SpringerPlus
a SpringerOpen Journal

RESEARCH Open Access

Computer-assisted coloring and illuminating based on a region-tree structure

Renata Nascimento[1*], Fabiane Queiroz[2], Allan Rocha[3], Tsang Ing Ren[2], Vinicius Mello[5] and Adelailson Peixoto[4]

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' workload. This paper presents a new method for illumination and colorization of 2D drawings based on a region- tree representation. Starting from a hand-drawn cartoon, the proposed method 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, combined with an interpolation method that accurately computes a normal mapping for the illumination process. Different operators for curve and region attributes can be applied independently, obtaining different rendering effects.

Keywords: 2D drawing, Region-tree, Illumination, Colorization, Animation

Introduction

Since the 1960s, computers became a great ally in the production of animated films, contributing to several areas that goes from the creation to the final touches [1].

In the conventional production of an animation sequence, first of all is necessary a representation of frame by frame [2]. Even today, this technique is used in the production of 2D animated cartoons in which each frame is represented by a free-hand sketch.

Although assisted animation accelerate the procedure, it still has many limitations which drives one of the largest areas of research in computer graphics: Computer Animation.

In the conventional process of computer assisted cartoon animation, the entire process for the production is very costly and may even take months for a short animation to be finished. Among the biggest challenges of an automated solution, the production of inbetweening and frame colorization as mentioned by [3] is of great importance.

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 perform 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 light present in a 3D environment, and is an important task in 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

* Correspondence: renata@mat.puc-rio.br
[1]Department of Mathematics, PUC-Rio, Rio de Janeiro, Brazil
Full list of author information is available at the end of the article

regions of two consecutive frames that can be retrieved through a recursive analysis of previous frames.

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 [6-8]. These techniques are ideal in processes such as image compression. However, in images like cartoons this structure could be simplified, since simpler 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. How-ever, 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 Sy´ cora et al. [15] 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 to 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 [4,16-18]. 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 Sy´ cora 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 this 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 2 formulates our proposed structure, the region-tree, that represents cartoon geometrical and topological elements. Section 3 describes the proposed illumination method. Section 4 presents a recursive colorization method based on region-tree. Section 5 presents some experimental results and Section 6 concludes.

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*.

Curves and regions

Essentially, a cartoon could be represented by their curves and regions. 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 the set of regions and curves that will compose our region-tree structure. Those operations includes: *skeletonization* and *region detection*.

Skeletonization or *thinning* is the extraction process of a linear subset of images' lines, which emphasizes geometrical and topological properties of the shape. The main idea is simplify the objects representation without changing its topology. Therefore, the skeletonization algorithm removes redundant points by eroding the image while preserving the lines connection and hence the topology of the drawing (see Figure 1 middle). The final points constitute the skeleton. In this work, the Zhang-Suen algorithm [20] was used.

Region detection consists on performing the segmentation of the thinning image identifying each of its curves and regions. Each curve is defined by a sequence of pixels from the skeleton. In our structure, each region is defined by three informations: its boundary (external curve), its interior points (the region itself), and its internal curves. This structure will help us to calculate more efficiently the relations of inclusion and adjacency between regions in our region-tree. To extract each region, we first compute the internal area by assigning a label to each connected internal pixel. So, each connected component of the image will have the same label

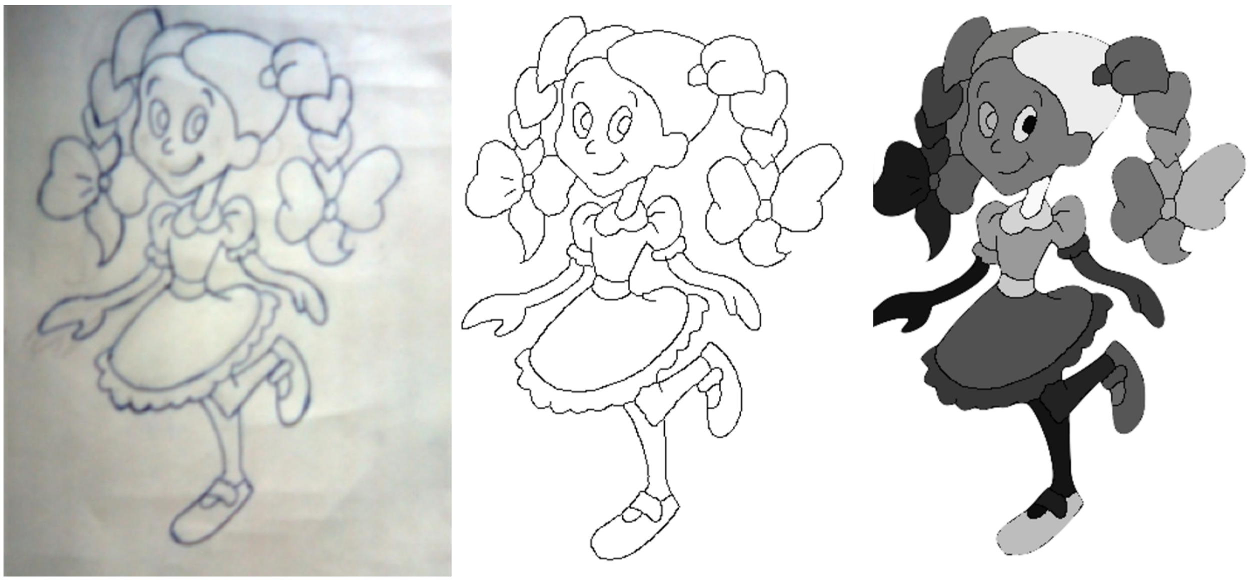


Figure 1 Input hand drawing cartoon (left). Skeletonization(middle). Region detection (right).

and will define the interior of a region. A flood-filling algorithm [21] is performed to make such labeling (see Figure 1 right).

To extract the curves of a region (boundary and internal curves) we will sort the skeleton pixels, introducing a polygonal representation to the curves. Although the internal curves are not necessary for a region definition, they have a great value in the representation since they provide a variety of details giving expressivity to the drawing. A polygonal representation for each regions curve is obtained by a chain-code algorithm [22] with 8-connected neighborhood. After this stage, for each region is possible to access immediately its internal area (set of labeled pixels), its boundary curve and the set of internal curves both in a polygonal representation (see Figure 2).

Region-tree generation

Now we describe our proposed region-tree structure. Usual topological structures are used in colorization methods to track regions of consecutive frames in the animation process [4,16-18]. Those structures explore 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. In order not only to provide time relational coherence between frames, the structure proposed in this paper aims to also allow exploring each images component individually. The possibility of navigation between the regions structures and their included regions allows treating the components one by one enabling a wide variety of effects in the drawings.

The structure proposed is defined as a tree, where each node contains a region R_i, and the set of its descendant nodes represents the internal regions (subsets). Our strategy to decide whether a region bounded by a curve C_1 is contained in another region bounded by a curve C_i is based on the theorem of Jordan [23]. Thus,

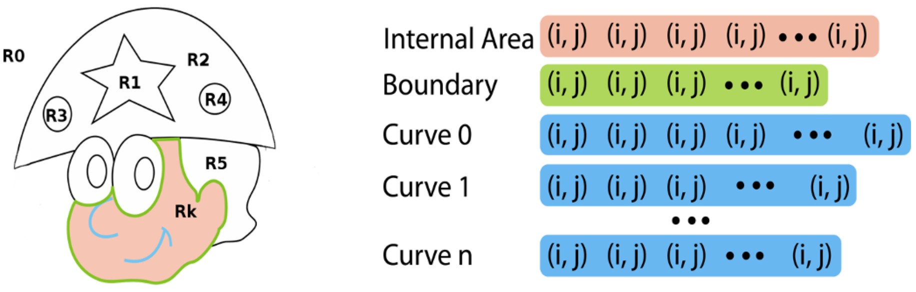

R0
R1
R2
R3
R4
R5
Rk
Internal Area (i, j) (i, j) (i, j) (i, j) ••• (i, j)
Boundary (i, j) (i, j) (i, j) ••• (i, j)
Curve 0 (i, j) (i, j) (i, j) (i, j) ••• (i, j)
Curve 1 (i, j) (i, j) (i, j) ••• (i, j)
•••
Curve n (i, j) (i, j) (i, j) ••• (i, j)

Figure 2 Region Structure: 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*.

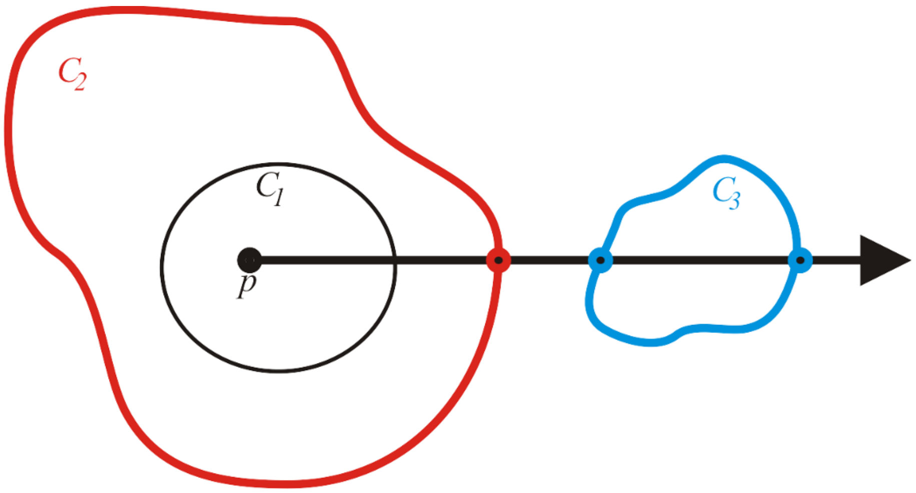


Figure 3 Since the ray casted from *p* intersect C_2 one time, C_1 is contained in C_2. On the other hand, the ray intersect C_3 an even number of times, so C_1 is not contained inC_3.

given a point p inside the curve C_1, we cast a ray from p and find the number n of intersections of the ray with the curve C_i. If n is odd, then C_1 is contained in C_i (see Figure 3).

Besides being a simple implementation structure, it contains more information than usual topological structures (see Figure 4), such as the ones described by Sy´cora et al. [15].

In Section 3, we present how the region-tree structure assists in the illumination process of a 2D drawing. Its advantage stems from the possibility of dealing individually, and in a consistent way, each images region allowing also process them in parallel.

The region-tree structure provides total control in the access of each regions component as 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 attribute 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.

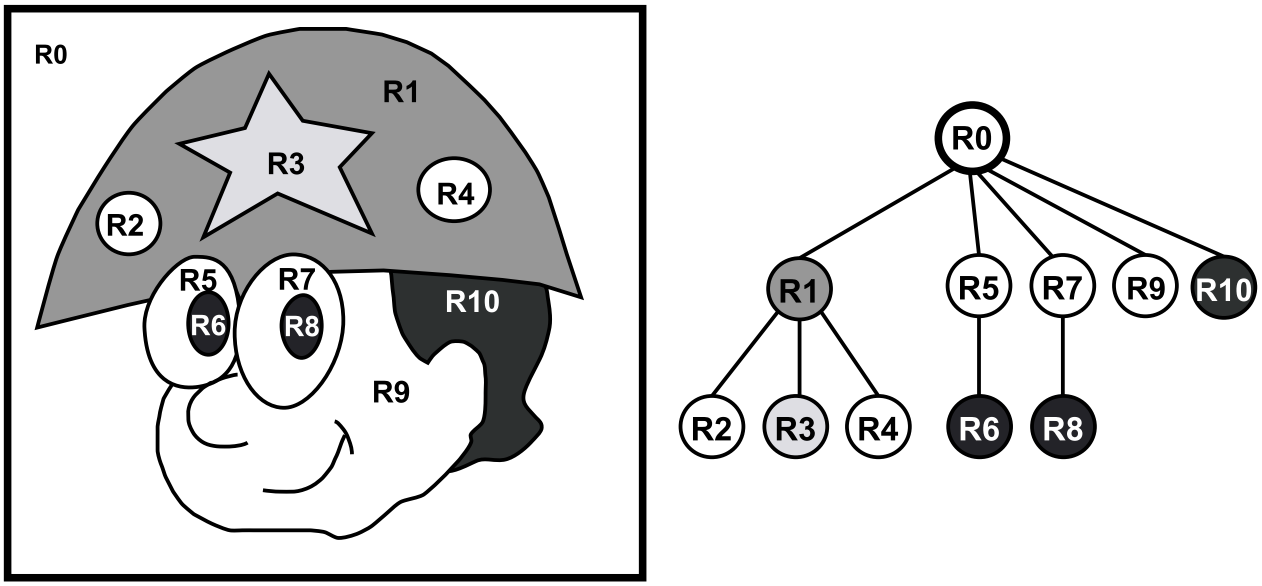


Figure 4 Region-Tree Structure: Each node represents one region in the image. In this example, the node *R*3 is a child node of *R*1. So the region *R*3 is an internal region of *R*1.

In the context of this work, the normal vectors to the curves and regions perform a role of great significance. The inference of a normal field to the image, make it possible for any illuminations model providing three-dimensional effects on the cartoon. The next section describes this attribute in detail.

Illumination and normal mapping

In this work, we propose a novel formulation for the normal field inside each region. This normal field enables the application of illumination effects, like Phong model, providing 3D impression to the cartoon. Our normal definition is accurate and sphere-preserving, which leads to smooth and coherent illumination.

Normal field of the curves

For each region, we first compute the normal field of the boundary [4]: given two consecutive boundary points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, we compute the tangent vector $v = (x_2 - x_1, y_2 - y_1)$. So, 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.

Since we are working in an 8-connected neighborhood, the normal vector obtained above is restricted to these eight directions. This constraint has as consequence a low quality at the illumination effect. Thus, if a point p_i has normal v_i, a smoothing process is obtained by applying a discrete Gaus-sian filter to the normal vi'. The new normal vector vi' is

$$v_i' = \frac{(v_{i-2} + 4v_{i-1} + 6v + 4v_{i+1} + v_{i+2})}{16},$$

where v_{i-2}, v_{i-1},v_{i+1} and v_{i+2} are the normal vectors of the p_i neighbors. The normal vectors of the internal curves of the region are computed in a similar way.

Normal field of the interior

The normal field of the interior of the region is interpolated from the boundary normal field. Our proposed formulation is 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:

$$n(s) = (nx(s), ny(s), 0) = (y'(s), -x'(s), 0).$$

Our task is to compute a 3D normal

$$n(p) = (nx(p), ny(p), nz(p)),$$

for each point $p = (x, y, z)$ in $\mathbb{R}$.

First of all, we compute the components n_x and n_y. This is done by integrating the normal contributions along the curve C, where the contribution of each point $C(s)$ is weighted by the inverse square distance from p to $C(s)$. So, for each $p \,\lfloor\, R$, $n_x(p)$ and $n_y(p)$ are obtained by:

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

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

where

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

The next step is to calculate the n_z component. This step is similar to [14]. We consider that the normal field we want to reconstruct is over a surface that rises smoothly from the curve *C (s)* contained in the image plane (where $n_z = 0$). This normal field points toward the viewer, so the n_z component is positive. In addition, nz should be calculated so that the normal vector field is normalized. So, the nz component is obtained by:

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

If the region has internal curves, the normal vectors of theses curves may be used to compose the final normal field of the interior of the region during the interpolation process presented above. If these internal curves are not considered during the interpolation process, only the boundary curves will affect the normal field of the region.

Normal field analysis

We now do an analysis of the surface "generated" by our normal interpolation method. Is it smooth? How it behaves over its domain (the image plane)? First of all, we demonstrate that if C is a circle our interpolation technique provides exactly the normal field of a sphere.

Proof In the case that C is a circle, $C(s) = (\cos(s); \sin(s))$. Due to the properties of functions *sine* and *cosine*, we considered just the case when $p = (x, 0)$, with $-1 < x < 1$. For the others values, we can simply rotate the coordinate system or get them by symmetries. 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 $\mathrm{n_y}$ (p) = 0 by the symmetry, $\mathrm{n_z}(p) = \sqrt{1-x^2}$. Therefore, the normal field is exactly the same of a sphere of radius 1 centered at the origin.

When we consider internal curves to interpolate the normals, points on these curves are presented as singularities on the surface. On the other hand, if we consider only the boundary, the surface behaves smoothly over the domain.

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 Figure 5.

Normal operators

The proposed region-tree structure is also suitable for applying attribute operators to the regions and to the curves of the cartoon. As the flexibility of our region-tree allows quick access to topological structures present in the cartoon, we can modify, in an efficient and independent way, the attributes of curves and regions. Some attributes are color, thickness, normals, depth, etc. To demonstrate this flexibility, we implemented two normal operators: *the scale region operator*, and the *depth curve operator*.

Scale operator

Given a region R, the operator consists in scaling the n_x, n_y, or n_z normal coordinate. If the scale is applied on the n_z coordinate, the visual effect is to lift the curve when illumination is applied. Figure 6 shows two different z-scales applied to the normal mapping of the smallest circle.

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, with normal $n(pi)$, we obtain its distance $d(pi)$ to the curve and calculate its new normal n (pi) by adding an increase vector $u(pi)$ to the vector $n(pi)$. The increase vector $u(pi)$ is calculated by some function considering that it converges to (0, 0, K) if p_i is near to the curve, and converges to (0, 0, 0) if vector p_i is away from the curve. We implemented the following equation:

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

Where, $\bar{n}(p_i)$ is the new normal vector of the point p_i, $n(pi)$ is the normal vector of the point p_i, k = (0, 0, 1) and ϕ is a function where $\phi(0) = 0$ and $\phi(1) = 1$. We used

$$\varphi(x) = \frac{-cos(\pi x) + 1}{2}$$

Figure 7 shows the depth operator applied to a sphere.

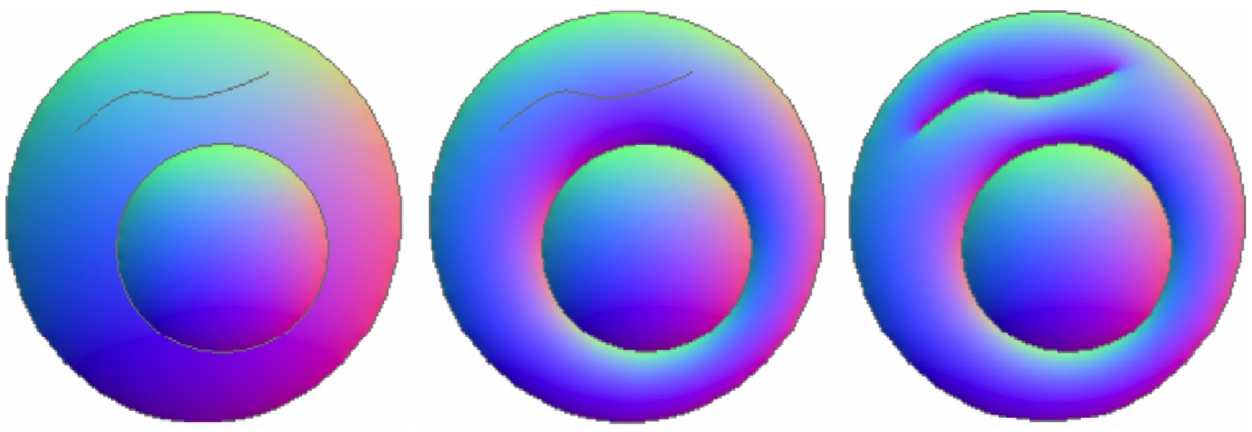


Figure 5 Only the boundary of each region affect the interpolation normals (left). The boundary of the smallest circle also affected the exterior region (middle). Both the boundary of the smallest circle and the internal curve also affected the normal map (right).

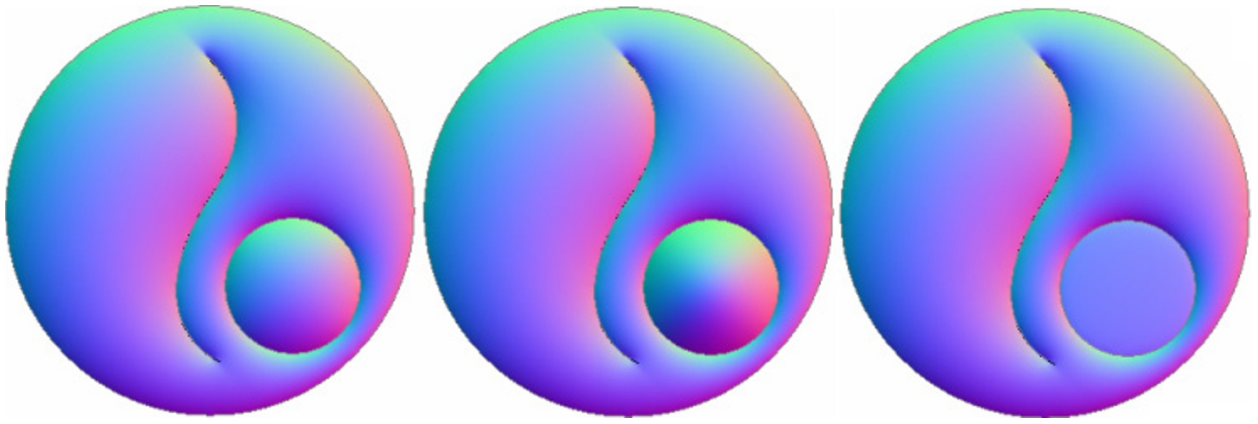


Figure 6 Scale Region Operator: Normalized normal mapping (left). Factor 5.0 *z*-scale applied to the smallest circle (middle). Factor 0.1 *z*-scale applied to the smallest circle (right).

Automatic colorization across a frame sequence

The process of coloring manually each frame in a cartoon animation sequence is one of the most laborious steps. This process requires extensive work by the artist, both due to the large number of frames as the amount of detail in each one of them.

We present an automatic colorization technique based on our region-tree representation. The region-tree of consecutive frames allow the color transfer along the cartoon sequence.

If during the animation the region-tree is changed (regions appear/ disappear) a backtrack is performed to recover the information.

Regions tracking

The first step to enable an automatic colorization is to find out the correspondence from consecutive frames. One way to obtain the best associations is to perform a regions tracking between both region-tree. In general, those frames present small variations throughout the scene, therefore, the association can be performed by analyzing parameters like position, area, shape and topology [4].

Let R_i be a region with p_i, $0 \le i \le n$ boundary points. The local area A associated to each p_i 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

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

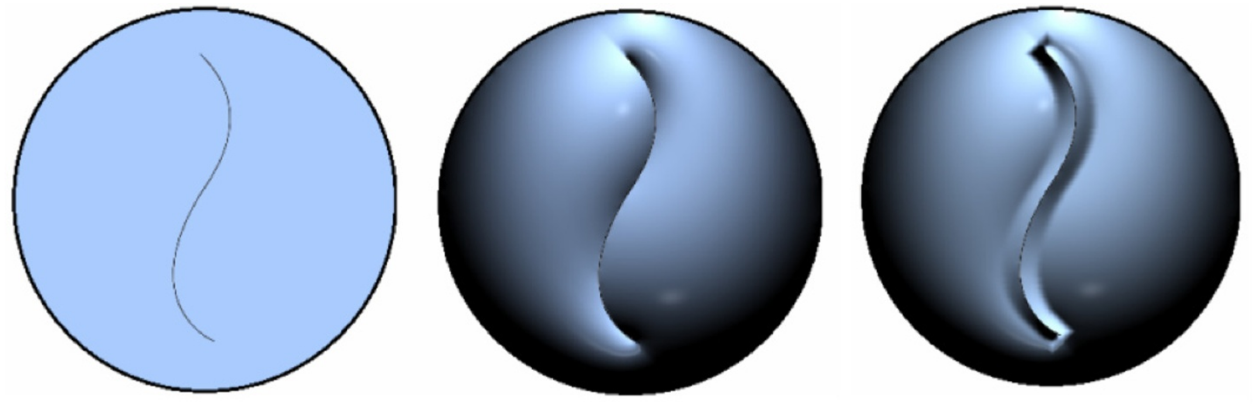


Figure 7 Depth Curve Operator: Initial region (left). Interpolated normal mapping (middle). Depth operator applied to the internal curve (rigth).

and

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

Adjacency graph and neighborhood function

The *adjacency graph* is utilized in this work as way to make the region's tracking in a consistent way.

The graph is extracted from the region-tree by looking for regions that satisfy two conditions: sharing of edges and inclusion relationship. In Figure 8 (left) the adjacency graph is shown, where the yellow edges represent inclusion 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

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

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

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

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

So, if an image's region R has n neighbors, it will have n neighborhood functions (see Figure 8 (right)).

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 the picture bellow, the ED between a region A from frame i and B from frame $i + 1$, where A and B have three and two neighboring regions respectively, is given by the sum of two smallest quadratic difference between its neighborhood function (see Figure 9 right).

If the number of regions in the destiny-graph, in the current iteration, is bigger than in the source-graph, these regions are taken as new regions in the animation and they are not colored Figure 10. Otherwise, the region color in the source-graph is discarded along the animation process.

Recursive regions tracking - recovering occluded regions

In the animation process, there are cases where a region can disappears in a frame and shows up in another one in the sequence. This happens every time that we have occlusion between regions during the animation. For those cases, the regions tracking (Section 4.1) is unsatisfying for the colorization process. Our work proposes tracking in a recursive way to improve the process by recovering the information of these occluded region.

In the recursive regions tracking, whenever a color information is passed from a region A to a region B in the consecutive frame, the vertex 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.

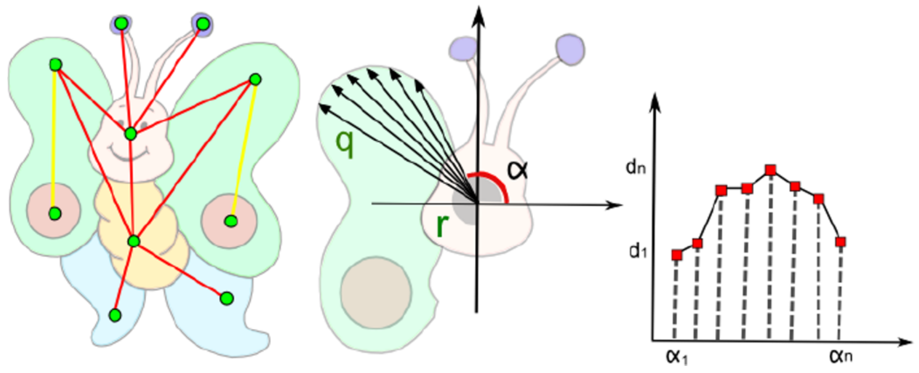
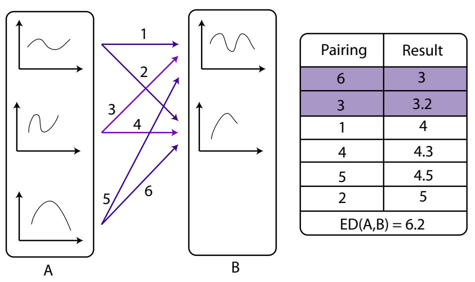


Figure 8 Adjacency graph (left). Scheme to obtain the neighborhood function for a region R, where α = tan(a) (middle). Neighborhood function of R (right).



Pairing	Result
6	3
3	3.2
1	4
4	4.3
5	4.5
2	5
ED(A,B) = 6.2	

Figure 9 Equivalence Graph: The equivalence degree ED of *A* and *B* is obtained by the sum of two smallest quadratic difference between its neighborhood function: ED(A,B) = 3 + 3.2 = 6.2.

If all regions are colorized, the process will continue making a better association between frames as described in Section 4.3.

For the case where there is a not colorized region, the process is temporarily interrupted. Through the images adjacency graph, we find out which are the n neighboring regions of the non-colored one. In this case a reverse search is done 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. Figure 11 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.

Results and discussion

This section presents some results of our automatic colorization and illumination technique. From a free-hand sketch drawing, a cartoon sequence could be colorized and than illuminated creating a 3D effect for the drawing (see Figure 12). In this work, we applied to all results a *phong* illumination [13], although with the normal mapping (Section 3) any illumination effect could be performed.

In Figure 13 is shown the main steps of the illumination process. From a black and white cartoon as input, the user provide the color information that will be used in the illumination. The algorithm perform the image's segmentation and computes the normal map which will be used to create the 3D effect. In Figure 14 the *Recursive Region Tracking* (Section 4.4) is performed to recover the color information that was lost during the animation. In this sequence the model's face (the green area) was split in two different regions changing the adjacency graph structure since ne1w nodes are inserted.

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 allows recover its color by matching its adjacency graph with the first one.

Figures 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 these cases, we can observe that meaningful changes in the adjacency graph resulting from occlusion and/or creation of regions may cause less impact on the sequence if the recursive tracking is performed. Once the color information is lost, methods like Bezerra et al. [4] are not able to recover the informations causing many frames without proper colorization.

Figure 15 shows the occlusion problem in the bear's left leg. In the first image sequence, the color information cannot be recovered since this information is lost in the fifth image because there was a significant

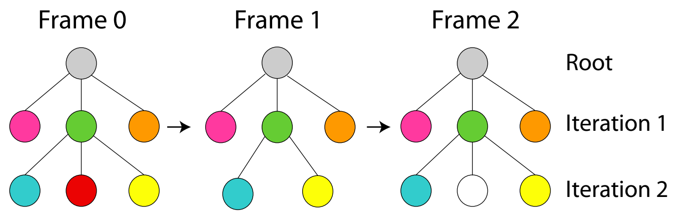


Figure 10 Automatic Colorization: In the Iteration 2, the red region disappeared in the second graph *i.e.*, the region was occluded. Therefore, this region color is discarded along the animation.

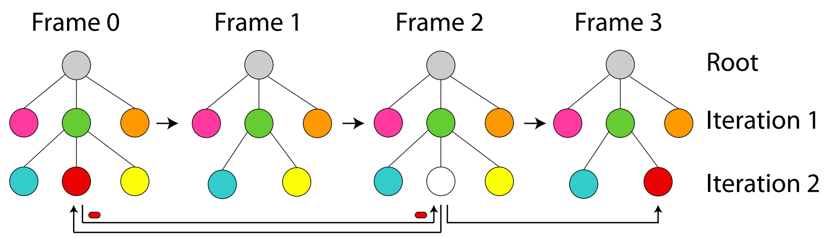


Figure 11 **Automatic Recursive Colorization: If a region disappeared, *i.e.*, the number of regions in the destiny-graph is bigger than in the source-graph, a reverse search through previous frames is made**.

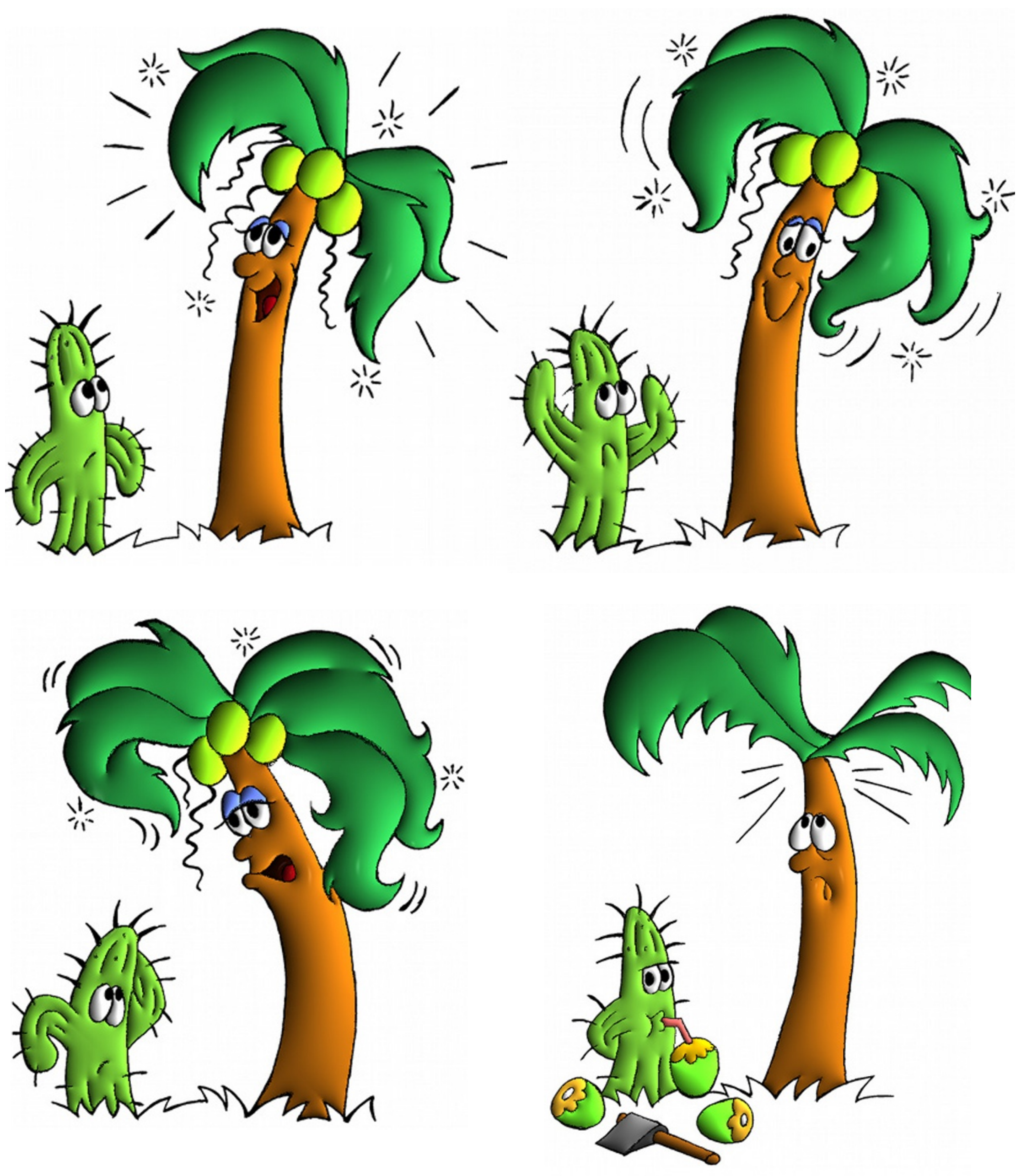


Figure 12 Colorization and Illumination from a free-hand sketch.

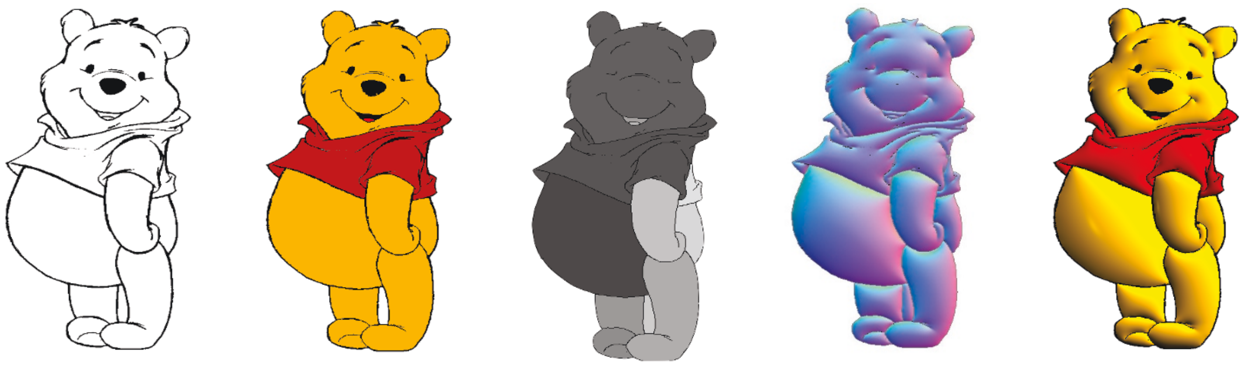


Figure 13 Illumination Process: From left: original image, colorized cartoon, segmented cartoon, normal map, illuminated cartoon.

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.

In Figure 16 we do not have a region occlusion. On the other hand, when the dog's hands touches his ears two regions are created changing the adjacency graph. Since

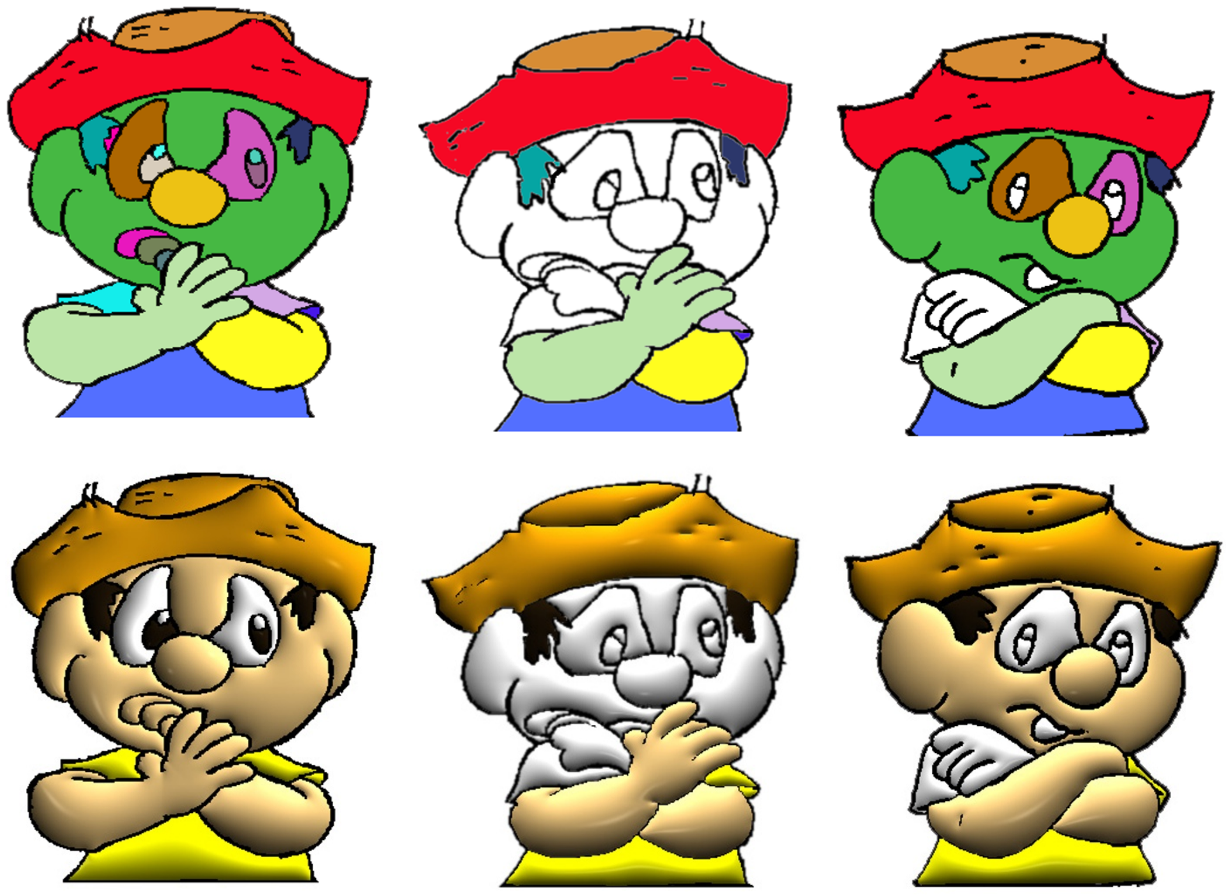


Figure 14 From left: segmented cartoon, split of two regions (face), recovery of the color.

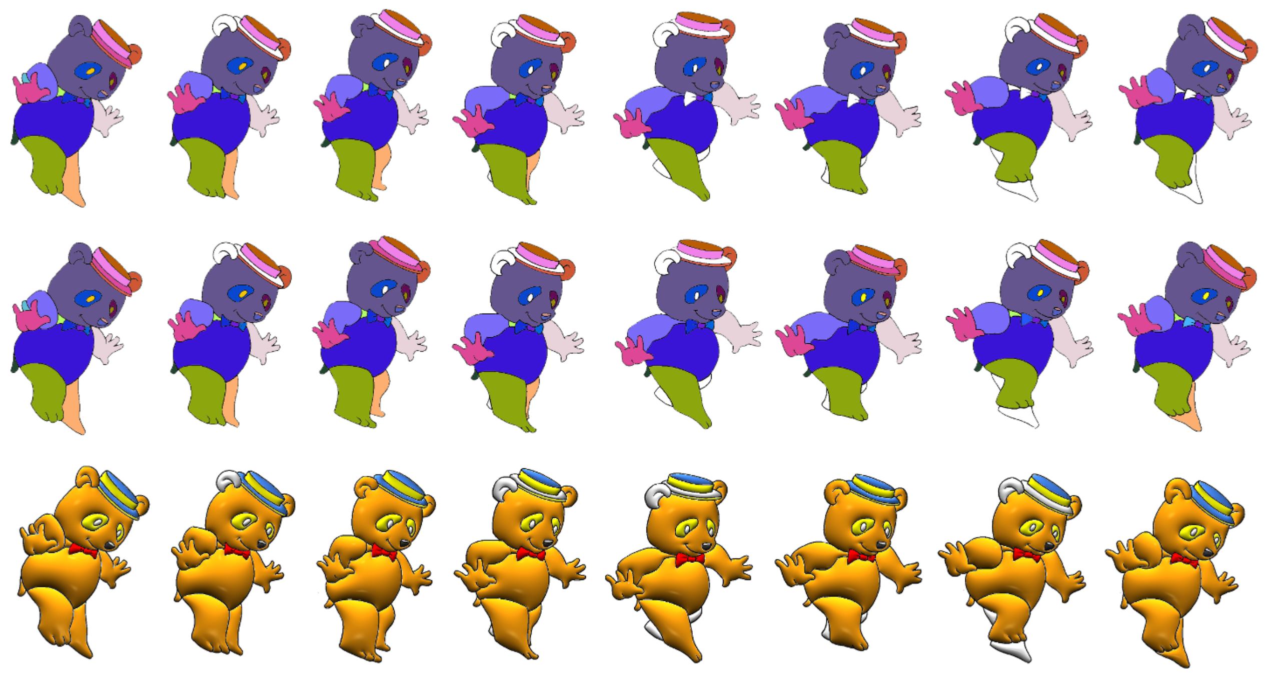


Figure 15 The first sequence shows the method from Bezerra et al. [4]. The second sequence shows our proposed method of colorization. The third sequence shows our illumination process.

these regions are new the algorithm do not colorize them. In the first sequence, once the color is lost, the method from Bezerra et al. [4] is not able to colorize the following frames. The recursive method however (second sequence), allows that the color information is propagated from previous frames making possible the gradual and complete cartoon's colorization (far right image).

The Figure 17 illustrates the result of the scale operator. We used different scalar factor values to illustrate the effect in our cartoon.

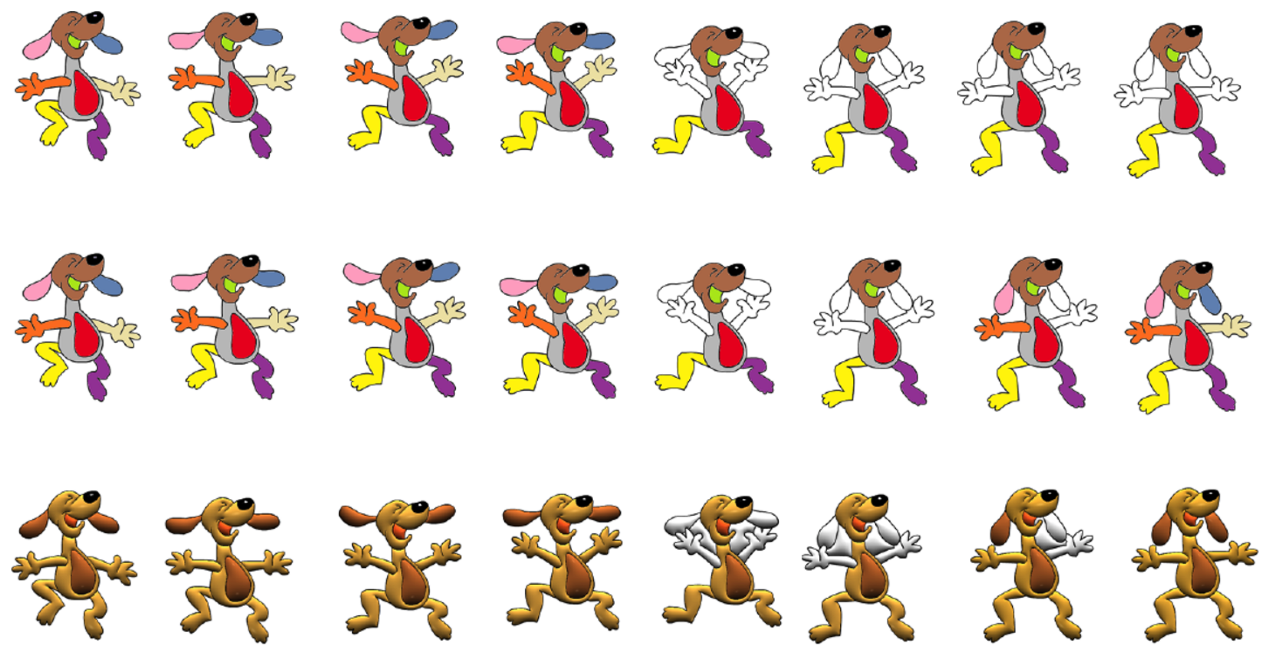


Figure 16 The first sequence shows the method from Bezerra et al. [4]. The second sequence shows our proposed method of colorization. The third sequence shows our illumination process.

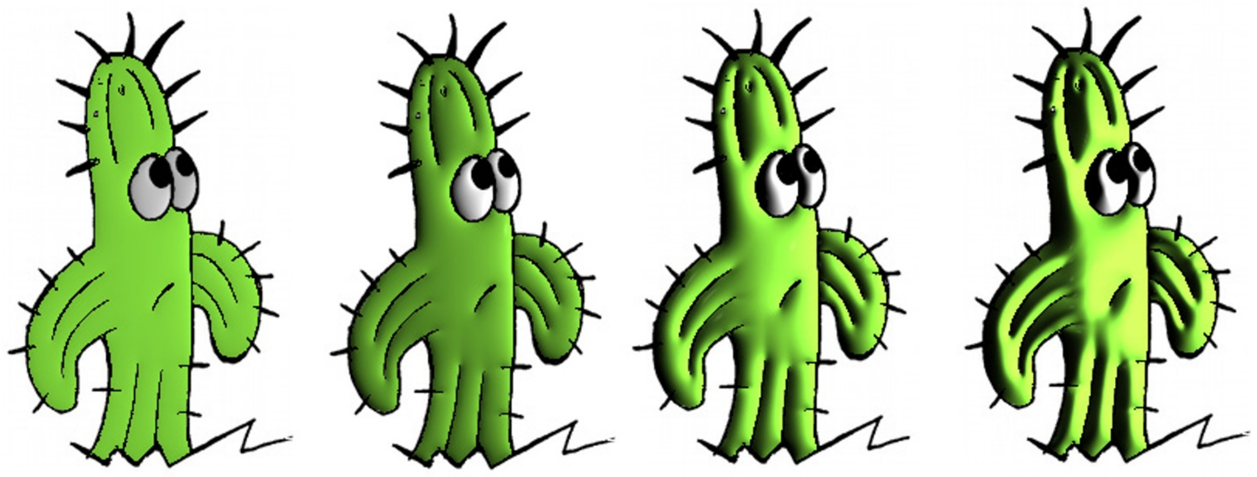


Figure 17 Scale Operator: From left the values are: 0.1, 0.5, 2.0, 10.0.

Limitations

Our method present some limitations both in the method of normal vector calculation as in the automatic colorization.

In the normal map calculation, if a point belongs to a curve, its z-component is 0. If this curve is interior to a region, our interpolation scheme will create singularities in the surface. It would be necessary to define normal operators to avoid these singularities.

In the automatic colorization, a problem could occurs in situations where the regions are similar, e.g., in a 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, since the position of the left leg is replaced by similar topological information right leg. Figure 18 shows this problem.

Another limitation is that illumination is not transferred frame by frame. It is computed separately in each frame.

Conclusion and future works

This work presented a new strategy based on a region-tree structure to represent 2D drawings. Unlike usual structures, our representation scheme 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 process.

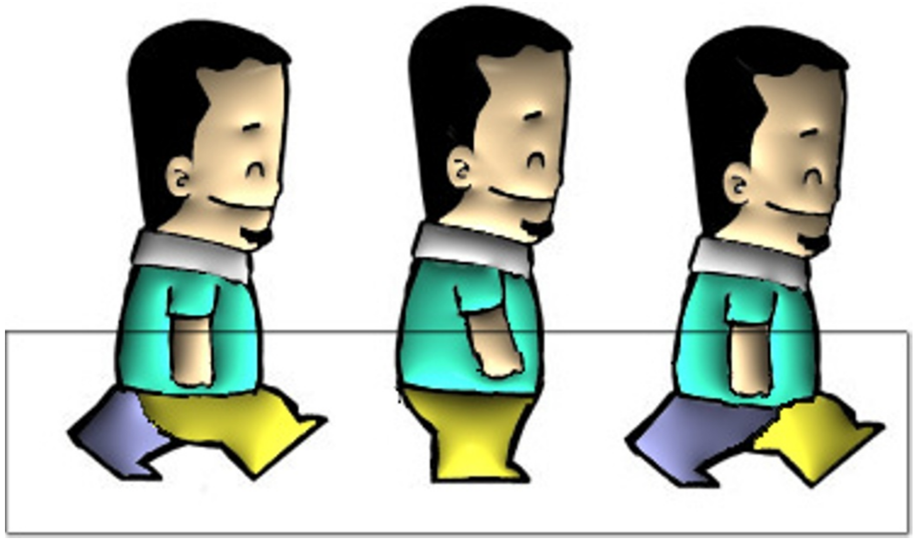


Figure 18 Problem with the tracking in walk motion.

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 mapping of the drawing based on our new topological structure. Our illumination method is suitable for independently shading parts of the drawing, like curves and regions.

The topological structure inspired a colorization method, where color is transferred recursively during the animation. With the recursive tracking method, we can recover most of the lost regions in the animation.

Future works include: verify time coherence together with spacial local structure to transfer frame by frame illumination; define new attribute operators; optimize the normal interpolation, by avoiding that invisible curve points affect the resulting normal inside the region.

Acknowledgements
The authors would like to thank L. Velho for encouraging this work's production. The authors also would like to thank J. Paixa˜o for the text's revision and D. Cedrim.

Author details
[1]Department of Mathematics, PUC-Rio, Rio de Janeiro, Brazil [2]Department of Computer Science, UFPE, Pernambuco, Brazil [3]Department of Computer Science, PUC-Rio, Rio de Janeiro, Brazil [4]Institute of Mathematics, UFAL, Maceio´, Brazil [5]Institute of Mathematics, UFBA, Salvador, Brazil

Received: 7 February 2012 Accepted: 6 March 2012
Published: 6 March 2012

References

1. Hunter GM (1997) Computer animation survey. Comput Graphics 2:225–229
2. Lasseter J (1997) Principles of traditional animation applied to 3D computer animation. Comput Graphics 21(4):35–44
3. Catmul E (1987) The problems of computer-assisted animation. Sig- graph 12(3):348–353
4. Bezerra H, Velho L, Feijo´ B (2006) A computer-assited colorization algorithm based on topological differences. In Sibgrapi 121–127
5. Chang CW, Lee SY (1998) Automatic cel painting in computer-assisted cartoon production using similarity recognition. In The Journal of Visualization and Computer Animation
6. Carvalho MA, Couprie M, Lotufo RA (2002) Image segmentation by analysis of scalespace. In Sibgrap 403–411
7. Carvalho MA, Lotufo RA (2001) Hierarchical regions matching in image sequences through association graph. In Sibgrap 396–404
8. Garrido S, Garrido L, Salembier P, Casas JR (1999) Representing and retrieving regions using binary partition trees. In International Conference on Image Processing
9. Horn B (1989) Shape from shading. MIT Press, Cambridge, MA
10. Wu TP, Sun J, Tang CK, Shum HY (2008) Interactive normal reconstruction from a single image. Siggraph Asia 27(119):1–119. 9
11. Igarashi T, Matsuoka S, Tanaka H, Teddy (2007) A sketching interface for 3D freeform design. In Siggraph courses
12. Zeleznik RC, Herndon KP, Hughes JF (2007) SKETCH: an interface for sketching 3D scenes. In Siggraph courses 2007
13. Phong BT (1975) Illumination for computer generated pictures. Communications of the ACM 18:311–317. doi:10.1145/360825.360839.
14. Johnston SF (2002) Lumo:illumination for cel animation. In Non-Photorealistic Animation and Rendering 45-ff
15. Sýkora D., *et al* (2010) Adding depth to cartoons using sparse depth (in) equalities. Computer Graphics Forum 29(2):615–623. doi:10.1111/j.1467-8659.2009.01631.x.
16. Hock SS, Tian F (2000) Computer-assisted coloring by matching line drawings. Visual Computer 16(5):289–304. doi:10.1007/s003719900068.
17. Sy´kora D, Burianek J, Zara J (2003) Segmentation of black and white cartoons. Spring Conference on Computer Graphics 245–254
18. Sýkora D, Burianek J, Zara J (2004) Unsupervized colorization of black-and-white cartoons. Non-Photorealistic Animation and Rendering 121–127
19. Trigo PG, Johan H, Imagire T, Nishita T (2009) Interactive region matching for 2D animation coloring based on feature's variation. Ieice Transactions 92:1289–1295
20. Zhang TY, Suen CY (1984) A fast parallel algorithm for thinning digital patterns. Communications of the ACM 27:236–239. doi:10.1145/357994.358023.
21. Vandevenne L (2004) Lode's computer graphics tutorial flood fill. http://lodev.org/cgtutor/floodfill.html
22. Parker J (1994) Pratical Computer Vision Using C. John Wiley and Sons, New York
23. Courant R, Robbins H (1941) What is Mathematics. Oxford University Press, England

doi:10.1186/2193-1801-1-1
Cite this article as: Nascimento *et al.*: **Computer-assisted coloring and illuminating based on a region-tree structure.** *SpringerPlus* 2012 **1**:1.