pixels), which compose its boundary. In Fig. 7 (left) shows a way to represent the topology of the images with its adjacency graph constructed from a region-tree: yellow edges represent subset relationships and red ones represent regions sharing the same boundary. Let R be a region and c_R its centroid. The neighborhood relationship between R and its neighbor Q with boundary points p_{Qj}, $1 \leq j \leq m$, is given by a neighborhood function nF_{RQ}:

$$nF_{RQ} := (\alpha_{RQ}, d_{RQ}),$$

where

$$d_{RQ} = \max_{1 \leq j \leq m} \{c_R, dist(p_{Qj})\},$$

and α_{RQ} is the slope of the line from c_R to

$$\arg\max_{1 \leq j \leq m} c_R, dist(p_{Qj}).$$

Each image region R will have n neighborhood functions, since they have n neighbors (see Fig. 7 (right)).

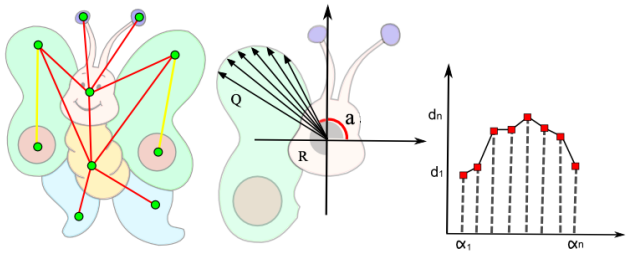

Fig. 7. This figure shows the adjacency graph (left); the scheme to obtain the neighborhood function for a region R, where $\alpha = \tan(a)$ (middle) and all neighborhood function of R (right).

B. Associations Between Consecutive Frames

The simplest way to create associations between regions from two consecutive frames could be comparing all the vertices from both frames graph. However, this option is also computationally expensive. To solve this problem, we propose the use of a region-tree in order to reduce the cost.

We execute a breadth-first search in the source-graph and destiny-graph initialized from the image background, since this is the only region of occurrence guaranteed in all images. In each search iteration, we analyze parameters of the area, position, contours, and neighborhood between regions represented by the compared vertices. The neighborhood parameter is defined by the *equivalence degree* (ED) from two regions. As shown in Fig. 8, the ED of two regions from consecutive frames is the sum of two smaller quadratic difference between its neighborhood function.

If the number of regions in the destiny-graph, in the current iteration, is bigger than in the source-graph, these regions

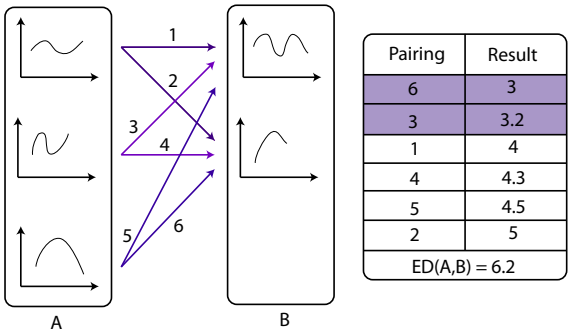

Fig. 8. Equivalence graph from two regions. In this example, the equivalence degree ED of A and B is obtained by the sum of two smaller quadratic difference between its neighborhood function. So, ED(A,B) = 3+3.2 = 6.2.

are taken as new regions in the animation and they are not colored Fig. 9. Otherwise, the region color in the source-graph is discarded along the animation process.

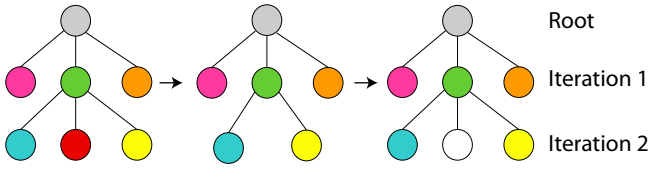

Fig. 9. Automatic colorization process. In the Iteration 2, the red region disappeared in the second graph, *i.e.*, the region was occluded. Therefore, this region color is discarded along the animation.

C. Recursive Regions Tracking - Recovering Occluded Regions

If the region disappears in a frame, and shows up in another one, we have an occluded region case in the sequence of frames, so this kind of tracking will be unsatisfying in the colorization process. Our work proposes tracking in a recursive way to improve the recovering of these occluded regions in the process.

In recursive regions with tracking, whenever a color information is passed from a region *A* to a region *B* in the consecutive frame, the vertice relative to the region *A* is marked. Once the frame is colorized, the algorithm verifies if there is any region not colorized in the frame. If all regions are colorized, the process will continue making a better association between frames as described in Section IV-B.

In the case where there is no colorized regions, the process is temporarily interrupted. Through the image adjacency graph, we find out which are the *n* neighboring regions of the non-colored. In this case do a reverse search through previous frames. In each analyzed frame, we look for regions corresponding to the n regions we identified. Once the corresponding regions are found, we check if they are unmarked neighboring regions. If this occurs, the unmarked regions are stored in a list of possible candidates to be equivalent to the

non-colorized region. These candidates are collected in each frame.

After the search, the algorithm goes back to the frame where the process stopped. All the candidates are analyzed by the same parameters: area, contour, position, and equivalence degree. An association between the best candidate and the non-colorized region is created passing all color information. If two or more candidates are selected, no association is created and the region is not colorized. Fig. 10 illustrates the recursive colorization process, where the color of the white region is recovered from the first frame and then is passed to consecutive frames.

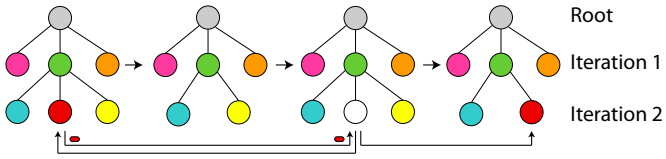

Fig. 10. Automatic recursive colorization process. In the recursive colorization process if a region disappeared, *i.e.*, the number of regions in the destiny-graph is bigger than in the source-graph, a reverse search through previous frames is made.

V. Results and Discussion

This section presents some results of our automatic colorization and illumination technique. Fig. 11 shows the illumination process from a black and white cartoon, and Fig. 12 shows the illumination process from a colorized cartoon. Fig. 13 shows frames from an animation after the colorization and illumination process.

Figs. 15 and 16 show some comparisons between our recursive colorization method based on a region-tree representation and the colorization method proposed by Bezerra *et al.* [4]. In both cases, we can observe that occluded regions are recovered by the recursive tracking when they reappear in the animation sequence, unlike Bezerra *et al.* [4] method. In Fig. 15, the character's face, which previously was a single region (green area) is split into two main regions. These regions are new in the frame so the algorithm does not recognize them. In the recursive tracking, when the character's face comes back to being one region representation the algorithm recovers its color.

Fig.16 shows the occlusion problem in the bear's left leg. In the first image sequence, the color information can not be recover since this information is lost in the fifth image because there was a significant topological change such as a region splitting into two. Secondly regions having its neighborhood drastically altered such as the left eye, the hats brim, and the tie are lost in Bezerra *et al.* [4] tracking. In the case of the eye, it occurs when the eye links to the bear's face region. In the hats brim case, it occurs when the brim is linked to a new region - the bear's ear. Finally, in the tie case, it occurs because one region disappears joining the bear's arm.

A. Limitations

A limitation of normal vector calculation is that we can not change the boundary normal vector. So, if a point is a boundary point, the z-component is always 0. This may affect the visual effect the process of illumination.

In the colorization tracking, a problem could occurs in situations where the regions are similar as in walk motion of a character. The two legs of the character are topologically identical, they have identical area, position and neighborhood. Then, in the motion, when the left leg surpassing the right one, the algorithm swaps the colors of both, because the position of the left leg is replaced by similar topological information - right leg. Fig. 14 shows this problem.

Another limitation is once the region track is done, only color information is propagated. So, in an animation sequence we need to illuminate frame by frame.

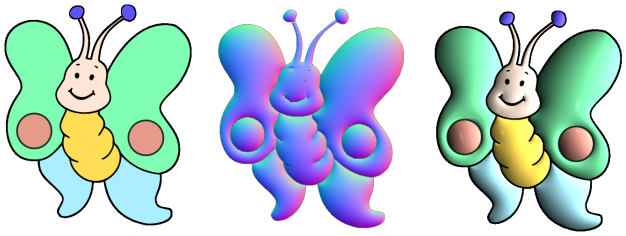

Fig. 12. Illumination process. From left: original image; normal map; illuminated cartoon.

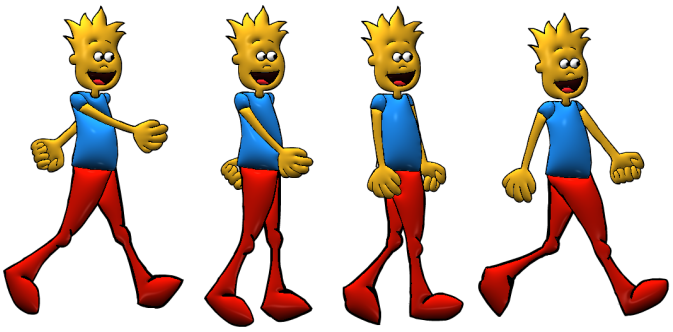

Fig. 13. Colorized and illuminated animation. In this result, we choose the color of the first frame than we apply the automatic colorization and finally, the cartoon is illuminated.

VI. Conclusion and Next Steps

This work presented a new strategy based on a region-tree structure to represent 2D drawings. Unlike usual colorization methods, our method explores both the time coherence of the topological structures, as well as the local spatial information of each frame. Therefore our method is both suitable for colorization and to illumination.

Due to the topological and geometrical information of this region-tree representation, we developed a method for approximating lighting to 2D drawings: we compute the normal

Fig. 11. Illumination process from a black and white cartoon. From left: original image; colorized cartoon; segmented cartoon; normal map; illuminated cartoon.

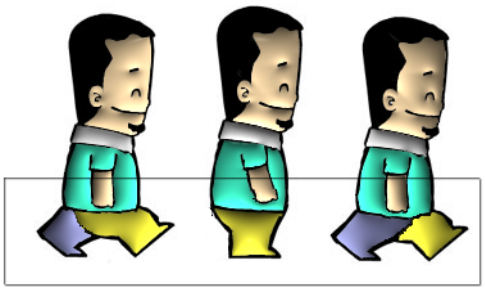

Fig. 14. Problem with the tracking in walk motion: The leg's colors are swapped since they have similar topological information.

mapping of the drawing based on a new topological structure. As we explore both the labeling structure of each region, as well as the vectorial structure of the boundaries, our method is especially suited for independently shading parts of the drawing, such as regions and curves.

With the recursive tracking method, we can recover the most of the lost regions in the animation. For example, in the movement of a character walking, where at times, one of the legs is occluded, the use of this method becomes more feasible, to obtain better results.

As future works we pretend to verify time coherence together with local space structure and the transference of the illumination's information during the normal map tracking as way to illuminate a cartoon sequence automatically. We also pretend to define new attribute operators and explore normal visibility in the interpolation process to avoid that invisible parts of the region contributes to the normal vector of a point.

ACKNOWLEDGMENT

The authors would like to thank L. Velho for encouraging this work's production. The authors also would like to thank J. Paixão for the text's revision.

REFERENCES

[1] G. M. Hunter, "Computer animation survey," *Computers and Graphics*, vol. 2, pp. 225–229, 1977.

[2] J. Lasseter, "Principles of traditional animation applied to 3d computer animation," pp. 263–272, 1998.

[3] E. Catmull, "The problems of computer-assisted animation," *Siggraph*, vol. 12, no. 3, pp. 348–353, 1978.

[4] H. Bezerra, L. Velho, and B. Feijó, "A computer-assited colorization algorithm based on topological differences." in *Sibgrapi*, 2006, pp. 121–127.

[5] C.-W. Chang and S.-Y. Lee, "Automatic cel painting in computer-assisted cartoon production using similarity recognition," in *The Journal of Visualization and Computer Animation*, 1998.

[6] G. Salembier, L. Garrido, P. Salembier, and J. R. Casas, "Representing and retrieving regions using binary partition trees," in *International Conference on Image Processing*, 1999.

[7] M. A. de Carvalho and R. D. A. Lotufo, "Hierarchical regions matching in image sequences through association graph," in *Sibgrapi*, 2001, pp. 396–404.

[8] M. A. Carvalho, M. Couprie, and R. de Alencar Lotufo, "Image segmentation by analysis of scale-space," in *Sibgrapi*, 2002, pp. 403–411.

[9] T.-P. Wu, J. Sun, C.-K. Tang, and H.-Y. Shum, "Interactive normal reconstruction from a single image," *Siggraph Asia*, vol. 27, pp. 119:1–119:9, 2008.

[10] B. Horn, "Obtaining shape from shading information," pp. 123–171, 1989.

[11] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: a sketching interface for 3d freeform design," in *Siggraph courses*, 2007.

[12] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes, "Sketch: an interface for sketching 3d scenes," in *Siggraph courses*, 2007.

[13] B. T. Phong, "Illumination for computer generated pictures," *Communications of the ACM*, vol. 18, pp. 311–317, 1975.

[14] S. F. Johnston, "Lumo: illumination for cel animation," in *Non-Photorealistic Animation and Rendering*. ACM, 2002, pp. 45–ff.

[15] D. Sýkora, D. Sedlacek, S. Jinchao, J. Dingliana, and S. Collins, "Adding depth to cartoons using sparse depth (in)equalities," *Computer Graphics Forum*, vol. 29, no. 2, pp. 615–623, 2010.

[16] S. Hock and F. Tian, "Computer-assisted coloring by matching line drawings," *The Visual Computer*, vol. 16, no. 5, pp. 289–304, 2000.

[17] D. Sykora, J. Burianek, and J. Zara, "Segmentation of black and white cartoons." *Spring Conference on Computer Graphics*, pp. 245–254, 2003.

[18] D. Sýkora, J. Burianek, and J. Zara, "Unsupervized colorization of black-and-white cartoons." *Non-Photorealistic Animation and Rendering*, pp. 121–127, 2004.

[19] P. G. Trigo, H. Johan, T. Imagire, and T. Nishita, "Interactive region matching for 2d animation coloring based on feature's variation," *Ieice Transactions*, vol. 92-D, pp. 1289–1295, 2009.

[20] Zhang and Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, pp. 236–239, 1984.

[21] L. Vandevenne. (2004) Lode's computer graphics tutorial flood fill. [Online]. Available: http://lodev.org/cgtutor/floodfill.html

[22] J. Parker, *Pratical Computer Vision Using C*. New York: John Wiley and Sons, 1994.

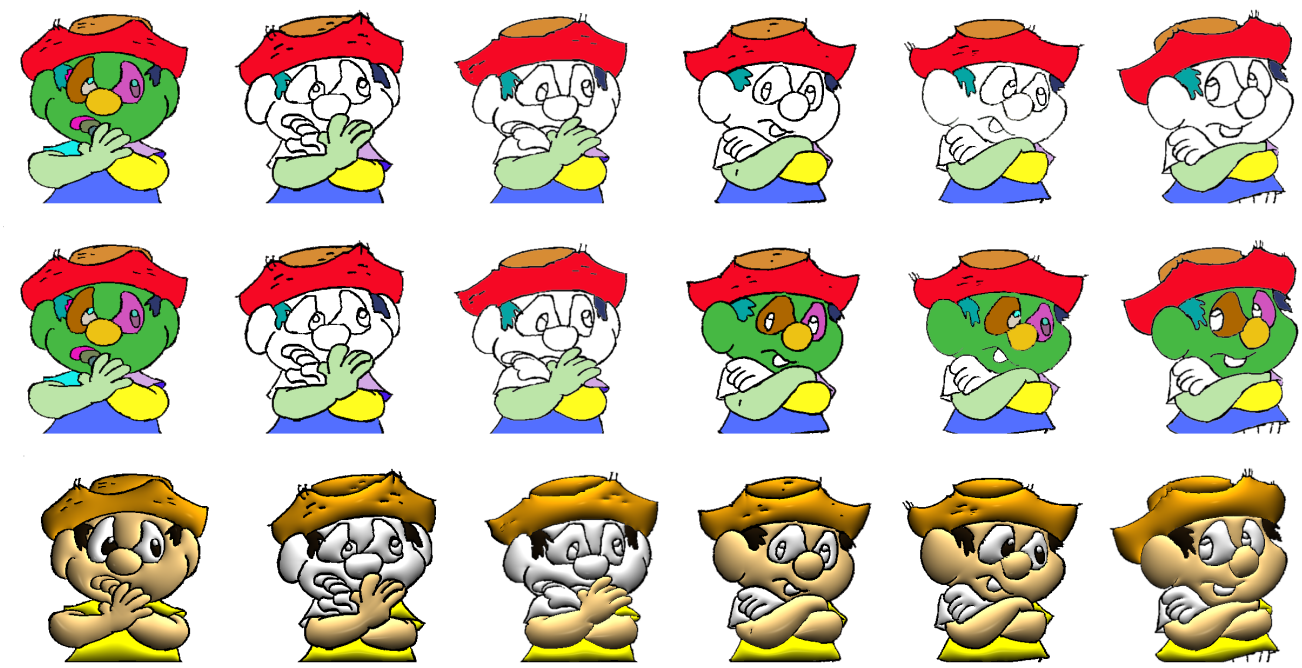

Fig. 15. First sequence: Bezerra *et al.* [4] colorization tracking. Second sequence: Our recursive colorization tracking based on a region-tree representation. Third sequence: Illumination process based on a region-tree representation. In the first sequence the character's face, which previously was a single region (green area) is split into two main regions. These regions are new in the frame so the algorithm does not recognize them. In the recursive tracking, when the character's face comes back to being one region representation the algorithm recovers its color.

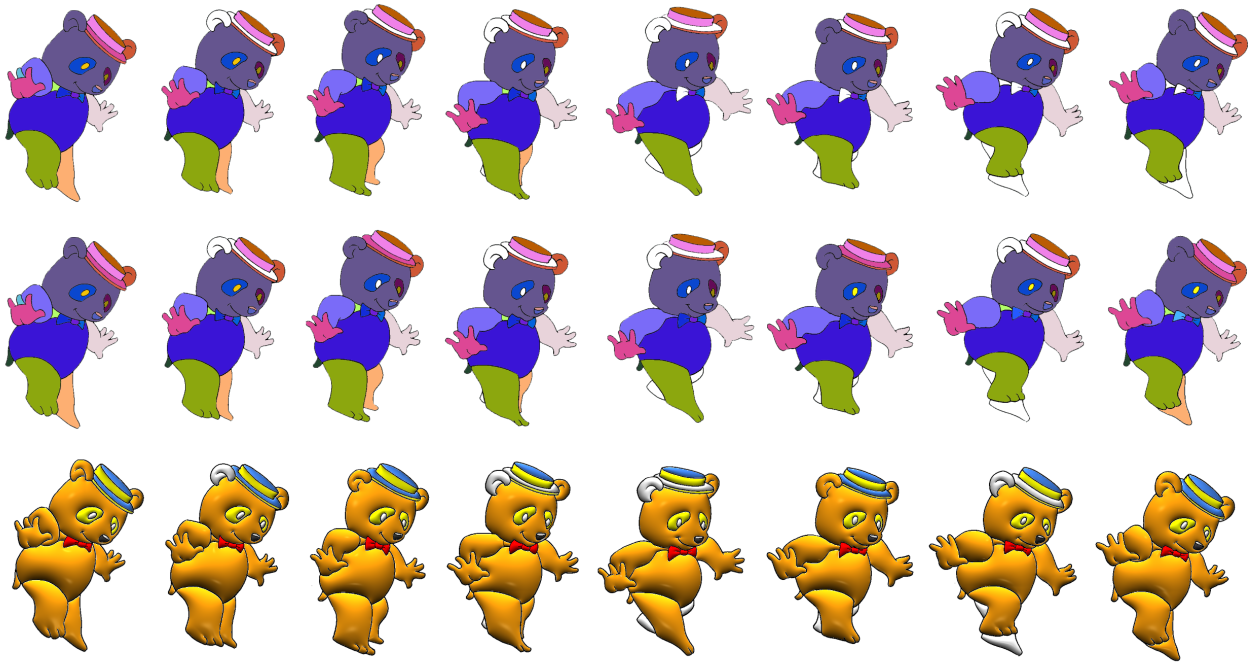

Fig. 16. First sequence: Bezerra *et al.* [4] colorization tracking. Second sequence: Our recursive colorization tracking based on a region-tree representation. Third sequence: Illumination process based on a region-tree representation. This result shows the occlusion problem. In the bear's left leg, at the first image sequence, the color information can not be recover since this information is lost in the fifth image because there was a significant topological change such as a region splitting into two. Secondly regions having its neighborhood drastically altered such as the left eye, the hats brim, and the tie are lost in Bezerra *et al.* [4] tracking. In the case of the eye, it occurs when the eye links to the bear's face region. In the hats brim case, it occurs when the brim is linked to a new region - the bear's ear. Finally, in the tie case, it occurs because one region disappears joining the bear's arm.