Available online at www.sciencedirect.com
SCIENCE DIRECT
Image and Vision Computing 23 (2005) 767–782
image AND vision COMPUTING
www.elsevier.com/locate/imavis

Colorization of black-and-white cartoons

Daniel Sýkora[a,*], Jan Buriánek[b], Jiří Žára[a]

[a]Department of Computer Science and Engineering, Czech Technical University in Prague, FEE, Technická 2, 166 27 Praha 6, Czech Republic
[b]Digital Media Production, Mečislavova 164/7, 140 00 Praha 4, Czech Republic

Received 23 September 2004; received in revised form 22 April 2005; accepted 5 May 2005

Abstract

We introduce a novel colorization framework for old black-and-white cartoons originally produced by a cel or paper based technology. In this case, the dynamic part of the scene is represented by a set of outlined homogeneous regions which superimpose the static background. To reduce a large amount of manual intervention we combine unsupervised image segmentation, background reconstruction, and structural prediction. Our system allows the user to specify the brightness of applied colors unlike the most of previous approaches which operate only with hue and saturation. We also present simple but effective color modulation, composition and dust spot removal techniques able to produce color images in broadcast quality without additional user intervention.

Keywords: Colorization; Image registration; Image segmentation; Edge detection; Image analogies; Image restoration; Color-by-example; Patch-based sampling; Probabilistic relaxation

1. Introduction

In the history of black-and-white cartoon-making, it is possible to find really valuable and artistically advanced works which still stand up in front of the world-wide, modern cartoon production [1]. They provide an invaluable source of inspiration for each new generation of children. However, now these works are usually stored in depositories with non-optimal humidity conditions where they undergo a progressive degradation of visual quality. When one decides to stop this process, it is usually necessary to perform digital conversion [2] that preserves the same visual quality over the time and additionally it also reduces the amount of labor connected with restoration [3].

In this paper we discuss *colorization*, a challenging form of restoration which has been originally introduced in 1970 and later patented by Wilson Markle [4]. Colorization is defined as a process of transferring color information into the original monochrome material. Unfortunately in past decades it became really unpopular due to the opinion that it defiles the original work [5]. However, behind the main motivation for this modification is the well known virtue of color to enhance some specific artistic impressions [6]. Namely the presence of color in cartoons significantly increases the visual attraction that is important especially for a young audience.

From the technical point of view, colorization is a strongly ill-posed problem. A large ambiguity in color-to-intensity assignment cannot be resolved without additional knowledge about a scene structure. This crucial information is usually provided by a human and plays an important role in the whole process. Consequently an artist who wants to colorize black-and-white cartoon usually has to focus on featureless and repetitive work that prevents him from doing creative artwork. Currently available cartoon authoring systems (e.g. *Toonz*,[1] *Retas*,[2] *FlipBook*,[3] etc.) do not provide any semi-automatic toolbox able to simplify colorization of already designed black-and-white cartoon animations.

We show that using our novel approach, one is able to automate colorization pipeline, reduce the amount of hand-driven intervention and make the whole process temporarily feasible and thus cost effective. We bring forward the final visual quality, robustness, and ease of implementation by

* Corresponding author. Tel.: +420 2 24357470; fax: +420 2 24923325.
E-mail address: sykorad@fel.cvut.cz (D. Sýkora).

0262-8856/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2005.05.010

[1] http://www.toonz.com
[2] http://www.retas.com
[3] http://www.digicelinc.com

which our method may be successfully incorporated into existing cartoon authoring systems.

The rest of paper is organized as follows. First we present an overview of previous colorization approaches. Next, we describe our method including implementation details and optimization issues and finally we perform several experiments on real cartoon animations to validate the efficiency of proposed solution.

2. Previous work

Various semi-automatic colorization approaches have been published previously. They exploit several heuristics to estimate color-to-intensity assignment with a partial user intervention. In this section we describe these methods in detail. We also address main disadvantages which led us to develop our novel approach.

2.1. Luminance keying

A basic colorization technique commonly known as *luminance keying or pseudo-coloring* [7] utilizes user-defined look-up table to transforms each level of gray-scale intensity into a specified hue, saturation and brightness. This straightforward technique provides no possibility to apply different colors on the pixels that have the same intensity but differ in a spatial location. Only additional segmentation allows the user to define simultaneously a few look-up tables for different parts of the original image. Another problem is connected with luminance fluctuation which is common in old movies. This type of degradation causes a large derangement when the user assigns different colors to the similar intensities. Anyway luminance keying is usually only one possibility how to perform colorization in standard image manipulation and movie post-production tools and is extensively used.

2.2. Color-by-example

To partially alleviate limitations of luminance keying Welsh et al. [8] proposed techniques that exploits local textural information to discriminate ambiguity in the color-to-intensity assignment. Their approach is inspired by a method of color transfer between images [9] and by a framework of image analogies [10]. In their method luminance and color components are separated using $l\alpha\beta$ color space. Then a luminance distribution from a small neighborhood of each pixel in the target image is matched with a set of representative distributions in the source image. Such set is selected either by jittered sampling or manually using pre-defined rectangular swatches. When the best matching distribution is found, only a color information is transferred into the target pixel. The original luminance remains unchanged. Although Welsh's technique is surprisingly successful when applied to some specific natural scenarios, in cartoons, where a scene structure is mainly represented by a set of homogeneous regions, the luminance distribution collapses to a single peak which has indeed the same discriminative power as look-up table in luminance keying.

2.3. Motion estimation

Another well known approach to colorization [11,12] assumes that changes between two consecutive frames are small, therefore, it is possible to use optical flow to estimate dense pixel-to-pixel correspondences. Chromatic information can be then transferred directly between the corresponding pixels. However, usually lots of key-frames have to be colorized manually to cover changes in a scene structure and also to correct colorization when the optical flow estimation failed. This problem is frequent especially in cartoon animations where the extent of motion between two consecutive frames is large as compared with real movies. Consequently colorization based on the motion estimation remains still labor intensive.

2.4. Color propagation

In contrast to previous approaches, Horiuchi [13] was the first who assumes that the homogeneity in the gray-scale domain indicates homogeneity in the color domain and vice versa. This assumption provides a possibility to propagate color from several user-defined seed pixels to the rest of the image. To accomplish this he suggested to exploit *probabilistic relaxation* [14] with a simple relaxation rule that propagates color information faster between the pixels that have similar gray levels. However, this color propagation scheme introduces a large computational overhead which is not tractable for real applications. Horiuchi and Hirano [15] later proposed an approximation that produces similar results much faster but on the other side it also introduces a lot of visible artifacts.

Levin et al. [16] introduced similar framework to Horiuchi. In their technique, the aim is to minimize squared difference between a pixel color and a weighted average of colors in a small neighborhood. According to the assumption of homogeneity they assign more weight to pixels that have similar intensities. To incorporate user-defined colors the minimization is performed subject to constraints defined in seed pixels. Such a global optimization scheme results in a large sparse system of linear equations which is solvable using number of standard methods. Levin's framework is also suitable for image sequences. In this case, motion estimation [17] is used to compensate movement between two consecutive frames and then the color information can be propagated both over the time and space.

Sapiro [18] proposed yet another color propagation scheme where the minimization is performed in the gradient domain. This optimization scheme yields a system of Poisson equations with Dirichlet boundary conditions [19] which can be also extended for image sequences using 3D

version of Laplace operator. To define color constraints Sapiro used Cb and Cr components in YCbCr color space. However, it is also possible to perform minimization in a full RGB color space because the original gradient field is preserved in each channel. The main problem is that Sapiro's method produces significant color bleeding near the strong edges which is visually disturbing.

Recently, Yatziv and Sapiro [20] introduced a novel scheme that significantly outperforms previous approaches both in the speed and visual quality. For each uncolored pixel in the gray-scale image they compute *geodesic distance* [21] to several nearest seed pixels with different colors assigned. The final color is then computed as a weighted average of seed colors where the weight is proportional to the corresponding geodesic distance. Similar to other color propagation schemes also this approach can be easily extended to image sequences. In this case, the color is propagated both in the space and time using 3D geodesic distance. Moreover, Yatziv and Sapiro also introduce a simple extension which allows the user to change the underlaying gray-scale intensity. Using two different seed pixels marked as background and foreground they first annotate the image and then they compute the colorization. This process results in an approximate alpha-matte which can be used for various image editing tasks such as intensity manipulation.

Although both Yatziv's and Levin's color propagation schemes produce visually compelling results in much shorter time frames, it is still necessary to draw many well-placed scribbles frame-by-frame to produce good-looking colorization.

2.5. Segmentation

Our approach is similar to [22]. In this framework, manual image segmentation is used to divide the original gray-scale image into a set of layers. Then for each of them the alpha channel is estimated using *Bayesian image matting* [23]. This decomposition allows to apply colorization (they use Welsh's approach) or any other image manipulation technique in each layer separately and then reconstruct the final image using alpha-blending. As one can see, this is the classical workflow commonly used in standard image manipulation software which is labor intensive and thus not tractable for cartoon animations.

3. Colorization framework

In this section, we describe our novel framework which eliminates shortcomings of previous approaches. First, we present an overview of colorization pipeline and later, in successive sections, we describe each step in more details including implementation and optimization issues.

3.1. Framework overview

The key assumption in our method is that each animation frame has been created as a composition of two planar layers (foreground and background) which have usually different visual appearance. A dynamic foreground layer consists of several homogeneous regions bounded by well visible outlines. On the other side, a background layer is a static image and usually contains a more complicated textural information. This important property leads us to process each layer separately using a different colorization technique (see Fig. 2).

In the first stage, we perform *image segmentation* (see Section 3.2) to divide the input frame into a set of regions. In this step an outline detector helps us to locate region boundaries (see Fig. 1c). An area size thresholding is then used to estimate which region belongs to a background and which to a foreground layer (see Fig. 1d). In the next phase, we track camera motion through the sequence and using several background fragments we mosaic one big image which contains a visible portion of the whole background layer (see Section 3.3). Such image can be then colorized at one snap using standard image manipulation tool (see Figs. 1g and 9).

In contrast to a static background, a dynamic foreground layer has to be colorized frame-by-frame. To speed up this process we retrieve *structural correspondences* (see Section 3.4) between consecutive animation frames. They help us to estimate color labels for each uncolored region in a novel frame using yet colorized frames as an example. During this step, a limited manual interaction avoids propagation of

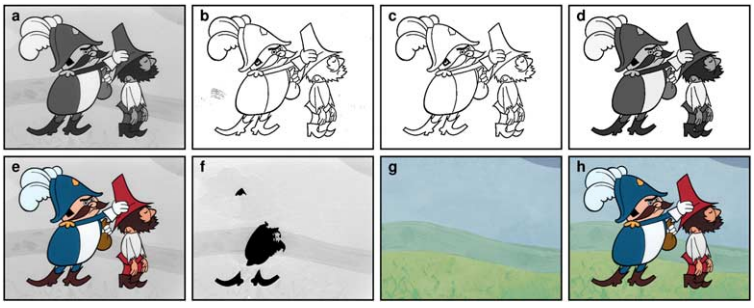

Fig. 1. Colorization framework in progress: (a) the original gray-scale image, (b) outline detection, (c) outline extraction, (d) foreground layer segmentation, (e) foreground layer colorization, (f) background layer reconstruction, (g) background layer colorization, and (h) the final composition.

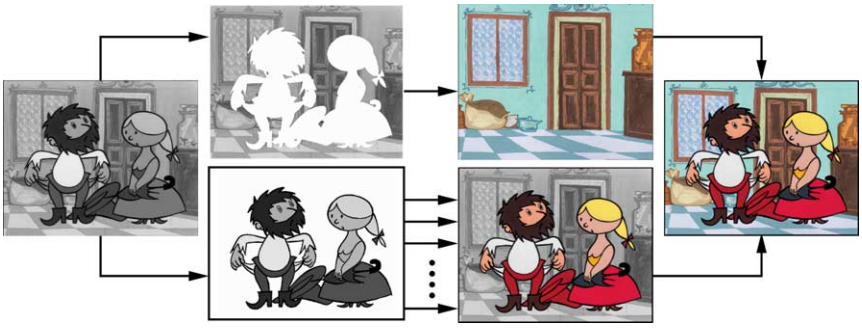

Fig. 2. Colorization pipeline—a static background (top) is reconstructed from a number of animation frames. Color is then applied on the whole layer at one snap using standard image manipulation tool. A dynamic foreground (bottom) is colorized semi-automatically frame-by-frame.

prediction errors into the consecutive frames and consequently guarantees the final visual quality.

A *color image synthesis* phase follows (see Section 3.5). In this step, user-defined hue and saturation are applied automatically on each pixel in the foreground layer while the brightness of the final color is modulated using the original gray-scale intensity (see Fig. 1e). Also dust spots and band scratches are removed automatically exploiting region homogeneity. Finally, camera motion estimation helps us again to extract a visible portion of already colorized background layer which is then composed with already colorized foreground layer (see Figs. 1h and 2).

3.2. Segmentation

In order to separate the original gray-scale image into a set of regions we adopt and improve unsupervised segmentation technique first presented in [24]. This method utilizes convolution with Laplacian-of-Gaussian masks ($\mathbf{L}\circ\mathbf{G}$) to detect outlines [25]. Using $\mathbf{L}\circ\mathbf{G}$ we can perform two operations in one pass: Gaussian suppresses noise and Laplacian estimates second-order derivatives of a noise-free image. Such a filter can be derived as follows

$$\mathbf{G} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{1}$$

and

$$\mathbf{L} = \nabla^2 = \nabla\mathrm{o}\nabla = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}, \tag{2}$$

where $\mathbf{G}$ denotes Gaussian filter and $\mathbf{L}$ is Laplace operator both in two dimensions. Due to convolution linearity $\nabla^2(\mathbf{G}\circ\mathbf{I}) = (\nabla^2\mathbf{G})\circ\mathbf{I}$, it is possible to pre-calculate $\nabla^2\mathbf{G}$ using symbolic derivation and thus $\mathbf{L}\circ\mathbf{G}$ can be expressed in a closed-form:

$$\begin{aligned}\mathbf{L}\mathrm{o}\mathbf{G} &= \nabla^2\mathbf{G} \\ &= \frac{1}{\pi\sigma^2}\left(\frac{x^2+y^2}{2\sigma^2} - 1\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right).\end{aligned} \tag{3}$$

The important feature of $\mathbf{L}\circ\mathbf{G}$ is that edges in the original image cause zero-crossings in the filtered image. Moreover, these zero-crossings form closed curves, consequently filter response inside the region should be only positive or negative. If we consider that outlines are usually darker in contrast to other regions then the response of $\mathbf{L}\circ\mathbf{G}$ inside the outline will be always negative (see Fig. 3b). This important property enables us to design a robust and accurate outline detector.

3.2.1. Outline detector

To detect outlines using $\mathbf{L}\circ\mathbf{G}$ we use the following adaptive algorithm. First we create a binary image where the black pixels denote negative response of $\mathbf{L}\circ\mathbf{G}$ in the original image. We call this image $\mathbf{L}\circ\mathbf{G}$-negative mask. It helps us to find out $\mathbf{L}\circ\mathbf{G}$-negative pixels p_i which has the

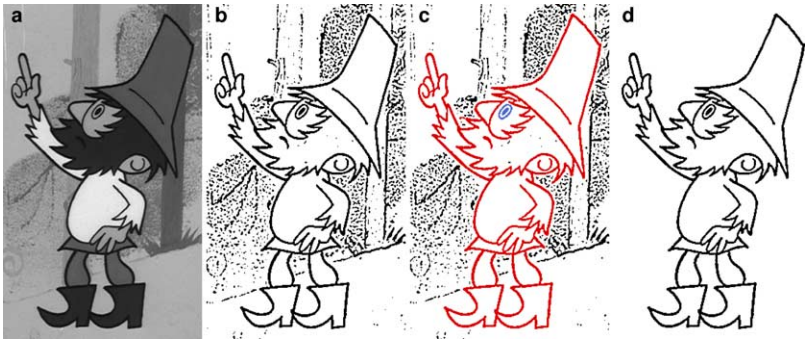

Fig. 3. Outline detector in progress: (a) the original image, (b) $\mathbf{L}\cdot\mathbf{G}$-negative mask, (c) adaptive flood-filling, and (d) extracted outlines.

Fig. 4. Unsupervised segmentation in progress: (a) outlines, (b) regions, (c) color-to-region assignment, and (d) region growing.

minimal gray-scale intensity $I_{\min}$. At these pixels we start a standard flood-fill algorithm to mark out other **L**○**G**-negative pixels which lie within the same **L**○**G**-negative region. Using pixels from this region we compute intensity median $\tilde{I}$. For the next flood-filling step we select yet unfilled **L**○**G**-negative pixels which satisfy the condition $I(p_i) < k\tilde{I}$ where convergency constant k has been experimentally adjusted to 0.5. We iterate this process until the intensity median of a new filled area is lower or the same as the value from the previous step. Finally we remove residual **L**○**G**-negative pixels to convert our **L**○**G**-negative mask in to the final mask of outlines.

This adaptive outline marking algorithm converges very fast. At average it stops after the second pass when a new median $\tilde{I}_{k+1}$ is not bigger than a previous $\tilde{I}_k$. This behavior is depicted in Fig. 3c where almost all outlines are marked during the initial step while only the eye has been marked in the second iteration.

3.2.2. Outline authentication

In most cases all important outlines are extracted without additional user intervention. Rarely outlines in the foreground layer coincide with **L**○**G**-negative areas in the background layer (see small beard protuberance in Fig. 3c). Usually **L**○**G**-negative region in the background layer is much brighter so it is possible to apply the following outline authentication test: we use the minimal intensity value $I_{\min}$ over a small neighborhood of the current pixel to compare the median value $\tilde{I}_n$ from the last iteration of our algorithm. If $\tilde{I}_n < I_{\min}$ we conclude that such a pixel does not represent the foreground outline.

3.2.3. Classification

Using extracted outlines it is now easy to assign unique index for each closed region. We achieve this by starting flood-fill at each white pixel using unique region index. However, this can be also done by a standard scan-line based two-pass region marking algorithm [26].

To classify whether the region belongs to a background or foreground layer we perform region area size thresholding. In our experiments the critical size has been set to 15% of the total image size. Regions with the area size below this threshold are classified as a foreground and others as a background. It is obvious that using such approach it is not possible to detect background regions which are in fact only small holes in the foreground (see region between legs in Fig. 4). Another possibility is to estimate region homogeneity by examining luminance histogram [27]. If we found two or more significant peaks, we classify region to be in the background otherwise we keep it in the foreground layer. However, when the occluded part of the background is also homogeneous or when the region contains, e.g. dust spots (see Fig. 15a) we are still not able to distinguish it. Such regions have to be first classified manually and later automatically using structural prediction.

3.2.4. Adaptive σ-fitting

The standard deviation σ is an important parameter which controls the response of **L**○**G** filter. If we vary σ we

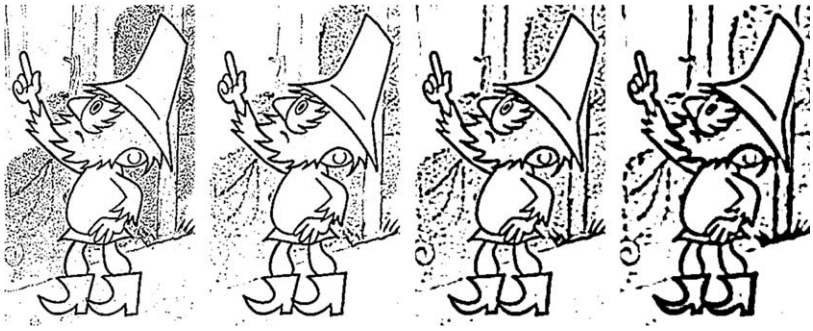

Fig. 5. **L**○**G**-negative mask with increasing σ (from left to right): 1.0, 1.5, 2.0, and 3.0.

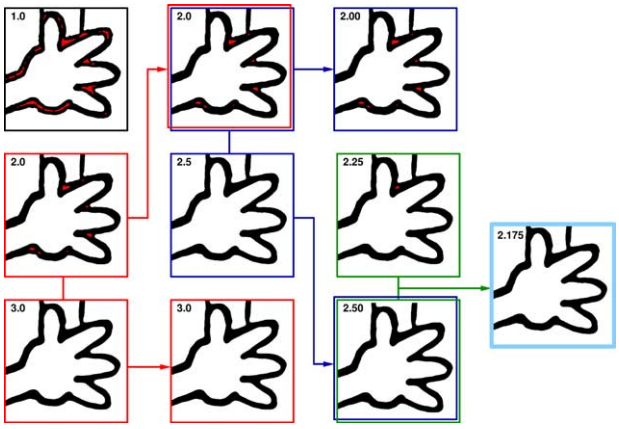

Fig. 6. Adaptive σ-fitting in progress—iterative elimination of spurious regions inside the outline. In this simple example we select two intervals $\sigma \in (1.0, 2.0)$ and $\sigma \in (2.0, 3.0)$. While $\sigma = 2.0$ does and $\sigma = 3.0$ does not produce spurious regions, we decide to refine the interval $\sigma \in (2.0, 3.0)$ using $\sigma \in (2.0, 2.5)$ and $\sigma \in (2.5, 3.0)$. The same approach is then applied recursively until the difference between interval endpoints is less than 0.25.

focus on edges at different scales [28]. See Fig. 5 for several examples of $\mathbf{L}\circ\mathbf{G}$-negative response in different scales. When the thickness of outlines is only several pixels (<6) usually a constant σ (e.g. we use $\sigma < 1.25$) is sufficient for most images. However, when the thickness is greater (>6) then so called *phantom edges* [29] will arise. They correspond to a local minima of first-order derivatives and bound small spurious regions inside the outline (see Fig. 6). In this case, the optimal σ have to be small enough to preserve important details but also large enough to eliminate phantom edges.

To avoid tedious manual tuning we propose a coarse-to-fine strategy to retrieve the optimal σ automatically. Initially we divide the scale-space into a set of smaller intervals. At each interval endpoint we compute the number of spurious regions after the segmentation. Then we select an interval I where the lower bound σ still produces several spurious regions but the upper bound σ guarantees that the number of spurious regions falls under a specified limit. We recursively apply the same strategy on I till the difference between the lower and upper bound σ is considerably small (see Fig. 6). This full search algorithm is invoked only when the first animation frame is processed. During the animation we only refine the optimal σ using a small initial interval around the optimal σ from the previous frame.

3.2.5. Allowable aliasing energy

The another important parameter which improves the quality of outline extraction is *allowable aliasing energy* p_a. If we truncate convolution kernel of $\mathbf{L}\circ\mathbf{G}$ filter in the spatial domain we obtain a periodical repetition in the frequency domain and vice versa. This repetition introduces the aliasing energy which can be expressed as follows

$$\frac{100 - p_a}{100} = \frac{\sigma^6}{2\pi} \int_{-\alpha}^{\alpha} \int_{-\alpha}^{\alpha} \frac{(u^2 + v^2)^2}{\exp(\sigma^2(u^2 + v^2))} \mathrm{d}u\, \mathrm{d}v, \tag{4}$$

where α is the aliasing frequency. Sotak and Boyer [30] compute α for a given p_a and a standard deviation σ using numerical integration. If we increase p_a we propagate smoothing due to low-pass property of a truncated kernel (see Fig. 7). According to several experiments performed on real cartoon images we found that it is possible to use constant value $p_a = 10\%$ which is optimal in most cases.

3.2.6. Optimization issues

While brute force $\mathbf{L}\circ\mathbf{G}$-filtering is time consuming, we recommend to use several optimization techniques that provide in practice more than tenfold speed-up. The original gray-scale image can be first pre-smoothed using separable version of Gaussian filter and afterwards $\mathbf{L}\circ\mathbf{G}$ convolution with smaller support can be applied [31]. We also exploit the fact that $\mathbf{L}\circ\mathbf{G}$ can be exactly decomposed into a sum of two separable filters [32]

$$\nabla^2\mathbf{G}(x, y) = \mathbf{h}_1(x)\cdot\mathbf{h}_2(x) + \mathbf{h}_2(x)\cdot\mathbf{h}_1(y), \tag{5}$$

where

$$\mathbf{h}_1(\rho) = \frac{1}{\sqrt{2\pi\sigma^2}}\left(1 - \frac{\rho^2}{\sigma^2}\right)\exp\left(-\frac{\rho^2}{2\sigma^2}\right) \tag{6}$$

and

$$\mathbf{h}_2(\rho) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\rho^2}{2\sigma^2}\right). \tag{7}$$

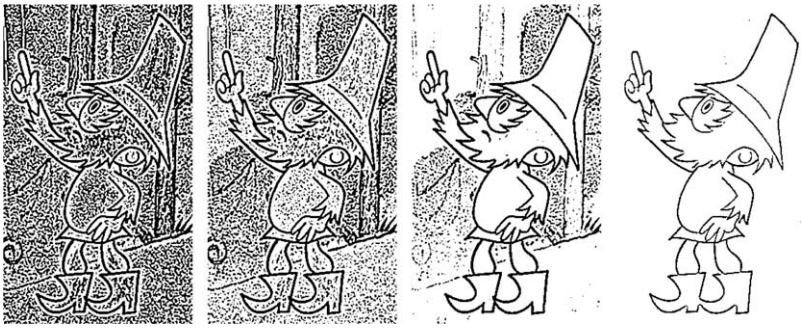

Fig. 7. $\mathbf{L}\circ\mathbf{G}$-negative mask with increasing allowable aliasing energy (from left to right): $p_a = 1$, 5, 10, and 25%.

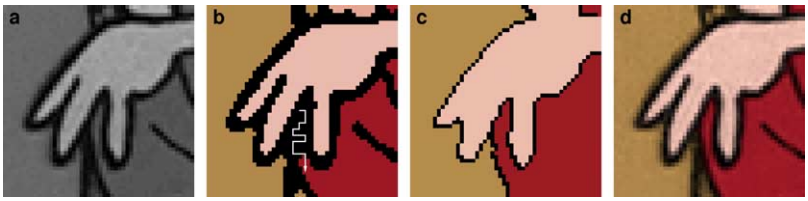

Fig. 8. Region growing: (a) a detail of the original image, (b) computing the minimal geodesic distance, (c) segmentation after region growing, and (d) segmentation driven colorization.

Additionally when the scale of outlines requires a large σ, we can apply image decimation [33] which allows to use smaller supports for $\mathbf{L}\circ\mathbf{G}$-filtering. Equivalent standard deviations for decimated supports are estimated using step-by-step filter design procedure suggested in [30].

3.2.7. Region growing

We additionally need to retrieve intrinsic boundaries of regions (see Fig. 4d) in order to estimate how to flood color into the outline. For a binary image the union of these axes is also known as *image skeleton* [34]. However, in our case we have additional information about the underlaying gray-scale intensity which provides us to reach better accuracy as compared with skeletonisation of a binary image. Due to these circumstances we do not extract skeleton, instead for each outline pixel we retrieve the nearest region using best-first approximation of the minimal *geodesic distance* as in [20] (see Fig. 8).

3.3. Background layer reconstruction

In order to extract the background layer from a given sequence of frames it is necessary to estimate camera motion through the sequence. To accomplish this we exploit robust hierarchical motion estimation [35] which is able to determine motion vectors between two consecutive frames

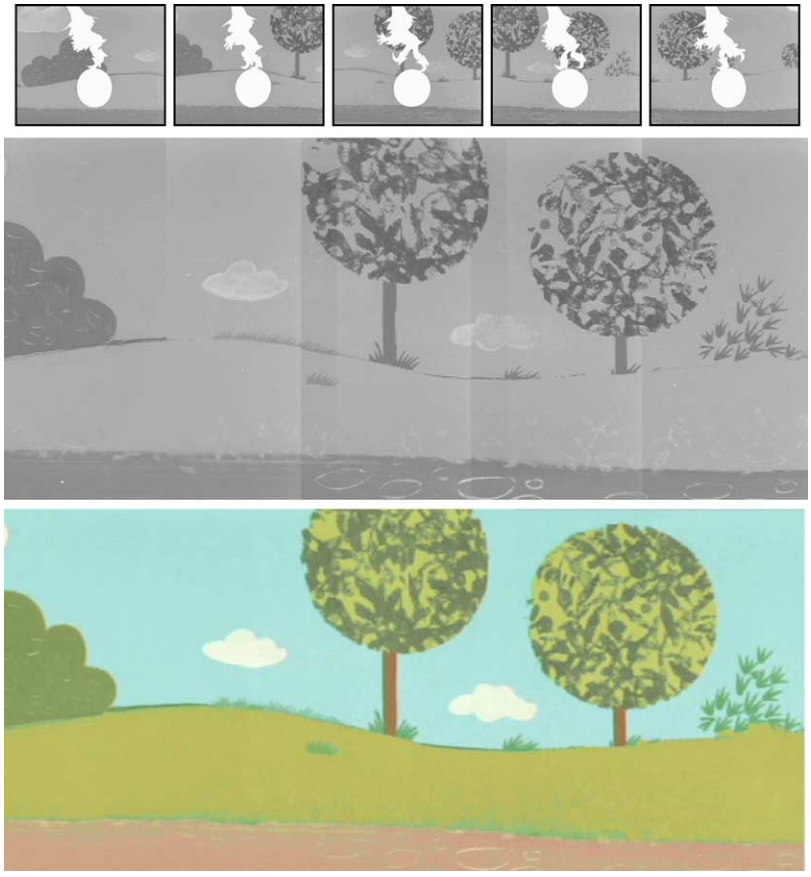

Fig. 9. Background layer reconstruction and colorization—incremental motion estimation over a set of consecutive frames (top): estimation support is restricted to the background layer, intermediate seamy reconstruction (middle): brighter vertical bands arise due to luminance fluctuation or image vignetting, and the final seamless reconstruction followed by manual colorization (bottom).

under a specified model. We use only translation and scale by the reason that camera motion in cartoon animations is usually restricted to a planar panning and zooming. To be more robust we also set up the motion estimation support to these parts of the original frame which have been classified as the background layer in the previous image segmentation phase.

The second issue is a seamless reconstruction of the whole background layer. The problem arises when several acquired fragments of background suffer from image vignetting, contour anti-aliasing, dust spots, band scratches, or other local and global distortions (see Fig. 9) which would degrade the final visual quality of the resulting image. Fortunately this problem has been extensively studied in the literature. It is possible to use, e.g. Poisson image editing [36] or other similar approaches [37,38] to alleviate such degradations.

3.4. Color prediction

In this section the aim is to estimate a color-to-region assignment for each frame in the original gray-scale sequence. We first assume that several frames have been already colorized. It means that each region has associated one label from the user-defined palette or is labeled as a background layer. Following this assumption it is possible to automatically retrieve the best color source from a pool of already colorized frames and propagate the color information between the corresponding regions. To do that we exploit scale, rotation and translation invariant pose alignment scheme based on the *log-polar phase correlation* [39]. This approach allows us to perform unsupervised colorization when the animation sequence consists of a small number of looped or randomly used phases.

3.4.1. Computer assisted auto-coloring

However, our challenge is also to speed up colorization of frames which have not yet been stored in the database. This type of problem has been also studied in the literature and is known as *computer assisted auto-coloring* [40–43]. Authors of these works attempt to automate *cel painting*, a more challenging variant of colorization where the gray-scale intensity cue is not available. They suggest to use region shape, size, position and topology to estimate the region-to-region correspondences. However, prediction performance is usually poor by the reason that such features are not robust to occlusions and topology variations imposed by a virtual depth commonly used in cartoon animations.

3.4.2. Intensity-based prediction

In contrast to cel painting in our case it is possible to exploit a more stable feature: *gray-scale intensity*. If we simply compare source and target regions in terms of their intensity medians, it is possible to propagate color labels between the regions which have the minimal median difference. However, like in luminance keying this straightforward approach suffers from wrong prediction when different colors are assigned to regions that have the same intensity median or when luminance fluctuation and image vignetting cause global and local intensity shifts in the original gray-scale image.

3.4.3. Patch-pasting

To overcome limitations of pure intensity-based approach we previously introduced color prediction scheme that exploits *patch-pasting* [44]. In this method dominant curvature points and junctions (see Fig. 10) are located using hierarchical *Kanade–Lucas–Tomasi* feature extractor [45]. Each feature is then represented by a set of square patches which cover local neighborhood of this feature in a specified number of orientations. Such representation allows us to retrieve for each target feature the best matching patch from the source image by searching over all possible source–target pairs in all possible orientations. Then a small rectangular area around each corresponding feature is extracted from the source color buffer and then corresponding color information is pasted on the proper position in the target color buffer (see Fig. 10). To avoid patch overlapping and to handle occlusions we exploit so called *quality buffer* [44] which is similar to well-known *z-buffer*, however, in place of depth it uses local matching quality to discard dissimilar portions of patches from pasting. Finally for each

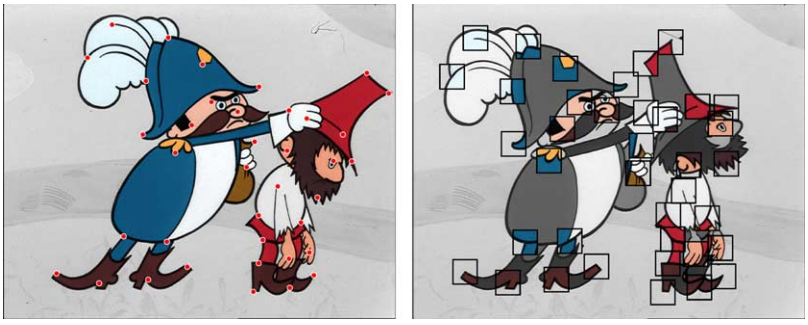

Fig. 10. Patch-pasting—yet colorized source image with detected features (left), target frame after the patch-pasting where corresponding source color patches have been used (right).

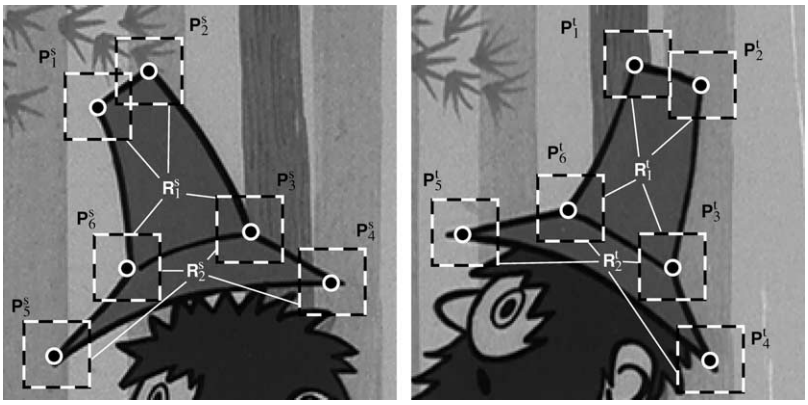

Fig. 11. Two sets of associated patches in the source (left) and target (right) frame.

target region the most frequently used color label is selected.

Prediction performance of patch-pasting is far limited by the number of local structural correspondences. When the source frame contains a few similar features, it is usually not possible to determine which of them corresponds to a similar feature in the target frame. To discriminate such ambiguity we suggested to use *probabilistic relaxation* scheme [44] which utilizes histogram of pasted color labels in the target region as a prior for color assignment. Posterior probability is then computed by iterating over each regions using color neighborhood model from the source frame. Unfortunately histograms of used color labels sometimes do not provide sufficient priors and consequently the correction after probabilistic relaxation brings no advantage in contrast to pure patch-pasting.

3.4.4. Color prediction scheme

In this section we introduce our novel color prediction scheme that is more robust in searching corresponding features than our previous approach. The substantial difference as compared with global matching scheme proposed in [44] consists in the use of region-to-region correspondences as a prior information which allows us to reduce the amount of false positives during the following patch-pasting phase.

Our novel method works as follows: we extract two sets of features (F^s and F^t) using *Kanade–Lucas–Tomasi* feature extractor [45] to cover important curvature points and junctions in the target and source frame. Each source $\mathrm{F}_k^s \in \mathrm{F}_s$ and target $F_l^t \in F^t$ feature is represented by a square patch P_k^s and P_l^t respectively that covers a small neighborhood around a feature point. For each source $R_i^s \in \Re^s$ and target $R_j^t \in \Re^t$ region we form a set of associated patches: $\mathcal{R}_i^s = \{P_k^s : P_k^s \cap R_i^s \neq \varnothing\}$ and $\mathcal{R}_j^t = \{P_l^t : P_l^t \cap R_j^t \neq \varnothing\}$, respectively, where $P \cap R \neq \varnothing$ denotes non-empty spatial intersection between patch P and region R (see Fig. 11). Afterwards it is possible to estimate the structural similarity of the target R_j^t and source region R_i^s using the best possible mapping between the sets of associated patches $\mathcal{R}_i^s$ and $\mathcal{R}_j^t$:

$$\mathrm{Sim}(R_i^s, R_j^t) = |\mathcal{R}_i^s| \cdot \left(\sum_{P_k^s \in \mathcal{R}_i^s} \min_{P_l^t \in \mathcal{R}_j^t} \mathrm{Diff}\,(P_k^s, P_l^t) \right)^{-1}. \tag{8}$$

To express prior probability that the region R_i^s corresponds to R_j^t (denoted $R_i^s \to R_j^t$) we normalize structural similarity $\mathrm{Sim}(R_i^s, R_j^t)$ as follows:

$$\mathbf{P}_0(R_i^s \to R_j^t) = \mathrm{Sim}(R_i^s, R_j^t) \cdot \left(\sum_{R_k^s \in \Re^s} \mathrm{Sim}(R_k^s, R_j^t) \right)^{-1} \tag{9}$$

Then a posterior probability is estimated using relaxation scheme inspired by [46]. In this approach posterior is refined iteratively using the following equation

$$\mathbf{P}_{n+1}(R_i^s \to R_j^t) = \frac{\mathbf{P}_n(R_i^s \to R_j^t) \cdot \mathbf{Q}_n(R_i^s \to R_j^t)}{\sum_{R_k^s \in \Re^s} \mathbf{P}_n(R_k^s \to R_j^t) \cdot \mathbf{Q}_n(R_k^s \to R_j^t)}, \tag{10}$$

where

$$\mathbf{Q}_n(R_i^s \to R_j^t) = \frac{1}{|\mathcal{N}_i^s|} \times \sum_{R_k^s \in \mathcal{N}_i^s} \sum_{R_l^t \in \mathcal{N}_j^t} \mathbf{P}_n(R_k^s \to R_l^t) \cdot \mathbf{P}_n(R_i^s \to R_j^t) \tag{11}$$

is the region compatibility function which expresses how feasible is the assignment $R_i^s \to R_j^t$ in the context of local neighborhood[4] $\mathcal{N}_i^s$ of the region R_i^s and $\mathcal{N}_j^t$ of the region R_j^t. After a few iterations of (10) when a relative difference between the prior $\mathbf{P}_n$ and posterior $\mathbf{P}_{n+1}$ falls under a specified limit, we compute *MAP* solution by assigning the most probable source region to each target region.

Now it is possible to transfer color labels directly between corresponding regions. However, the problem will arise when the user assigns two different colors to regions that have similar intensities and share a few associated

[4] Two regions are in neighborhood if and only if they share at least one associated patch.

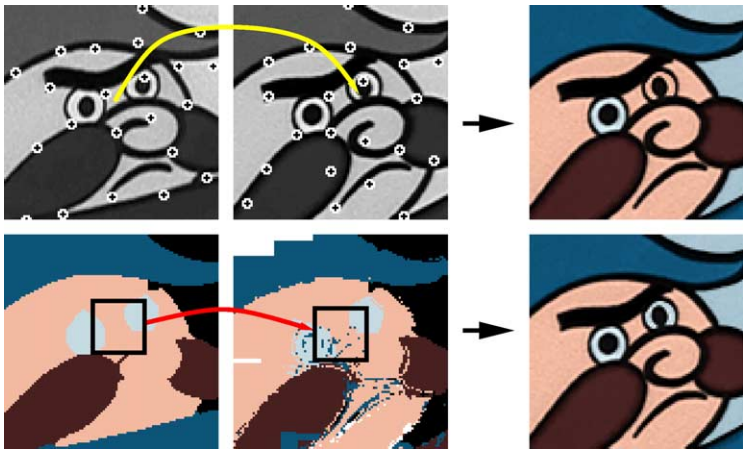

Fig. 12. Transferring color information—locally inconsistent region correspondences may occur when several regions have similar gray-scale intensity and share most of features (top). We remove this misclassification using patch-pasting where a more precise pixel-to-pixel color transfer is performed (bottom).

patches (see Fig. 12). In this case it is not possible to provide a good discriminative power using (8). Although in some special cases *MAP* solution removes such ambiguity, generally it still produces erroneous results.

To overcome this problem we utilize patch-pasting. We first sort region-to-region correspondences by non-decreasing structural similarity (8) and in this order we perform patch-pasting between the sets of associated patches using the same approach as in [44]. Finally for each target region we compute the most frequently used color.

The main advantage of patch-pasting is that it allows to transfer color information pixel-by-pixel between the corresponding features. As we stated above this is useful especially when two regions share a lot of associated patches (see Fig. 12). In this case, it does not matter whether region correspondences are locally inconsistent because they determine the same spatial context for more precise patch-pasting.

3.4.5. Implementation issues

An issue is the implementation of Diff(A,B) in (8). It stands for the metric to compare patches A and B in terms of their structural similarity. A good prediction performance is reached when the selected metric is invariant to some subset of local and global deformations. This is classical problem of early vision and it has been extensively studied in a literature (for survey see [47]). Authors usually trade off computational complexity and accuracy under a specified deformation model. In our case we assume only rigid deformations by the reason that global freeform changes in cartoons can be locally approximated using translation, rotation and scale. We additionally omit local scale changes because in cartoons size of objects and camera field-of-view remain usually static during the animation. If not, it is usually possible to resample whole image to reach the same scale for all objects. Following these assumptions, we express translation and rotation invariant metric as follows:

$$\mathrm{Diff}(A,B) = \min_{[x_0,y_0,\alpha]} \sum_x \sum_y (A(x+x_0, y+y_0) - \mathrm{Rot}(B(x,y), \backslash\alpha))^2 \quad (12)$$

where Rot denotes bitmap rotation.

A time-consuming brute force approach to evaluation of (12) guarantees exact value however is not tractable for a real application. To speed up evaluation of (12) we exploit approximation which will return nearly exact value much faster. We perform several experiments with a number of approximation approaches namely: *log-polar phase correlation* [39], *gradient descent* [17], *approximate nearest neighbor* [48], *fast Fourier transform* [49] and *hierarchical block matching* [50]. The best ratio between prediction performance and computational overhead has been reached using *oriented hierarchical block matching*. In this approach each target feature is represented by a set of rotation patches that cover local neighborhood around the feature in a selected number of possible orientations. Hierarchical block matching over this set is used to approximate (12). In this step we also exploit winner-update strategy [51] to reduce the number of tested positions.

Furthermore it is also possible to speed up computation of (8) using the following approximation: for each target region we first pre-select a set of possible candidates that have similar intensity median. To accomplish this we use a single threshold that takes into account luminance fluctuation and image vignetting in the original sequence. Regions which fall above this threshold receive zero prior probability in (9). We also allow for the fact that some regions share one or more associated patches (see Fig. 11 and 12). In this case it is possible to exploit a simple look-up table in which yet computed structural differences (12) are stored. Anytime we want to compare source and target patch we simply look at this table and if we found the appropriate value we reuse it,

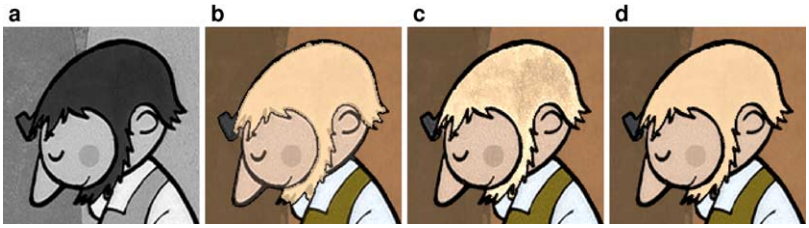

Fig. 13. Changing the brightness of colors: (a) the original gray-scale image, (b) intensity shifting produces step changes, (c) intensity scaling empowers a noise scattering, and (d) our approach.

otherwise we recall the oriented hierarchical block matching to refresh the look-up table.

3.5. Color image synthesis

In this section, we show how to apply color information on the gray-scale image. We also address a seamless composition of foreground and background layers, and a simple dust spot removal technique.

3.5.1. Color modulation

The brightness of the user-defined colors usually does not correspond to the intensity in the original gray-scale image. When one wants to apply a bright color on a dark region and vice versa, it is necessary either to shift or scale the original intensity to reach the required brightness. However the problem is that intensity shifting produce step changes and scaling empowers a noise scattering inside the region. Both artifacts are visually disturbing (see Fig. 13).

To produce visually acceptable results we use smooth transition between the shift and scale modulation. Only pixels that belong to outline anti-aliasing are scaled, remaining pixels vary color brightness via shift modulation. This technique yields following color modulation equation:

$$C = (1 - \alpha)(C_0 + I - \hat{I}) + \alpha C_0 I/\hat{I}, \tag{13}$$

where C_0 is the user-defined color, $\hat{I}$ represents the region intensity median, I is the intensity of the actual pixel in the original gray-scale image, and finally α is a spatially driven blending factor that allows us to smoothly combine shift and scale modulation. It is computed for each pixel using blurred version of original outlines (see Fig. 14).

3.5.2. Layer composition

We use similar technique to compose foreground and background layer seamlessly. In general, the original background layer is not homogeneous, hence it is necessary to compute the intensity median in a small neighborhood (e.g. 7×7) of the current pixel. For median computation we consider only pixels which do not belong to detected outlines and foreground regions. In contrast to (13) we then blend together the original color in a background layer with the scaled version of this color. To do that we use the following formula:

$$C = (1 - \alpha)C_b + \alpha C_b I/\hat{I}_b, \tag{14}$$

where C_b denotes the color of the actual pixel in the background layer, I is the intensity of the actual pixel in the original gray-scale image, and $\hat{I}_b$ is the intensity median of pixels from a local neighborhood of the current pixel. See Fig. 14 to overview results of proposed background and foreground color modulation.

3.5.3. Restoration

Thanks to the region homogeneity we are also able to simply detect and remove distortions which reside on the

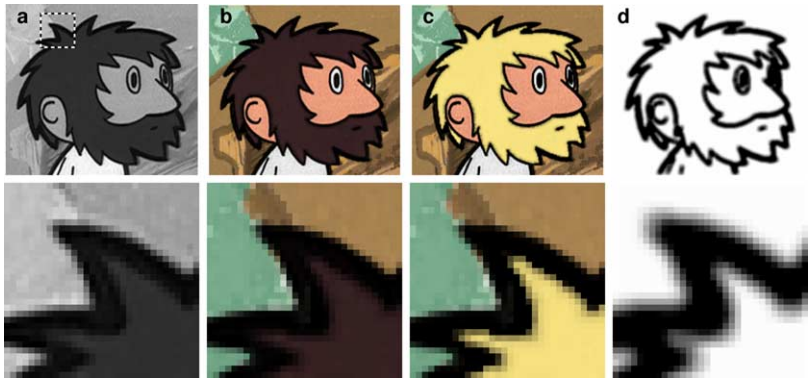

Fig. 14. Color modulation in the foreground layer and composition with the background layer: (a) the original gray-scale image, (b) an example of the final color image, (c) the case where a bright color has been applied on a dark region, and (d) alpha channel: white color denotes $\alpha=0$ and black $\alpha=1$.

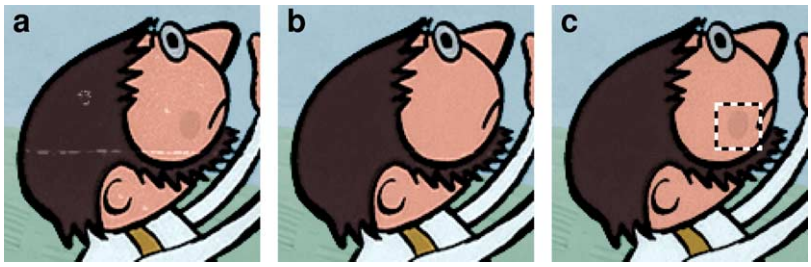

Fig. 15. Unsupervised dust spot removal: (a) the original color image, (b) black and white dust spot removal, and (c) white dust spot removal.

original celluloid. Using the intensity median $\hat{I}$ we compute the standard deviation of the pixel intensities σ_I in the current region. Pixels which exceed σ_I above the user-defined threshold are restored using a novel intensity taken as a random sample from the Gaussian distribution which has the following probability density function:

$$\mathrm{pdf}(x) = \exp(-(x-\hat{I})^2/\sigma_I^2). \tag{15}$$

This technique is not suitable for regions where some wanted inhomogeneities exist (like cheek in Fig. 15c). Usually we perform digital conversion from the original negative celluloid where dust spots are black and due to the digital inversion they become white in the resulting gray-scale image. If the local inhomogeneities are black it is possible to perform only white dust spot removal.

4. Experiments

In this section we discuss several experiments preformed on real cartoon images. They confirm usability of proposed framework both in the reduction of manual intervention and in the improvement of the final visual quality. Images used in these experiments are scanned in a resolution of 720×576 from the original celluloid negative of the Czech black-and-white cartoon *O loupežníku Rumcajsovi* which has been designed and produced by Radek Pilař in 1967.

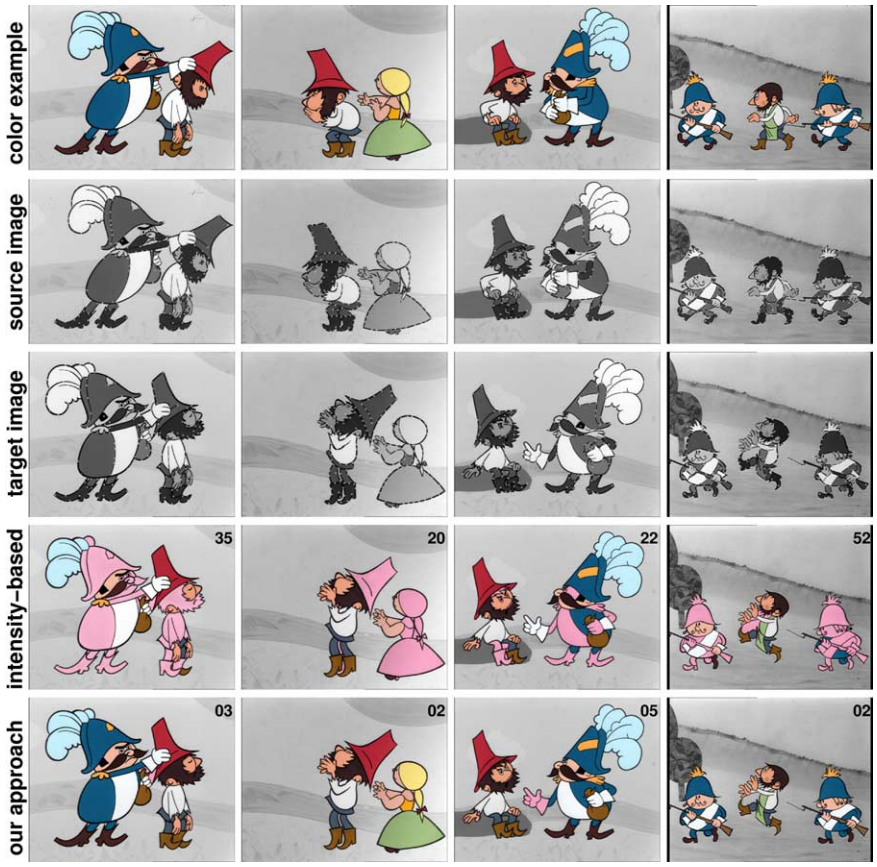

Fig. 16. Prediction performance (common case)—intensity-based prediction versus our novel color prediction scheme: pink regions denote prediction errors and numbers in images represent the overall number of prediction errors. White dots in the source and target image denote extracted features.

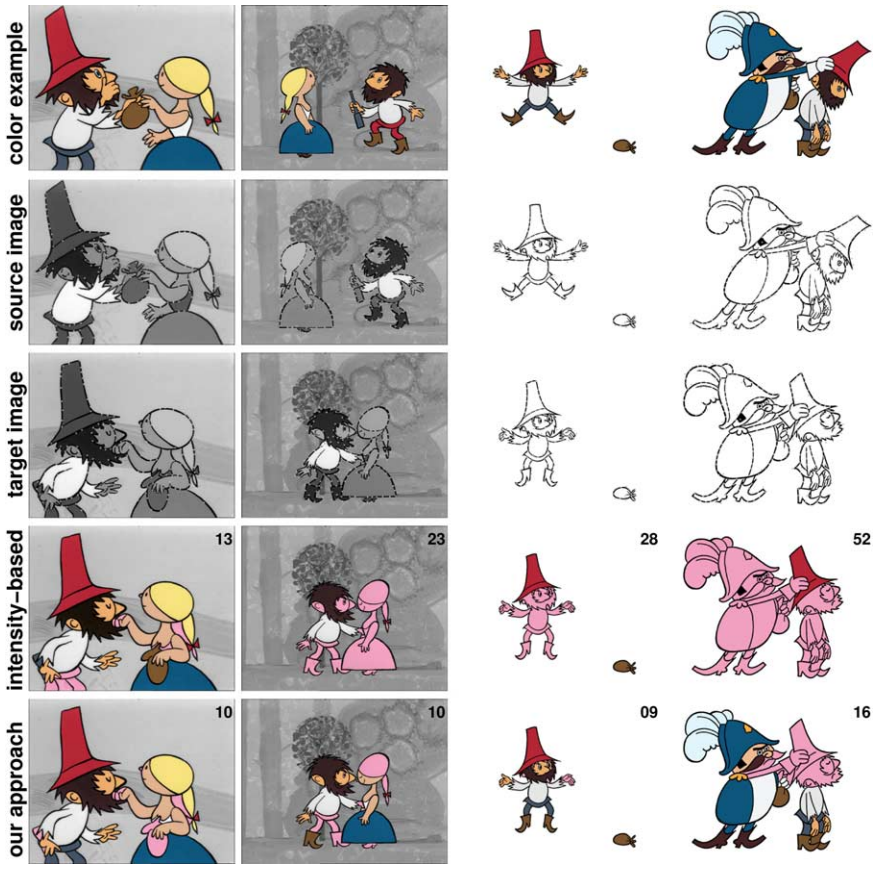

Fig. 17. Prediction performance (difficult case)—ordering is the same as in Fig. 16. Two difficult cases where many regions arise or change their position/shape (left side). See also how our method predicts color-to-region assignment for two cel paintings where no gray-scale information is available (right side).

4.1. Prediction performance

To verify the performance of our novel color prediction scheme (see Section 3.4) we realize two experiments. In the first one (Fig. 16) we show how our approach behaves in a typical situation. For this purpose we select four representative pairs of nearly consecutive frames from several cartoon animations. In the second experiment (Fig. 17) we demonstrate four difficult cases where our approaches tend to fail, however still produce helpful results.

In each experiment we first let the artist to prepare color examples. Then we extract 200 features for structural prediction where size of corresponding patches is set to 32×32 and the number of possible orientations for each patch is 32. We also set the intensity median threshold for pruning dissimilar regions to 16.

We compare our novel color prediction scheme with a naive intensity-based technique (see Section 3.4.2). In this comparison we consider the amount of user intervention needed to correct prediction errors. This amount is expressed by the number of prediction errors. Each error requires from the user to move mouse pointer over the erroneous region and place there a marker that has the correct color label. This action usually takes several seconds per error.

The numbers of errors confirm that in contrast to intensity-based approach our technique significantly reduces the amount of hand-driven intervention. Results are also good for cel painting where no gray-scale information is available. Nevertheless our method usually fails to predict correct labeling for some small or partially occluded regions. The fundamental problem is also how to assign appropriate color labels when a new structural pattern arises in the target frame. Due to these circumstances limited manual interaction is still required to produce errorless color sequences.

4.2. Visual quality

In this section we show how our color transferring technique outperforms previous approaches in the sense of visual quality. See Fig. 18 to compare our results with *Levin's* colorization framework [16]. Besides visual quality it is also interesting to compare the amount of user intervention needed to produce the final colorization.

The first problem is that artists usually prefer to prepare colors in a full color space by the reason that color

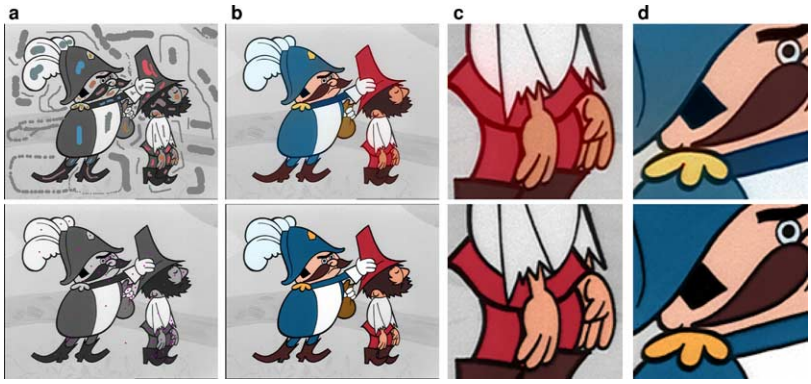

Fig. 18. Compare the final visual quality—colorization produced by *Levin's* framework (top) in contrast to our novel approach (bottom): (a) color markers, (b) foreground layer colorization, (c) and (d) detail views.

components provide only a limited variance in the final appearance. In Fig. 18, see e.g. soldier's epaulets where the artist assigns slightly darker color in contrast to intensity in the original image.

Similar situation is also visible on the soldier's hand where he wants to preserve the same color as for the uniform although the hand is slightly darker in the original image. The another problem in Fig. 18 is a low contrast of outlines.

Fig. 19. Selected sequences from cartoon *O loupežníku Rumcajsovi*, which has been semi-automatically colorized using our framework.

This artifact arises due to YUV color modulation which has been used in [16]. *Levin's* method also produces color bleeding over the sharp edges. This is due to least-square solution which does not preserve discontinuities.

5. Results

Proposed framework has been implemented as a stand-alone application in which the colorization pipeline consists of three independent stages. In the first stage we perform off-line pre-calculations that will prepare raw input data for interactive processing. This phase covers outline detection, image segmentation and structural similarity computations. It takes in average tens of seconds per frame on commodity hardware. In the second interactive stage the user colorizes the foreground layer frame-by-frame. He or she uses mouse or tablet to select pre-defined color labels from the user-defined palette and corrects labeling of regions when prediction scheme fails. During this step the camera motion estimation is also performed and visible parts of the background layer are extracted to form one big image. It is then colorized manually using standard image manipulation software. Finally in the third stage the color image synthesis phase follows. Here already colorized foreground and background layer are automatically putted together to produce the final color image.

Selected results are presented in Fig. 19. At average two trained artists were able to colorize one episode (roughly 10,000 frames) during one week (40 working hours) in contrast to previous approaches which takes more than two months to process the same amount of frames. In our framework the most of time has been spent on background colorization which is manual and requires a large amount of hand-driven intervention. In the case of foreground colorization the user interaction takes in average 5 s per frame. However the real processing speed strongly depends on the complexity of a given sequence.

6. Conclusion

In this paper we have made the following contributions. A novel colorization framework has been presented. Especially we addressed probabilistic reasoning scheme which takes into account region similarity and neighborhood relations to predict feasible matching context for patch-pasting. Further a novel labor saving σ-fitting algorithm has been presented to retrieve the optimal response of outline detector when the thickness of outlines is large. Finally we have introduced color modulation, composition and dust spot removal techniques to produce the final color images in a broadcast quality without additional user intervention. Our practical experiments on real cartoon images confirm that proposed framework allows one to produce high quality colorization of black-and-white cartoons within much shorter time frames as compared with previous approaches.

Acknowledgements

The authors would like to thank Jiří Bittner, Tomáš Pajdla, and Štěpán Hrbek for many useful comments. Images are published by the courtesy of © Vít Komrzí, Universal Production Partners and Lubomí r Celar, Digital Media Production. This work has been partly supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research program no. Y04/98:212300014 (Research in the area of information technologies and communications), and under the student research programs: FRVŠ-2004-2067, FRVŠ-2005-1170.

References

[1] L. Lenburg, The encyclopedia of animated cartoons Facts on File, New York, NY (1999).
[2] D.J. Bancroft, Advanced and economical telecine technology for global DTV production in: Broadcast Engineering Conference Proceedings (2000).
[3] A.C. Kokaram, Motion Picture Restoration: Digital Algorithm for Artefact Suppression in Degraded Motion Picture Film and Video, Springer, London, 1998.
[4] W. Markle, The development and application of colorization, SMPTE Journal (1984) 632–635.
[5] R. Cooper, Colorization and moral rights: should the united states adopt unified protection for artists?, Journalism Quarterly (Urbana, Illinois) 68 (1990) 465–473.
[6] F. Leibowitz, Movie colorization and the expression of mood, Journal of Aesthetics and Art Criticism (Cleveland, Ohio) 49 (4) (1991) 363–364.
[7] R.C. Gonzalez, R.E. Woods, Digital Image Processingsecond ed., Addison–Wesley Publishing, Reading, MA, 1987.
[8] T. Welsh, M. Ashikhmin, K. Mueller, Transferring color to greyscale images in: ACM SIGGRAPH 2002 Conference Proceedings (2002) pp. 277–280.
[9] E. Reinhard, M. Ashikhmin, B. Gooch, P. Shirley, Color transfer between images, IEEE Transactions on Computer Graphics and Applications 21 (5) (2001) 34–41.
[10] A. Hertzmann, C.E. Jacobs, N. Oliver, B. Curless, D.H. Salesin, Image analogies in: ACM SIGGRAPH 2001 Conference Proceedings (2001) pp. 327–340.
[11] W. Markle, B. Hunt, Coloring a black and white signal using motion detection, Canadian Patent No. CA 01291260 (1991).
[12] Z. Pan, Z. Dong, M. Zhang, A new algorithm for adding color to video or animation clips in: Proceedings of WSCG—International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (2004) pp. 515–519.
[13] T. Horiuchi, Estimation of color for gray-level image by probabilistic relaxation in: Proceedings of IEEE International Conference on Pattern Recognition (2002) pp. 867–870.
[14] J. Kittler, J. Illingworth, Relaxation labelling algorithms—a review, Image and Vision Computing 3 (1985) 206–216.

[15] T. Horiuchi, S. Hirano, Colorization algorithm for grayscale image by propagating seed pixels in: Proceedings of IEEE International Conference on Pattern Recognition (2003) pp. 457–460.
[16] A. Levin, D. Lischinski, Y. Weiss, Colorization using optimization in: ACM SIGGRAPH 2004 Conference Proceedings (2004) pp. 689–694.
[17] B.D. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision in: Proceedings of the International Joint Conference on Artificial Intelligence (1981) pp. 674–679.
[18] G. Sapiro, Inpainting the colors, IMA Preprint Series #1979, Institute for Mathematics and its Applications, University of Minnesota, 2004. http://www.ima.umn.edu/preprints/may2004/may2004.html
[19] G. Sapiro, Geometric Partial Differential Equations and Image Processing, Cambridge University Press, Cambridge, 2001, http://www.ece.umn.edu/users/guille/book.html
[20] L. Yatziv, G. Sapiro, Fast Image and Video Colorization using Chrominance Blending, IMA Preprint Series #2010, Institute for Mathematics and its Applications, University of Minnesota, 2004. http://www.ima.umn.edu/preprints/dec2004/dec2004.html
[21] V. Caselles, R. Kimmel, G. Sapiro, Geodesic active contours, International Journal of Computer Vision 22 (1) (1997) 61–79.
[22] T. Chen, Y. Wang, V. Schillings, C. Meinel, Grayscale image matting and colorization in: Proceedings of Asian Conference on Computer Vision (2004) pp. 1164–1169.
[23] Y.-Y. Chuang, B. Curless, D.H. Salesin, R. Szeliski, A Bayesian approach to digital matting in: Processings of IEEE Conference on Computer Vision and Pattern Recognition (2001) pp. 264–271.
[24] D. Sýkora, J. Buriánek, J. Žára, Segmentation of black and white cartoons in: Proceedings of Spring Conference on Computer Graphics (2003) pp. 245–254.
[25] D. Marr, E.C. Hildreth, Theory of edge detection in: Proceedings of Royal Society vol. B207 (1980) pp. 187–217.
[26] A. Rosenfeld, A.C. Kak, Digital Picture Processing vol. 1, Academic Press, Orlando, 1982.
[27] H.-D. Cheng, Y. Sun, A hierarchical approach to color image segmentation using homogeneity, IEEE Transactions on Image Processing 9 (12) (2000) 2071–2082.
[28] A.P. Witkin, Scale space filtering in: A.P. Pentland (Ed.), From Pixels to Predicates: Recent Advances in Computational and Robot Vision (1986) pp. 5–19.
[29] J.J. Clark, Authenticating edges produced by zero-crossing algorithms, IEEE Transactions on Pattern Analysis and Machine Intelligence 11 (1) (1989) 43–57.
[30] G.E. Sotak, K.L. Boyer, The Laplacian-of-Gaussian kernel: a formal analysis and design procedure for fast, accurate convolution and full-frame output, Computer Vision, Graphics, and Image Processing 48 (2) (1989) 147–189.
[31] J.S. Chen, A. Huertas, G. Medioni, Fast convolution with Laplacian-of-Gaussian masks, IEEE Transactions on Pattern Analysis and Machine Intelligence 9 (4) (1987) 584–590.
[32] D. King, Implementation of the Marr–Hildreth theory of edge detection, Tech. Rep. ISG-102, The University of Southern California, 1982.
[33] A. Huertas, G. Medioni, Detection of intensity changes with subpixel accuracy using Laplacian–Gaussian masks, IEEE Transactions on Pattern Analysis and Machine Intelligence 8 (6) (1986) 651–664.
[34] J. Serra,fourth ed. Image Analysis and Mathematical Morphology vol. 1, Academic Press, Oval Road, 1993.
[35] J.-M. Odobez, P. Bouthemy, Robust multiresolution estimation of parametric motion models, Journal of Visual Communication and Image Representation 6 (4) (1995) 348–365.
[36] P. Pérez, M. Gangnet, A. Blake, Poisson image editing in: ACM SIGGRAPH 2003 Conference Proceedings (2003) pp. 313–318.
[37] A. Levin, A. Zomet, S. Peleg, Y. Weiss, Seamless image stitching in the gradient domain in: Proceedings of European Conference on Computer Vision (2004) pp. 377–389.
[38] J.R. Burt, E.H. Adelson, A multiresolution spline with application to image mosaics, ACM Transactions on Graphics 2 (4) (1983) 217–236.
[39] B.S. Reddy, B.N. Chatterji, An FFT-based technique for translation, rotation and scale-invariant image registration, IEEE Transactions on Computers 5 (8) (1996) 1266–1271.
[40] J.S. Madeira, A. Stork, M.H. Grob, An approach to computer-supported cartooning, The Visual Computer 12 (1996) 1–17.
[41] C.W. Chang, S.Y. Lee, Automatic cel painting in computer-assisted cartoon production using similarity recognition, The Journal of Visualization and Computer Animation 8 (1997) 165–185.
[42] H.S. Seah, T. Feng, Computer-assisted coloring by matching line drawings, The Visual Computer 16 (2000) 289–304.
[43] J. Qiu, H.S. Seah, F. Tian, Q. Chen, K. Melikhov, Computer-assisted auto coloring by region matching in: Proceedings of Pacific Conference on Computer Graphics and Applications (2003) pp. 175–185.
[44] D. Sýkora, J. Buriánek, J. Žára, Unsupervised colorization of black-and-white cartoons in: Proceedings of the Third International Symposium on Non-photorealistic Animation and Rendering (2004) pp. 121–127.
[45] C. Tomasi, T. Kanade, Shape and motion from image streams: a factorization method. Part 2. Detection and tracking of point features, Tech. Rep. CMU-CS-91-132, Carnegie Mellon University, School of Computer Science, 1991.
[46] A. Ahmadyfard, J. Kittler, Region-based object recognition: pruning multiple representations and hypotheses in: Proceedings of British Machine Vision Conference (2000) pp. 745–754.
[47] B. Zitová, J. Flusser, Image registration methods: a survey, Image and Vision Computing 21 (2003) 977–1000.
[48] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, A.Y. Wu, An optimal algorithm for approximate nearest neighbor searching in fixed dimensions, Journal of the ACM 45 (6) (1998) 891–923.
[49] S.L. Kilthau, M.S. Drew, T. Möller, Full search content independent block matching based on the fast Fourier transform in: Proceedings of IEEE International Conference on Image Processing (2002) pp. 669–672.
[50] K.M. Nam, J.-S. Kim, R.-H. Park, Y.S. Shim, A fast hierarchical motion vector estimation algorithm using mean pyramid, IEEE Transactions on Circuits and Systems for Video Technology 5 (4) (1995) 344–351.
[51] Y.-S. Chen, Y.-P. Hung, C.-S. Fuh, Fast block matching algorithm based on the winner-update strategy, IEEE Transactions On Image Processing 10 (8) (2001) 1212–1222.