PropColor: Interactive Color Propagation for 2D Animations

Hari Hara Gowtham, Amal Dev Parakkat, Marie-Paule Cani

Vision, Modeling, and Visualization (2024)
L. Linsen and J. Thies (Eds.)

PropColor: Interactive Color Propagation for 2D Animations

Hari Hara Gowtham[1,2], Amal Dev Parakkat[1], Marie-Paule Cani[2]

[1]LTCI-Telecom Paris, Institut Polytechnique de Paris, France
[2]LIX-Ecole Polytechnique and CNRS, Institut Polytechnique de Paris, France

Abstract

Coloring is a fundamental yet time-consuming task in the 2D animation production pipeline. Traditional methods typically rely on frame-by-frame user interaction, leading to high user time and production costs. In this paper, we introduce PropColor, an interactive yet simple tool to propagate colors between adjacent frames of a hand-drawn animation. Starting with an initial frame colored by the user, our method propagates colors to neighboring frames based on the Delaunay triangulations computed from the sketch contours and the color hints. In addition to propagating colouring between frames, our method also associates a confidence score with each of them. This enables to identify the frames where user intervention is needed the most, either to validate the result or to provide additional color hints. Experiments show that our lightweight tool gives real-time feedback and significantly cuts down the animator's time.

CCS Concepts

• ***Applied computing*** → ***Arts and humanities;*** • ***Theory of computation*** → ***Computational geometry;***

1. Introduction

2D animations are distinguished by their stylized shapes and flat visual style. Though they were introduced in the early 20th century, they are still popular among artists and general audiences for their expressiveness. This versatile and accessible form of visual storytelling is, therefore, widely used in movies, animated series and mobile applications. Thanks to the advances in related technologies, various steps in the 2D animation production pipeline were made easier with automatic or semi-automatic tools, including sketch simplification and vectorization, inbetweening, lighting and shading.

Despite being one of the most important steps in the pipeline, the colorization process received relatively little attention in the vast literature on 2D animation tools. Although some works addressed the direct coloring of 2D animations, these approaches typically assumed sketches with closed contours, enabling colors to propagate into well-defined, closed regions. These methods encounter difficulties when handling artifacts, such as un-closed boundaries in the sketches. With the recent advances in deep/machine learning, various learning-based techniques were explored to robustly color 2D line art. While they can already handle vector drawings and sketches with gaps and only use minimal user inputs, they are restricted to coloring a single sketch. Therefore, to color a complete 2D animation, the artist has to individually edit each frame by giving color hints - a boring and time-consuming task. In practice, as most animations do not maintain closely connected contours at each frame, the cost of 2D production remains intrinsically linked to the number of images per second.

An important observation is that, while they make subtle adjustments to the character's poses, expressions, or positions in consecutive frames to create the illusion of life, animators try to minimize variations between adjacent frames to ensure that the animation runs smoothly, avoiding abrupt changes that would disrupt visual continuity. For example, the inset shows (as onion skinning) six consecutive frames in the animation of a change of expression in a character's face.

Based on this observation, we introduce PropColor, an interactive and easy-to-use tool for propagating colouring within a full animation sequence: The user starts by colouring an initial frame, using Delaunay Painting [PMC22], a fully geometric method robust to open contours. The colouring is then intelligently and seamlessly propagated to the subsequent frames. For example, Figure 1 shows a sample coloring given by our system. The user only had to provide a few color hints to colorize the initial frame. The hints were then automatically propagated to all the other frames - note their changes in the position shown in the inset - which cut down the animator's work by a factor of six in this case. Since a fully automatic propagation of coloring would not make sense, especially during a cut/scene transition or when a new region appears, our interface also associates a confidence score with the color propagation, allowing the animators to verify a few results and provide additional color hints or recolor a new frame if needed. Since the overall system does not require any complex/expensive

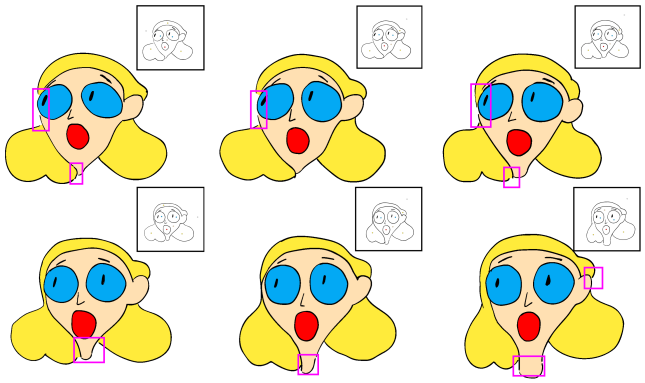


Figure 1: *Frames in an animation colored by propagating color hints from the first frame. The color hint positions are shown in the inset, and the gaps in the contours are emphasized in magenta.*

processing and only relies on Delaunay triangulation of the input sketch contours, PropColor is quite responsive and provides interactive feedback. It is also worth noting that the method does not add any restrictions over the input type: it can be used for animations sketched in a freehand style (for example, using a digital drawing pad), designed using scanned paper sheets or using vector sketches. If the input is a raster sketch (the focus of this paper), the pixels that are part of the contours are extracted, and their positions are used to compute the Delaunay triangulation.

In summary, our technical contributions are two-fold:

- An algorithm, robust to gaps in the contours of line drawings, to propagate color hints between adjacent frames of 2D animation;
- A "Jaccard index"-based confidence measure for identifying frames where user intervention is needed.

2. Related Work

Single sketch colorization/segmentation: Traditional methods can be categorized into two groups: methods which are tailor-made for handling open contours in line art coloring and methods that are designed for closing gaps in strokes or identifying regions. The latter, which can then be completed by a bucket-filling tool was typically used in semi-automatic coloring methods.

One of the prominent works in the first category, LazyBrush [SDC09], poses colorization as an energy minimization problem, allowing the colorization of line art with open contours from simple color hints. However, it is not simple to use as the resulting coloring depends on multiple user-defined parameters, necessitating a trial-and-error approach to create the desired coloring. There are also a few works that make use of available additional information, such as black and white versions [SBŽ05] or patterns in Manga drawings [QWH06], to help coloring. Later, Fourey et al. [FTR18] introduced an algorithm that creates a set of segments from the reference image as a way to handle stroke discontinuities. These segments, along with user-provided color hints, are then used for coloring. In [JSL21], the authors established a relationship between a gap point and its surrounding strokes to quantify gaps and used it to create regions for coloring, using an energy-based formulation. Recently, Delaunay Painting [PMC22] was introduced as a simple and interactive solution to color line art with gaps. A geometric approach to extend it to vector line art can be seen in Ohrhallinger et al. [OPM23]. It is worth mentioning that there are many learning-based solutions for coloring line arts [HA17,HJRD19,KJPY19,ZLSS*21,ZLW*18,CCBSS23]. However, these methods are usually trained on a specific class of inputs and do not always result in the desired flat coloring, especially when the regions are not well-defined, as they often fail to distinguish between intended and unintended gaps [YLL*22]. A very relatable work in this direction is FlatGAN [KLL*23], which segments the region maps for coloring sketches with gaps.

In stroke completion, one of the first works is by Gagnet et al. [GVTF94], who introduced an automatic gap-closing method based on a connected component analysis near potential gaps. Zhang et al. [ZCZ*09] introduced trapped-ball segmentation as a way to create regions from cartoon animations and is used subsequently to create regions from line drawings [FLB16]. Based on Gestalt principles, Liu et al. [LWH15] introduced a region identification procedure for cleaning rough sketch strokes. A Delaunay-based triangle grouping was used to identify regions for stroke simplification and is introduced in [PPM18], which was later improved by user intervention [PCS21]. Recently, perception-driven stroke connectivity for vector drawings was introduced by Yin and Liu et al. [YLL*22]. From a learning perspective, inpainting [SISSI17, WPSZ21], the relation between raw and clean drawings [SSII18a, SSII18b] or GAN [LDL*19] was used to close unintended gaps in raster drawings.

Colorizing 2D animations: While fast interaction methods robust to open contours such as LazyBrush [SDC09] (later extended to texture mapping [SBCČ*11]) may be used to colorize short animation sequences on a frame-by-frame basis, the automatic colorization of animations was only tackled so far for line-art with properly closed contours: Starting from [CL97], several methods such as [SF00, QST*05] tried to establish correspondences between closed regions in adjacent frames in order to propagate colors. In addition to the regions, the properties of the line drawings in both the raster [QST*] and vector formats [LWWS20] were later used. Instead of matching regions, Kanamori [Kan] compared ellipses that approximate regions to propagate colors between frames. Lastly, the correspondence between frames was further posed as a quadratic assignment problem matching problem [SMYA, LWL*23]. Instead of relying just on region shapes, the motion of regions was also taken into consideration in [ZLWH16]. Recently, region mappings were done under a learning framework [CZG*20, MKS*21, CPL21, DZL*24]. Unfortunately, all these works make the non-trivial assumption that the sketches can be decomposed into properly-defined regions.

In summary, to the best of our knowledge, no other work than LazyBrush can handle the coloring of 2D animations of sketches with gaps. As mentioned earlier, LazyBrush is very sensitive as it heavily relies on the input parameters and would require frame-by-frame interaction. In contrast, our method only requires user intervention every few frames and detects which frames the user should double-check in priority.

3. Overview

Our method was primarily developed for raster sketches - a particularly challenging task compared to vector animation. Nevertheless, without loss of generality, it is extendable to vector line art by appropriately sampling contour curves [OPM23].

The input is a 2D animation sequence with a colored initial frame. Though we could use any sketch coloring method ranging from simple bucket fill tool to more sophisticated methods to achieve this initial coloring, we use Delaunay Painting as it requires minimal inputs and can handle gaps in the sketch (common in hand-drawn animations). Using these colors, the sketch is then decomposed into regions as a subset of Delaunay triangulation of the foreground pixels and is used to compute representative circles corresponding to each region. Using an intersection-over-union ratio (Jaccard index) with respect to the representative circle, the best position for the color hint in the next frame is then identified (Figure 1 shows the propagation of color hints over the frames). We finally use all the hints to best propagate the colors. We also associate a confidence score with each propagation to identify the regions where we request the user to verify (or provide additional hints for) the coloring.

4. Color propagation algorithm

Let us first discuss the simplest solutions for propagating color hints from one frame to the next in animation before discussing our solution:

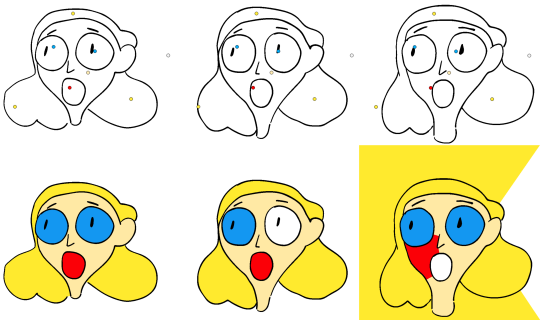


Figure 2: *Color hints propagated to the second and third frames using direct projection. Note that the blue hint for the left eye falls in the pupil in the second frame.*

- The most naive solution for color propagation is to directly project the user-defined color hints from the first frame to subsequent ones. But unfortunately, as shown in Fig. 2, this method misses the intended regions if the color hints are close to the contours of colored regions.
- A simple alternative is to move the color hint to the centroid of each colored region before propagating it to the next frame. Unfortunately, for complex shapes, this centroid might lie inside a smaller shape, as shown in Fig. 3.

Since these two naive solutions do not work, a third idea could be to propagate hints based on region overlap. However, in hand-drawn animations, large regions of an object tend to have more overlap between consecutive frames compared to thin regions of the same object. This is because larger regions of an animated object maintain more consistent shape and position across frames, while thinner or smaller parts may experience more dramatic changes in position or shape, resulting in less frame-to-frame overlap. Based on this observation and on the inspiration provided by the centroids idea, we proceed as follows:

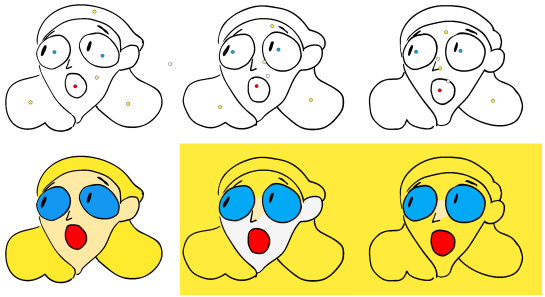


Figure 3: *Color hints moved to the centroid of each colored region in a frame before projection to the next one. Note that both the top-hair yellow and white background hints migrate to the face in the second frame.*

Starting from the first frame colored using Delaunay Painting (which propagates colors, based on the user's hints, among the triangles of the Delaunay triangulation of the contours), we identify the largest circle which could represent a colored region as the largest circumcircle of the Delaunay triangles that define it. By focusing on the largest circumcircle, which is, in fact, the largest medial ball, we can effectively capture and represent the most significant part of the region. We assume that the sketch strokes coarsely define the required boundary, making the disks defined by a circumcircle contained inside the region as much as possible. As a sanity check, we discard circles whose centers do not lie inside the region. Figure 4 shows a sample sketch with these circles, which we call representative circles (in green color). Since this representative circle is enclosed within the largest part of the region, which, as explained earlier, will have a significant overlap with the next frame, making the circumcenter of this circle a good place to keep the color hints.

Once the representative circles are identified, their centers are projected to the next frame. However, instead of directly placing color hints in the $k+1$th frame at these projected centers, we find a circle (among the circumcircle of Delaunay triangles) in the $k+1$th frame that best matches the representative circles in the kth frame. The color hints are moved to the centers of these best-matching circles, which makes the propagation of color hints more robust.

Jaccard Index: In practice "matching" between two circles is defined based on the Jaccard index, a well-known statistical measure for the similarity between two sets. Also known as the Jaccard similarity coefficient, it is defined as the size of the intersection of the sets A and B divided by the size of their union:

$$J(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|}$$

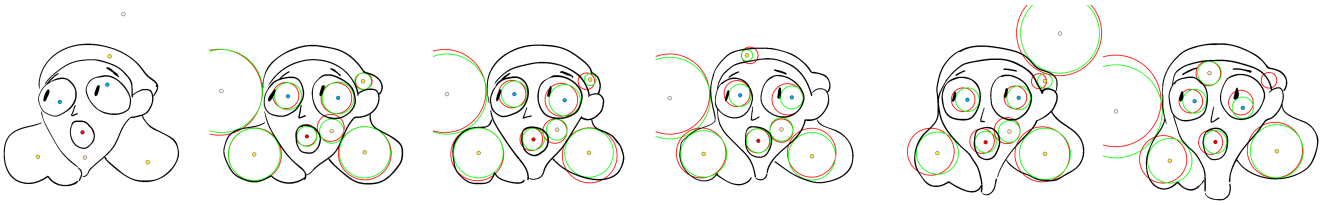


Figure 4: *The representative circles (in red) from kth frame and it's best matching circle in $k+1$th frame (in green). The initial user-given color hints are shown on the left.*

By definition, $0 \leq J(A,B) \leq 1$. if two sets A and B are disjoint, $J(A,B) = 0$. If both sets are identical, $J(A,B) = 1$. The value J, therefore, indicates how much the two sets are identical. In our case, the measure used in the above formula is the area of the two circles to be compared.

In practice, given a representative circle C_1 of a colored region in kth frame, we avoid the unnecessary overhead of computing the Jaccard index with respect to all possible cicumcircle of Delaunay triangles in $k+1$th frame as follows:

- The center of the C_1 is projected to the Delaunay triangulation DT_{k+1} of $k+1$th frame.
- Select the triangle $t \in DT_{k+1}$ in which the projected point is located and test the Jaccard measure with its circumcircle $C2$.
- Recursively traverse the neighbors of t (using a queue) until their circumcircle gives a Jaccard index of 0.

Please note that if all the objects in the frames had closed boundaries, a direct region-region intersection would have been enough to find this correspondence. But since we do not make such an assumption, the regions in $k+1$th frame are unknown until the color is propagated. Figure 4 shows an example demonstrating the use of representative and matching circles in color hint propagation.

The overall color propagation algorithm is shown in Algorithm 1. The $Delaunay\_Painting()$ procedure takes a frame as input and initiates the Delaunay Painting interface to color it. Whereas the $Delaunay\_Triangulate()$ function takes a frame as input and computes a Delaunay triangulation with the foreground pixels as the input points along with the corner points of the frame (corner points are $(0,0),(width_{frame},0),(0,height_{frame}),(width_{frame},height_{frame})$). The $Center()$ function computes the center of a circle, $CC()$ computes the circumcenter of a triangle, and $Locate()$ locates a point inside a given triangulation. Finally, the function $Add\_Color\_Hint()$ adds a new color hint of the given color at the given location for Delaunay painting in the $k+1$th frame.

Once the algorithm returns the new colored $k+1$th frame (i.e. the new set of color hints), we can then propagate the color to the $k+2$th frame, starting with the $Delaunay\_Painting()$ of the $k+1$th frame.

Algorithm 1 Color propagation

procedure PROPAGATE(k^{th} AND $k+1^{th}$ FRAMES)
  DP_k = Delaunay_Painting(k^{th} frame)
  $R_C = Q = \phi$
  for each colored region in DP_K **do**
    Identify the representative circle C
    $R_C = R_C \bigcup C$
  DT_{k+1} = Delaunay_Triangulate($k+1^{th}$ frame)
  for each representative circle $C \in R_C$ **do**
    c = $Center(C)$
    $Max_{JI} = 0$
    t = $Locate(c, DT_{k+1})$
    $Enqueue(Q,t)$
    while $Q \neq \phi$ **do**
      t_Q = $Dequeue(Q)$
      $JI = Jaccard\_Index(CC(t_Q), C)$
      if $JI > Max_{JI}$ & $CC(t_Q)$ lies inside the region **then**
        $Max_{JI} = JI$
        $Match\_Circle = CC(t_Q)$
      if $JI \neq \phi$ **then**
        for each neighbor $t_i \in DT_{k+1}$ of t **do**
          if t_i is not already visited **then**
            $Enqueue(Q, t_i)$
    $Add\_Color\_Hint(Center(Match\_Circle), Color(C))$
  return Colored $k+1$th frame

5. Confidence score

While the proposed color propagation algorithm is effective in many cases (see Figure 1), it may not always provide the desired coloring in all regions, particularly when there is a large jump in the animation or when a region appears or disappears. So, in order to assist the artist in this coloring process, we assign a confidence score with each propagation based on the Jaccard index. Depending on the confidence, the artist can then decide to verify the coloring at a specific frame and provide additional color hints, if needed.

To achieve this, we classify each color-hint propagation to a new frame into one of these four categories:

- $1 \geq Jaccard\_Index(C_1,C_2) > 0.9$: The circles align in a perfect manner.
- $0.9 \geq Jaccard\_Index(C_1,C_2) > 0.4$: Circles align nicely and the color is propagated.
- $0.4 > Jaccard\_Index(C_1,C_2) > 0.2$: Relatively less confidence. In this case, the area of the circles will be compared, and if they are similar (the ratio of their radii falls between 0.9 and 1.1), the

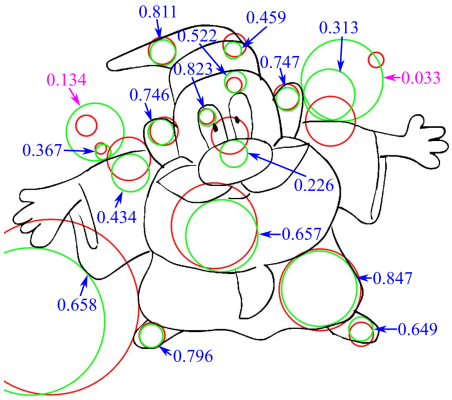


Figure 5: *Confidence score between propagating circle (red) and circle under consideration (green) in adjacent frames. The confident and unconfident scores are in blue and magenta colors, respectively.*

color will still be propagated; otherwise, we do not propagate the associated color hint, and prompt the user to check this frame.

- $0.2 > Jaccard\_Index(C_1, C_2) > 0$: Possibility of a significant change - region disappearing, a large change in the size of the region, or a jump in the animation. The color hint is not propagated.

The different confidence scores computed for the color-hints propagation between two consecutive frames are shown in Figure 5. Based on these confidence scores, we assign a global confidence score for each frame, which is defined as the least confidence score among all the color hints propagated from the previous frame. Please note that these are constant values determined by experiments and do not vary with different inputs.

In addition, to detect cut frames or scene transitions (where the camera switches between different shots), at the end of computing representative and matching circles, we count the proportion of representative circles that could be matched with high confidence. In practice, if the number of unmatched representative circles is more than 70%, we classify it as a cutscene. In such cases, all the colors are stopped from propagating.

6. Interface

Figure 6 shows the interface of PropColor. The user can load an animation, which will be treated as a sequence of images. As in Delaunay Painting, the user can color it accordingly using the provided color palette. Once the coloring is done, our interface automatically propagates the color hints to subsequent frames.

Based on the propagation and the mapping, as explained in the previous section, our interface includes a feature that provides users with real-time feedback on the confidence level of color propagation. As our algorithm applies coloring to successive frames of the animation, a small colored rectangle - whose color varies depending on the minimum confidence - is added as a thumbnail. This functionality offers users valuable insights into the coloring process, allowing them to verify or refine it as required. To make the

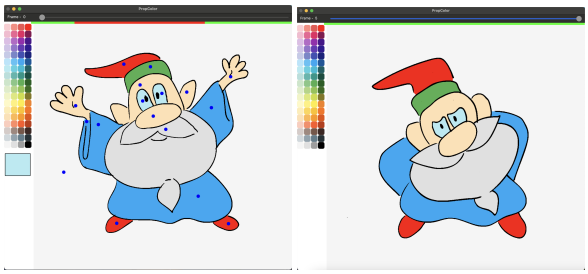


Figure 6: *Our interface after coloring the first frame (Left) and after coloring the entire animation (Right).*

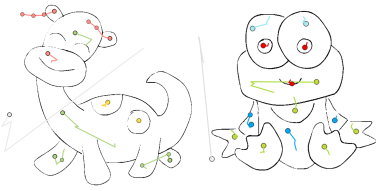
Figure 7: *Onion-skinned views of the animations of Omelette, Dino, Frog, Prawn, Birds and Santa shown in Figure 8.*

overall color coding of the rectangles easy, we use the following colors:

- Green: Indicates the frames where colors were propagated with high confidence, requiring no user intervention
- Orangish yellow to red: A color gradient that transitions from orangish-yellow (high confidence) to red (low confidence) for representing frames where the user has to verify or provide additional information respectively.
- Blue: Frames where the algorithm suspects a cut/scene transition, requiring user intervention to recolor it. Whenever such a frame appears, all the remaining rectangles will be colored in grey to denote that there is no color information to propagate.

Based on this color coding, the user can decide to jump to the frames which are colored with low confidence and provide additional color hints.

7. Results & Discussion



Figures 7 and 8, respectively, show the onion-skinned view of a few input animations and the coloring given using our method. Please note that all these examples have ill-defined boundaries and are raster sketches. The blue-filled circles show places where the user had to give a color hint. For simple animations like the Omelette, the user could give color hints only in the first frame and are successfully propagated to all other frames. As can be seen, whenever there is a comparatively large difference in the region size/position (ear of the Dino) or whenever a new region appears (leg of the Dino), the user has to provide additional user hints. Moreover, in some cases, the gap between regions

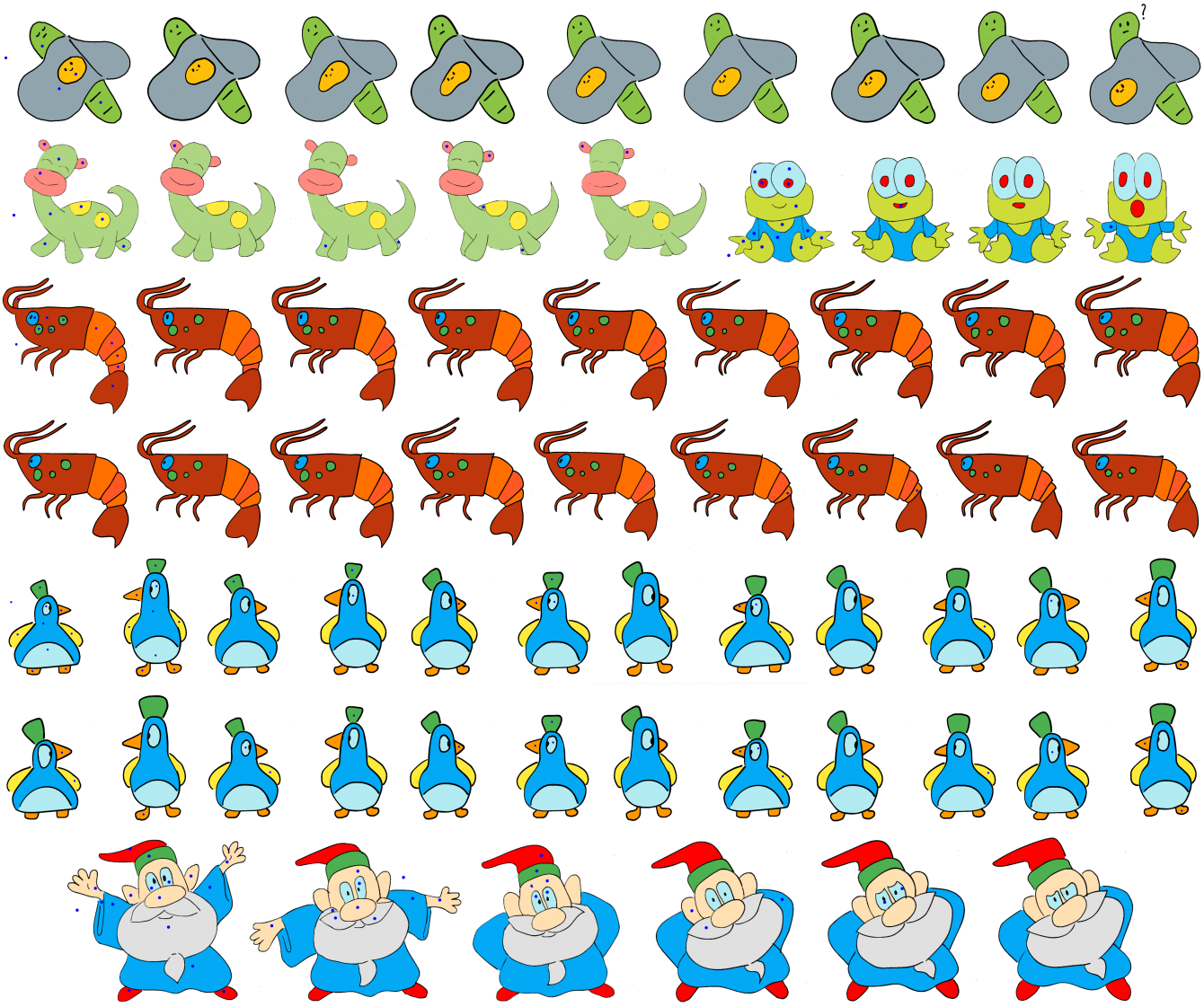


Figure 8: *Various frames in the animations of Omelette, Dino, Frog, Prawn, Birds and Santa, colored using PropColor. The places where the user had to give a hint are given in blue circles.*

changes, and color gets propagated from unintended parts and requires additional user hints (the moustache of Santa). As shown in the prawns, disappearing regions do not cause any issues. Our method also could handle slowly growing regions (hands of the Frog) and slowly translating regions (the birds) except whenever there is a big change in the size of the regions. The figures in the inset show samples of how the color hints are propagated during the animation.

Thanks to our interface, the user can directly skip the regions not requiring additional color hints, and hence save substantial time. Although the user still needs to provide color hints for a few frames, we argue that this is much easier than manually coloring each frame. In a small comparative study (shown in Figure 9) where we asked different users to color given sketches using multiple techniques, we found that except for the animation of Santa, using PropColor saved more than half of the time required for manual coloring, and up to 3/4 of the time. In the case of Santa, due to large jumps in poses, we had to give multiple user hints to get the desired coloring. This process took more time than usual but was still faster than all other techniques. Note that much smoother motion (high frame rate) is usually provided in a professional animation sequence. Also, it is worth noting that alternative techniques, such as projecting mouse clicks and projecting centroids, occasionally consumed more time than manual frame-by-frame coloring. The reason behind this is the fact that projecting the centroids and mouse clicks often creates wrong regions, so the user has to rectify this mistake before recoloring the regions appropriately. The chart also displays the processing time required by our system for computations, which, as shown, is lightweight.

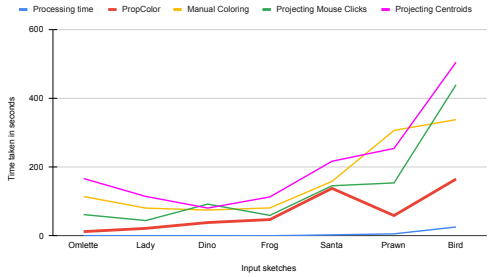
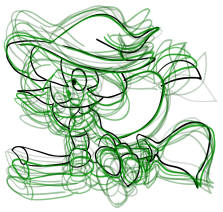


Figure 9: *Time required by different techniques to color various animations.*

The supplementary video showcases a short demonstration of our system in action, illustrating the coloring processes on three cases, ranging from an easy case (Lady) to a moderately easy (Prawn) and to a difficult one (Santa).



7.1. Limitations

As shown by the video, coloring an animation is made simple by our method, and using our interface was easy. However, we acknowledge that certain aspects of it could be improved:

Firstly, the interface shows a red color on the timeline to signify a frame where a region disappears, such as the merging of Lady's hair or a disappearing segment on the Prawn's body and Santa's hand, but the user then needs to search for this missing region. Finding some way to direct the user's attention to suspected places would be a nice addition, made easy by our knowledge of the circumcircles that mismatch between two consecutive frames and cause the bad confidence value.

In addition, while we assigned a continuous color code for confidence, we recognize that visually distinguishing them may be challenging. Instead, displaying an ascending/descending curve representing the frame's global confidence scores would offer a more intuitive means for inspection. The user could then directly jump to the frames corresponding to the local minima of this curve, the new hints being both propagated forward and backwards to the doubtful or still uncolored frames.

Moreover, despite its simple and easy-to-use nature, our color propagation algorithm also has two main drawbacks:

Reappearing regions: Currently, our system lacks the ability to track disappearing regions. As a result, whenever a region reappears, it is treated as a new region, necessitating user intervention. In the future, inspired by [ZLWH16], we aim to implement movement tracking to anticipate the reappearance of regions.

Jumpy animation: We made the assumption that the animation is smooth, meaning that adjacent frames - except in cut frames - should be almost similar. Therefore, our method is not robust enough for jumpy animations (a large difference between adjacent frames), so the user might have to provide more inputs. Figure 10 illustrates this case: Despite the large movement, regions like the hat, broom bristles and part of the hair did not require any additional inputs, whereas small regions like the eye need inputs in almost all the frames as the position of the eyes varies considerably. The figure in the inset shows the onion-skinned view of this animation to show the jumpiness (please note that there are only six frames). In the future, to address this drawback, we plan to take the region's shape into consideration while propagating color hints to neighboring frames.

8. Conclusion

We introduced PropColor, an interactive and easy-to-use interface for coloring 2D animations with ill-defined boundaries. As illustrated by many examples, our system requires the user to color the initial frame, and these color hints are propagated to subsequent frames based on Jaccard Index. To avoid issues that might arise due to the automatic color propagation, our system also assigns a confidence score to each propagation. Depending on this confidence measure, our interface prompts users for additional hints during the coloring.

While retaining its simplicity, we plan to address the main issues of our method in future work. Firstly, we will keep track of regions to avoid user intervention when a disappeared region reappears. Second, we plan to associate geometric parameters to colored regions while propagating color hints to address jumpy animations where consecutive frames may display large differences.

Acknowledgement

The authors would like to thank the reviewers for their valuable comments. This work is partially funded by the IP Paris Prematuration project DYNCOLOR.

References

[CCBSS23] CARRILLO H., CLÉMENT M., BUGEAU A., SIMO-SERRA E.: Diffusart: Enhancing line art colorization with conditional diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 3485–3489. 2

[CL97] CHANG C.-W., LEE S.-Y.: Automatic cel painting in computer-assisted cartoon production using similarity recognition. *The Journal of Visualization and Computer Animation 8*, 3 (1997), 165–185. 2

[CPL21] CASEY E., PÉREZ V., LI Z.: The animation transformer: Visual correspondence via segment matching. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021). 2

[CZG*20] CHEN S.-Y., ZHANG J.-Q., GAO L., HE Y., XIA S., SHI M., ZHANG F.-L.: Active colorization for cartoon line drawings. *IEEE Transactions on Visualization and Computer Graphics 28*, 2 (2020). 2

[DZL*24] DAI Y., ZHOU S., LI Q., LI C., LOY C. C.: Learning inclusion matching for animation paint bucket colorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 25544–25553. 2

[FLB16] FAVREAU J.-D., LAFARGE F., BOUSSEAU A.: Fidelity vs. simplicity: a global approach to line drawing vectorization. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 1–10. 2

[FTR18] FOUREY S., TSCHUMPERLÉ D., REVOY D.: A fast and efficient semi-guided algorithm for flat coloring line-arts. In *International Symposium on Vision, Modeling and Visualization* (2018). 2

Figure 10: *Various frames in a complex animation colored using PropColor (additional user-given color hints are given in blue circles).*

[GVTF94] GANGNET M., VAN THONG J.-M., FEKETE J.-D.: Automatic gap closing for freehand drawing. In *Siggraph'94* (1994). 2

[HA17] HENSMAN P., AIZAWA K.: cgan-based manga colorization using a single training image. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)* (2017), vol. 3, IEEE. 2

[HJRD19] HATI Y., JOUET G., ROUSSEAUX F., DUHART C.: Paintstorch: a user-guided anime line art colorization tool with double generator conditional adversarial network. In *Proceedings of the 16th ACM SIGGRAPH European Conference on Visual Media Production* (2019), pp. 1–10. 2

[JSL21] JIANG J., SEAH H. S., LIEW H. Z.: Handling gaps for vector graphics coloring. *The Visual Computer 37* (2021), 2473–2484. 2

[Kan] KANAMORI Y.: Region matching with proxy ellipses for coloring hand-drawn animations. In *SIGGRAPH Asia 2012 Technical Briefs*. 2

[KJPY19] KIM H., JHOO H. Y., PARK E., YOO S.: Tag2pix: Line art colorization using text tag with secat and changing loss. In *Proceedings of the IEEE/CVF international conference on computer vision* (2019). 2

[KLL*23] KIM H., LEE C., LEE J., KIM D., LEE K., OH M., KIM D.: Flatgan: A holistic approach for robust flat-coloring in high-definition with understanding line discontinuity. In *Proceedings of the 31st ACM International Conference on Multimedia* (2023), pp. 8242–8250. 2

[LDL*19] LIU F., DENG X., LAI Y.-K., LIU Y.-J., MA C., WANG H.: Sketchgan: Joint sketch completion and recognition with generative adversarial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 5830–5839. 2

[LWH15] LIU X., WONG T.-T., HENG P.-A.: Closure-aware sketch simplification. *ACM Transactions on Graphics (TOG) 34*, 6 (2015). 2

[LWL*23] LIU S., WANG X., LIU X., WU Z., SEAH H. S.: Shape correspondence for cel animation based on a shape association graph and spectral matching. *Computational Visual Media 9*, 3 (2023), 633–656. 2

[LWWS20] LIU S., WANG X., WU Z., SEAH H. S.: Shape correspondence based on kendall shape space and rag for 2d animation. *The Visual Computer 36* (2020), 2457–2469. 2

[MKS*21] MAEJIMA A., KUBO H., SHINAGAWA S., FUNATOMI T., YOTSUKURA T., NAKAMURA S., MUKAIGAWA Y.: Anime character colorization using few-shot learning. In *SIGGRAPH Asia 2021 Technical Communications*. 2021, pp. 1–4. 2

[OPM23] OHRHALLINGER S., PARAKKAT A. D., MEMARI P.: Feature-sized sampling for vector line art. In *Pacific Graphics 2023-The 31th Pacific Conference on Computer Graphics and Applications* (2023). 2, 3

[PCS21] PARAKKAT A. D., CANI M.-P. R., SINGH K.: Color by numbers: Interactive structuring and vectorization of sketch imagery. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (2021). 2

[PMC22] PARAKKAT A. D., MEMARI P., CANI M.-P.: Delaunay painting: Perceptual image colouring from raster contours with gaps. In *Computer Graphics Forum* (2022), vol. 41, pp. 166–181. 1, 2

[PPM18] PARAKKAT A. D., PUNDARIKAKSHA U. B., MUTHUGANAPATHY R.: A delaunay triangulation based approach for cleaning rough sketches. *Computers & Graphics 74* (2018), 171–181. 2

[QST*] QIU J., SEAH H. S., TIAN F., CHEN Q., MELIKHOV K.: Computer-assisted auto coloring by region matching. In *11th Pacific Conference on Computer Graphics and Applications, 2003*. 2

[QST*05] QIU J., SEAH H. S., TIAN F., WU Z., CHEN Q.: Feature-and region-based auto painting for 2d animation. *The Visual Computer 21* (2005), 928–944. 2

[QWH06] QU Y., WONG T.-T., HENG P.-A.: Manga colorization. *ACM Transactions on Graphics (ToG) 25*, 3 (2006), 1214–1220. 2

[SBCČ*11] SỲKORA D., BEN-CHEN M., ČADÍK M., WHITED B., SIMMONS M.: Textoons: practical texture mapping for hand-drawn cartoon animations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering* (2011). 2

[SBŽ05] SỲKORA D., BURIÁNEK J., ŽÁRA J.: Colorization of black-and-white cartoons. *Image and Vision Computing 23*, 9 (2005). 2

[SDC09] SỲKORA D., DINGLIANA J., COLLINS S.: Lazybrush: Flexible painting tool for hand-drawn cartoons. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 599–608. 2

[SF00] SEAH H. S., FENG T.: Computer-assisted coloring by matching line drawings. *The Visual Computer 16* (2000), 289–304. 2

[SISSI17] SASAKI K., IIZUKA S., SIMO-SERRA E., ISHIKAWA H.: Joint gap detection and inpainting of line drawings. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2017). 2

[SMYA] SATO K., MATSUI Y., YAMASAKI T., AIZAWA K.: Reference-based manga colorization by graph correspondence using quadratic programming. In *SIGGRAPH Asia 2014 Technical Briefs*. 2

[SSII18a] SIMO-SERRA E., IIZUKA S., ISHIKAWA H.: Mastering sketching: adversarial augmentation for structured prediction. *ACM Transactions on Graphics (TOG) 37*, 1 (2018), 1–13. 2

[SSII18b] SIMO-SERRA E., IIZUKA S., ISHIKAWA H.: Real-time data-driven interactive rough sketch inking. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–14. 2

[WPSZ21] WANG J., PAN G., SUN D., ZHANG J.: Chinese character inpainting with contextual semantic constraints. In *Proceedings of the 29th ACM International Conference on Multimedia* (2021). 2

[YLL*22] YIN J., LIU C., LIN R., VINING N., RHODIN H., SHEFFER A.: Detecting viewer-perceived intended vector sketch connectivity. *ACM Transactions on Graphics (TOG) 41*, 4 (2022), 1–11. 2

[ZCZ*09] ZHANG S.-H., CHEN T., ZHANG Y.-F., HU S.-M., MARTIN R. R.: Vectorizing cartoon animations. *IEEE Transactions on Visualization and Computer Graphics 15*, 4 (2009), 618–629. 2

[ZLSS*21] ZHANG L., LI C., SIMO-SERRA E., JI Y., WONG T.-T., LIU C.: User-guided line art flat filling with split filling mechanism. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021). 2

[ZLW*18] ZHANG L., LI C., WONG T.-T., JI Y., LIU C.: Two-stage sketch colorization. *ACM Transactions on Graphics (TOG)* (2018). 2

[ZLWH16] ZHU H., LIU X., WONG T.-T., HENG P.-A.: Globally optimal toon tracking. *ACM Transactions on Graphics (TOG)* (2016). 2, 7