Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering

Computer Assisted Analysis of Classical Cartoon Animations

by

Daniel Sýkora

A doctoral thesis submitted to
the Faculty of Electrical Engineering, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Doctor.

Ph.D. Programme: Electrical Engineering and Information Technology
Branch of study: Computer Science and Engineering

September 2006

Thesis Supervisor:
Doc. Ing. Jiří Žára, CSc.
Department of Computer Science and Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague
Karlovo náměstí 13
121 35 Prague 2
Czech Republic

Abstract and contributions

In this thesis a novel approach to analysis and processing of classical cartoon animations is presented. It focuses on a popular animation technique where each frame is created as a planar composition of two visually distinct layers: static textural background and dynamic homogenous foreground. The aim of the proposed framework is (1) to precisely detach these two layers to reach the state before the final composition and (2) to estimate structural correspondences between animation frames and homogenous regions in the foreground layer. Such analysis allows to partially understand the `2.5D` structure of classical cartoon animations that is crucial for numerous applications including semi-automatic colorization, example-based synthesis, and video compression.

The main contributions of the thesis:

1. Unsupervised sub-pixel accurate outline detection and image segmentation algorithms suitable for cartoon images where outlines represent region boundaries.
2. Hierarchical estimation of frame-to-frame and region-to-region correspondences based on local structural similarity and neighborhood relations.
3. Example-based approach to semi-automatic colorization of black-and-white cartoon animations dramatically reducing the amount of manual interventions.
4. Example-based synthesis of cartoon animations using an intuitive scribble-based interface allowing to quickly sketch new cartoon characters and poses in the style of masters.
5. Efficient video compression technique for classical cartoon animations based on a hybrid encoding scheme providing state-of-the-art visual quality for low encoding bit-rates.

Keywords:

cartoon animation, image processing, image registration, image segmentation, edge detection, colorization, image restoration, example-based image synthesis, video compression.

Acknowledgements

First of all, I would like to express special thanks to my supervisor *Jiří Žára* whose constant help and insightful guidance make my dissertation possible. Thanks must also go to my colleague *Jan Buriánek* who stay behind the main idea of this thesis and whose overwhelming insight to the area of image processing pushed me toward in the beginning of this work.

Another special appreciation goes to *Vít Komrzí* from *Universal Production Partners* and *Lubomír Celar* from *Digital Media Production* for providing me a financial support in the beginning of this research and allowing me to make the internal results publicly available including all experimental data. At this point it is necessary to remember great artists *Jiří Šabata*, *Helena Keslová* and *Jiří Štamfest* who stay behind the final realization of the cartoon colorization project from which my own research originates.

I am also grateful to *Tomáš Pajdla*, *Jiří Bittner* and *Štěpán Hrbek* who provide me with detailed and illuminating comments on an early version of the draft that considerably improved the quality of the thesis.

My work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under research programs No. `Y04/98:212300014` and `MSM-6840770014` (Research in the area of information technologies and communications) and under student research programs: `FRVŠ-2004-2067` and `FRVŠ-2005-1170`. I would like to thank to staff of our department and especially to *Josef Kolář* and *Pavel Tvrdík* for taking care of my financial support and providing a pleasant environment for my research.

This thesis would not exist without devoted support and patience of my great family: wife *Pavla*, daughter *Štěpánka* and sons *Matyáš* and *Mikuláš*. Now, when the work is done, I have to compensate their starvation for my attention.

Dedication

To my wife Pavla and children Štěpánka, Matyáš and Mikuláš.

Contents

List of Figures

1 Introduction

This thesis introduces a novel approach to computer assisted analysis of classical cartoon animations. The aim is to understand `2.5D` structure of the scene by disassembling the original composition of layers to a set of logical parts and estimating their mutual correspondences in the animation. Such analysis allows to reduce a large amount of manual intervention in various renewal tasks including colorization, color restoration, outline enhancement, noise suppression, dust spots removal, synthesis of new animations in the traditional drawing style, and also leads to an efficient video compression scheme for classical cartoon animations.

Figure 1.1: **Various examples of traditional cartoon animations.**

1.1 Motivation

A long history of traditional cartoon animation provides a respectable amount of artistically advanced works [10, 101] (see Figure 1.1). Generations of children and also adults enjoy typical characters, drawing/animation styles, scenarios, and soundtracks. They wish to watch favorite stories again and again in the best possible visual quality. However, lifetime of former archival formats (such as celluloid negative) is strongly limited. Variety of chemical and mechanical degradations reduce the level of visual quality and can also completely destroy the original

artwork. When a classical cartoon is broadcasted, younger audience usually do not understand why the picture on the screen is so noisy, dilute or even completely black-and-white. They suppose that this is something like technical failure and often prefer modern visually more attractive cartoon animations despite of their overall artistic quality.

A common way how to rescue and disseminate cultural heritage of traditional works for future generations is to transfer motion picture from the original celluloid negative to a digital video tape using *telecine* [5] and then perform careful computer assisted restoration [152]. This process usually consists of several independent tasks such as noise suppression, dust spots and vignetting removal, outline enhancement, color restoration or full colorization when the original cartoon was shot in black-and-white. Unfortunately, these tasks often require laborious manual intervention even when professional post-production tools are available. Consequently, the restoration process is not cost effective and usually the original material is broadcasted in a raw form without restoration.

The main drawback of such common workflow is that professional post-production tools are primarily designed for real-world videos and usually do not take into account much simpler structure of classical cartoon animations. A typical cartoon frame is usually created as a planar composition of two layers: background and foreground. The background layer is static textural image and the dynamic foreground layer consists of several homogenous regions enclosed by well visible outlines (see Figure 1.1). This thesis describes a novel cartoon analysis framework that utilizes such a priori knowledge in order to reduce the amount of manual interventions in various applications and also to design new cartoon oriented video compression algorithm.

1.2 Cartoon analysis framework

The aim of the proposed cartoon analysis framework is to detach original layers and estimate structural correspondences in the foreground layer. To accomplish this goal first an unsupervised image segmentation algorithm is used to separate the original input frame (Figure 1.2a) into a set of regions. During this step a robust outline detector locates boundaries of regions (Figure 1.2b) and then region area size is used to roughly estimate whether they belongs to the background or to the foreground layer. In the next phase foreground layer is converted from raster to vector representation (Figure 1.2d) and all visible fragments of background layer are registered and stitched together to form one image (Figure 1.2c) that is immediately reused to refine the foreground/background classification. Finally patch-based structural similarity and neighborhood relations are exploited in order to estimate correspondences between animation frames and foreground regions (Figure 1.2e). To illustrate usability of such analysis three different applications have been developed.

1.2.1 Colorization and restoration

The first important application of the proposed framework is *colorization* – a challenging form of movie restoration that brings new color information into gray-scale images to enhance their visual appeal. It has been originally developed in 1970 by *Wilson Markle* [112] who pioneered color restoration of classical feature films. Despite of recent advances in color transferring techniques [102, 190] the problem is still very tedious and time consuming.

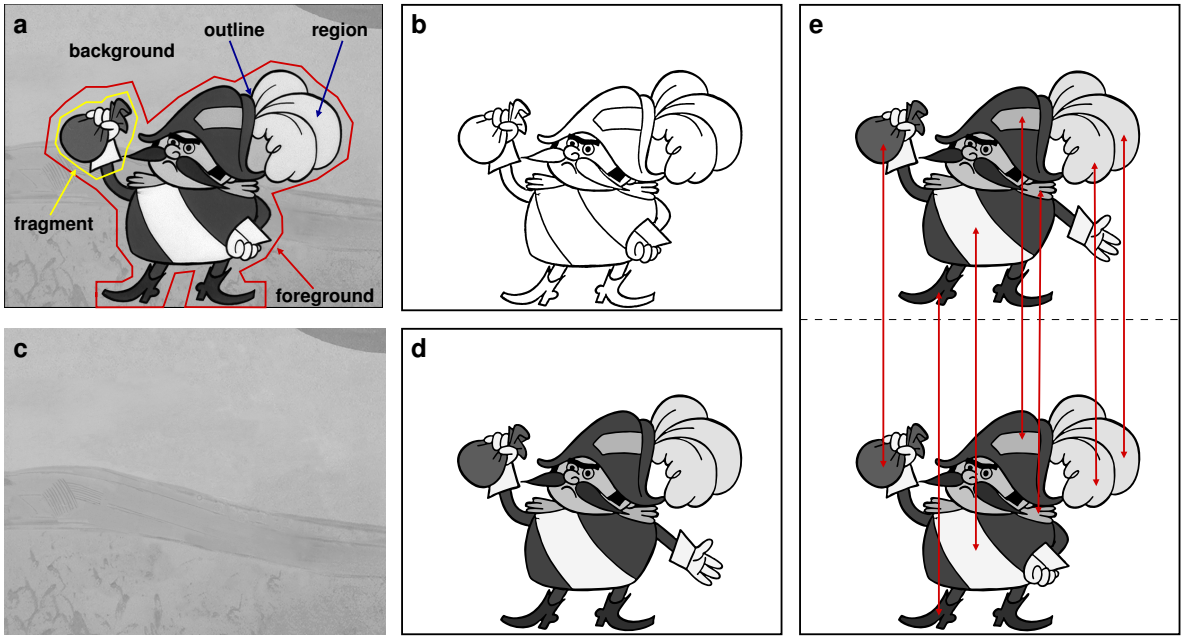

Figure 1.2: **An overview of the proposed cartoon analysis framework**: (a) one animation frame from the input sequence, (b) outline detection, (c) reconstructed background layer, (d) vectorized foreground layer, (e) retrieval of correspondences.

Unfortunately, colorization became unpopular in past decades mainly due to belief that it defiles the original artwork [42] since black-and-white movies require different lighting conditions. However, in the case of cartoon animation, the situation is different. Presence of color can significantly improve the visual attraction as well as artistic impression [100] therefore it usually appeals less controversy compared to classical feature films (see Figure 1.3).

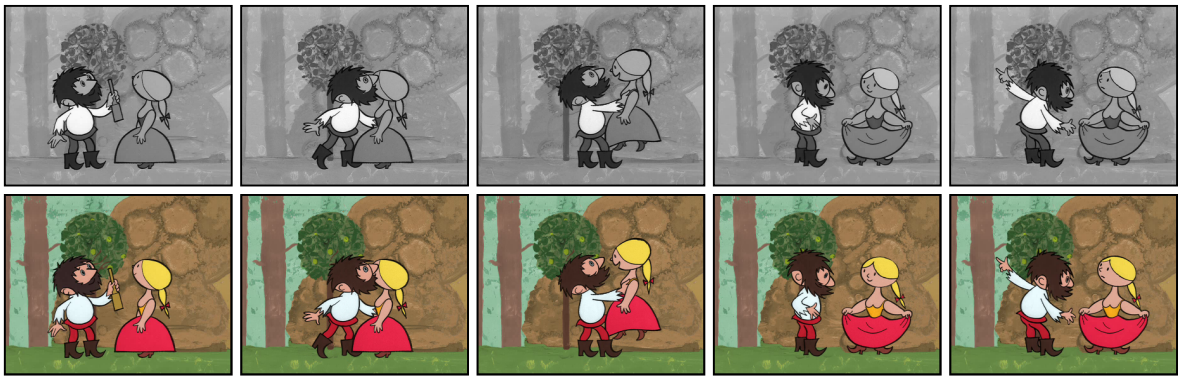

Figure 1.3: **Semi-automatic colorization of classical black-and-white cartoon animation.**

Since the proposed framework performs precise segmentation and also estimates correspondences between regions, high-quality colorization or color restoration of the foreground layer become much easier as compared to previous approaches. Moreover, reconstructed background can be colorized at one snap and then reused in a whole animation. Including the

assumption of region homogeneity it is also possible to simplify detection and suppression of typical aged movie artifacts such as additive noise, dust spots, low contrast of outlines, frame vignetting, and luminance fluctuation.

1.2.2 Cartoon-by-example

In computer assisted cartooning skilled artists first prepare a set of fragments from which more complex scenarios and animations are composed [48]. However, the problem arises when sources are hand-made cartoon drawings (a.k.a. master frames). In such a case stand-alone fragments are no longer available. Similar situation also occurs when one wants to create new stories in the style of masters using only fragments of the original artwork. The common question is how to extract and seamlessly compose fragments from ready-made compositions. Using standard image manipulation tools this task is tedious and time consuming.

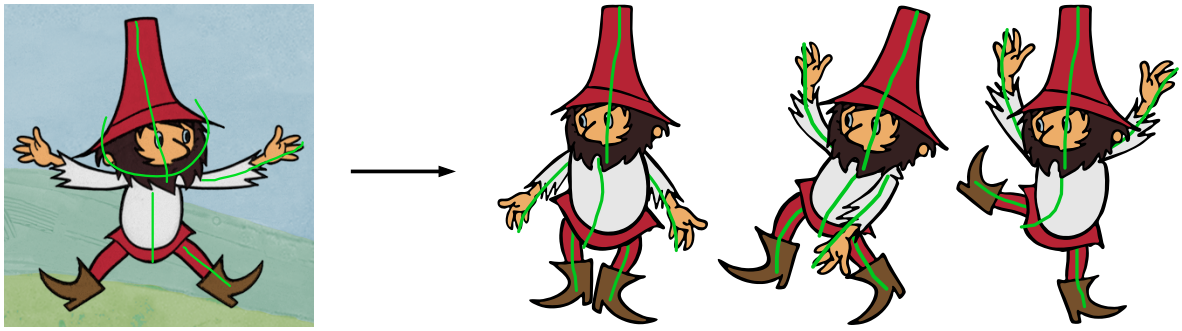

Figure 1.4: **An intuitive scribble-based interface for designing new poses from fragments of the original artwork.**

Framework described in this thesis allows to reduce burden connected with fragment extraction and composition. Moreover, for ease of manipulation an intuitive scribble-based interface is proposed. It allows the user to simply select an interesting part in the original drawing and then adjust it in a new composition using several control scribbles (see Figure 1.4). Proposed approach is suitable both for experienced artists and unskilled users (e.g. children) who wish to create new stories in the style of masters. High-quality cartoon drawings can be produced from traditional sources with much little effort as opposed to standard approaches.

1.2.3 Video compression

Standard approaches to video compression including `MPEG-2` specification [118, 186] assume that strong spatial and temporal discontinuities are rare in natural image sequences. They exploit *discrete cosine transform* (`DCT`) [3] or *discrete wavelet transform* (`DWT`) [157] to appropriately rearrange the image energy so that it can be further quantized and encoded efficiently. However, due to block decomposition and quantization, blocking and ringing artifacts arise when `DCT` or `DWT` is applied to classical cartoon animations where each animation frame consists of many sharp edges.

The proposed framework is used to recover a more natural data representation suitable for classical cartoon animations. Based on the analysis of the input sequence the background

Figure 1.5: **Detail views compare the visual quality of image sequence compressed by DivX** (top) **and by the proposed cartoon-oriented video codec** (bottom) **using the same encoding bit-rate.**

layer is stored as a single image and the foreground layer as a sequence of vector images. Such a hybrid form can be encoded more compactly as compared to pure `2D DCT` or `DWT`-based approaches (see Figure 1.5) therefore it provides superior visual quality for equivalent coding bit-rates. Moreover, real-time hardware accelerated playback allowing partial spatial scalability is possible thanks to widely available modern graphic cards.

1.3 Related work

The topic of this thesis is closely related to many research areas. This section briefly highlights some important pointers to computer-assisted cartoon animation, colorization, restoration, example-based image synthesis, and video compression. For detailed treatment and more specific references see also related sections in the main part of this thesis.

1.3.1 Computer-assisted cartoon animation

In 1970's *Nestor Burtnyk* together with *Marceli Wein* [20, 21, 22] pioneered the idea to simplify tedious and time consuming tasks in classical cartoon animation by utilizing computer assistance. Since then computer-assisted cartoon animation become popular research area and is still very active. In preceding decades various semi-automatic *animation systems* have been developed [103, 26, 163, 174, 107, 45, 48, 138]. However, these systems were mainly focused on image synthesis. Their authors do not attempt to perform any deeper analysis such as *correspondence retrieval*. This fundamental problem has been studied later in connection with *computer-assisted auto coloring* [111, 28, 154, 132, 134, 135, 133] where region correspondences are used to predict transfer of color markers. Correspondences are also important in *computer-assisted inbetweening* [51, 93, 115] to register consecutive key-frames or in *motion capturing* [15, 176] where the original motion is reused for new 2D drawings and 3D models.

Besides retrieval of region correspondences research in past decades focused on many other important tasks in computer-assisted cartoon animation such as *thinning* and *vectorization* of line drawings [194, 197, 31], synthesis of multi-perspective *panoramic backgrounds* [188], shape sensitive *texture mapping* [43], adding pseudo 3D *shadows* [129], generating realistic 3D-like *illumination* [83] and *smoke* effects [155], creating simple cartoon animation from *real-world video* [1, 178], frame *reordering* and *warping* to create much longer animation sequences from existing footage [85, 63], *sketching motion* of cartoon character [168], and also several advanced *shape manipulation* techniques [77, 151].

1.3.2 Colorization and restoration

Computer-assisted colorization has been studied since 1970 [112]; however, research in this field did not become popular until quite recently. The problem itself is strongly ill-posed and thus various semi-automatic approaches have been developed to estimate color-to-intensity assignment exploiting partial user intervention.

Historically, *motion estimation* was the first published and patented approach to semi-automatic colorization [113, 126, 72]. The basic assumption here is that key-frames are colorized manually and the color information can be propagated to the rest of the sequence using motion estimation. However, usually a lot of frames have to be colorized or corrected manually since rapid changes in the image sequence cause incorrect estimation of motion vectors.

Another popular method to colorization known as *colorization-by-example* [184, 81, 191, 79, 106] transfers chromatic information from selected source color image to the gray-scale target using feature-based classification. Feature vector is measured from gray-scale information in a local neighborhood of each target pixel (or inside the region [164]) and then classified to match similar sample in the source image. From the selected sample a chromatic information is then extracted and applied on the corresponding location in the target image. Unfortunately, such approaches work only when similar gray-scale features represent similar color information in the target and source images. Only tedious manual pre-classification [32] can alleviate such limitation.

The last group of techniques can be jointly called *color inpainting* [70, 102, 74, 131, 149, 190]. Here the aim is to propagate color information directly from the user-specified seed pixels to the rest of the image (or sequence of images) by exploiting the assumption that homogeneity in the gray-scale domain usually indicates homogeneity in the color domain and vice versa. However, in this technique the problem is that typically many seed pixels have to be placed carefully and also the selection of hue and saturation is very laborious.

Although in this thesis colorization framework is mainly used for enhancement of black-and-white cartoon animations, the process itself can be also exploited for restoration of color footage. Research in this area is active since then the digital image processing become applicable and cost effective. The main research interests include detection and removal of common artifacts such as additive noise [17], dust and dirt sparkles[92, 167], line scratches [84, 11], missing frames [56], mold caused color/intensity variations [166, 27] and others (for comprehensive overview see [91, 152]).

1.3.3 Example-based synthesis

One of the aims of this thesis is to allow quick creation of new character poses preserving the original drawing style. This challenging task has still not been solved sufficiently, nevertheless several initial approaches exist. *Hertzmann et al.* pioneered pixel-by-pixel transfer of style [67] by matching local image statistics. From this technique colorization-by-example originates [184]. Similar approach has been also used in 1D [57, 68] to transfer particular drawing styles to line drawings. *Jodoin et al.* [82] presented hatching-by-example technique that combines ideas of texture synthesis in order to stylize user-defined curves. Finally *Drori et al.* [44] proposed approach where the original image is first decomposed into a couple of small fragments, and then stitched together to form new image that follows several constraints. The common problem of these approaches is that the transfer is only local. This is usually insufficient since typically salient global features are much more important to recognize particular drawing style.

Several authors attempt to overcome this key limitation by asking an artist to prepare a set of stand-alone fragments that can be later reused in a variety of different ways [18, 29, 52]. In this thesis the assumption is that only the final composition exists and the original fragments are no longer available. To create truly new poses and characters, it is necessary to extract and seamlessly compose fragments of the original artwork. This is challenging task requiring extensive manual intervention. *Barrett* and *Cheney* [6] and later *Saund et al.* [150] proposed object-based image editing tools that can significantly reduce the amount of manual intervention, however, for classical cartoon images the fragment extraction process is still tedious and the final compositions contain various noticeable artifacts.

1.3.4 Video compression

Video compression research remains popular for several decades (see survey [139, 39]). The most of published approaches in this field were aimed to real-world video compression. As discussed in Section 1.2.3, the key assumption here is that large discontinuities are not so frequent therefore `DCT` or `DWT` can by used to efficiently compress difference images. However, when such transformations are applied to cartoon images with lots of sharp edges, ringing and blocking artifacts may arise due to quantization errors even when standard `DVD` encoding bit-rates are used.

To address this issue, various post-processing techniques have been developed [189, 47]. However, they usually require additional processing time that is not tractable for real-time playback. Another strategy is to completely avoid coding across strong edges, i.e. to partition the input image into a set of homogeneous regions and then process them separately [80, 160, 96, 179, 7]. Unfortunately, due to inaccuracy of segmentation, these techniques require typically much higher bit-rates to preserve compelling visual quality.

Kwatra and *Rosignac* [95] presented approach where each region is first represented as a 3D volume by sweeping its 2D shape through the time axis. Then this volume is triangulated and compressed using custom tailored mesh compression technique. *Concolato et al.* [41] shown how to embed existing cartoon animations (yet stored in some vector format such as *Flash*) into the `MPEG-4` framework [46] to reach lower bit-rates. However, since these approaches

assume heavily preprocessed input or directly vector representation, region extraction phase is not discussed and remains an open problem.

Recently, *Lee* and *Kassim* [98] proposed an approach that seems to be the first practically usable video codec for cartoon animations. It utilizes a novel hybrid wavelet-based encoding scheme where the sets of basis functions are called *wedgelets* and *beamlets*. Wedgelets are designed to represent step changes in color (boundaries of homogenous regions) whereas beamlets are suitable for thin outlines. Despite of an interesting encoding scheme, the central problem – how to precisely extract regions and outlines – has not been solved sufficiently. Authors only suggest to exploit `EDISON` library based on *mean-shift* segmentation algorithm [40] that is not very precise therefore the image quality can be notably reduced before the encoding scheme is applied.

1.4 Contributions of the thesis

This thesis introduces a novel cartoon analysis framework suitable for cel- or paper-based animation techniques where each frame is created as a planar composition of static textural background and dynamic homogenous foreground. The framework allows to pop-up the `2.5D` structure of the scene by extracting outlines, locating foreground regions, vectorizing foreground layers, and reconstructing the visible portion of background layer. On the basis of this decomposition structural similarity analysis is performed to estimate frame-to-frame and region-to-region correspondences in the foreground layer. To verify the usability of this framework three different applications have been developed: example-based colorization, example-based synthesis, and video compression.

The main contributions of the thesis:

1. Unsupervised *segmentation algorithm* exploiting robust sub-pixel accurate *outline detector* that utilizes negative response of the Laplacian of Gaussian filter with adaptive scale selection and outline classification [A.5, A.3, A.1].

2. Hierarchical estimation of structural *correspondences* between animation frames and homogenous regions using phase correlation, patch-based similarity and probabilistic reasoning over neighborhood relations [A.4, A.1].

3. Example-based *colorization* and *color restoration* using semi-automatic prediction of color-to-region assignment together with precise color modulation, temporal and spatial noise suppression, unsupervised dust-spot and vignetting removal [A.4, A.1].

4. Example-based drawing style preserving *synthesis* of cartoon animations using an intuitive scribble-based interface, unsupervised fragment extraction, and high-quality vectorization [A.3].

5. Efficient *video compression* technique for outline-based cartoon animations using hybrid coding scheme (adaptively sampled 1D DCT for region shapes and classical 2D DCT for reconstructed background) with hardware accelerated playback [A.2].

1.5 Organization of the thesis

The thesis is divided into six sections. This Section 1 introduces and motivates the proposed cartoon analysis framework and also includes a brief overview of related work and contributions. The following Section 2 represents the main body of the thesis. Here a novel cartoon analysis framework is described in detail together with implementation issues and experimental results. In the next three sections practical applications of the proposed framework are discussed: colorization and restoration (Section 3), cartoon-by-example (Section 4) and video compression (Section 5). Section 6 concludes the thesis and presents several new avenues for the future work.

2 Cartoon analysis framework

In this section a novel cartoon analysis framework is described. First a brief overview of the whole processing pipeline is presented and later, in successive sections, each step is described in more detail including optimization issues and experiments.

2.1 Framework overview

The proposed framework has been designed to process classical cartoon animations created by cel- or paper-based animation technique where each animation frame is composed of two planar layers: background and foreground. The key advantage here is that each layer has notably different visual appearance. The dynamic foreground consists of outlined homogeneous regions while the background is static (except for camera pan and zoom) and contains a more complicated textural information (see Figure 2.1a).

The input to the proposed cartoon analysis framework is a sequences of animation frames containing common background layer (Figure 2.1a). For each frame of this sequence the aim is to detach foreground and background layers and reconstruct the state before the final composition. For this task a novel outline-based cartoon segmentation algorithm is designed (Figure 2.4) that utilizes a robust outline detector in order to locate region boundaries with sub-pixel accuracy (Figure 2.1b). After the segmentation each region is roughly classified whether it belongs to the foreground (Figure 2.1c) or to the background layer (Figure 2.1d). Then planar camera movements are tracked through the input animation sequence using standard image registration techniques. By exploiting estimated motion vectors and pre-classified parts of the background layer one image is stitched together (Figure 2.1e). Reconstructed background layer is then reused to further refine foreground/background classification of regions and outlines. Finally, shape of each region is converted from raster to vector representation using standard contour tracing algorithm.

The final step is retrieval of structural correspondences between frames and regions. Such a valuable information is later used in applications to predict color transfer between regions and also to lower encoding bit-rate by exploiting frame redundancy in the animation. The retrieval itself is hierarchical. The most similar animation phase is selected first and then local structural similarity and neighborhood relations are used to estimate region correspondences (see Figure 2.1f).

2.2 Segmentation

At the beginning of this section state-of-the-art of image segmentation is discussed briefly to show why the most of previously published approaches do not suit for the task of precise segmentation in the case of classical cartoon images. Afterwards a novel outline-based cartoon segmentation algorithm is described in detail including practical experiments on real cartoon images.

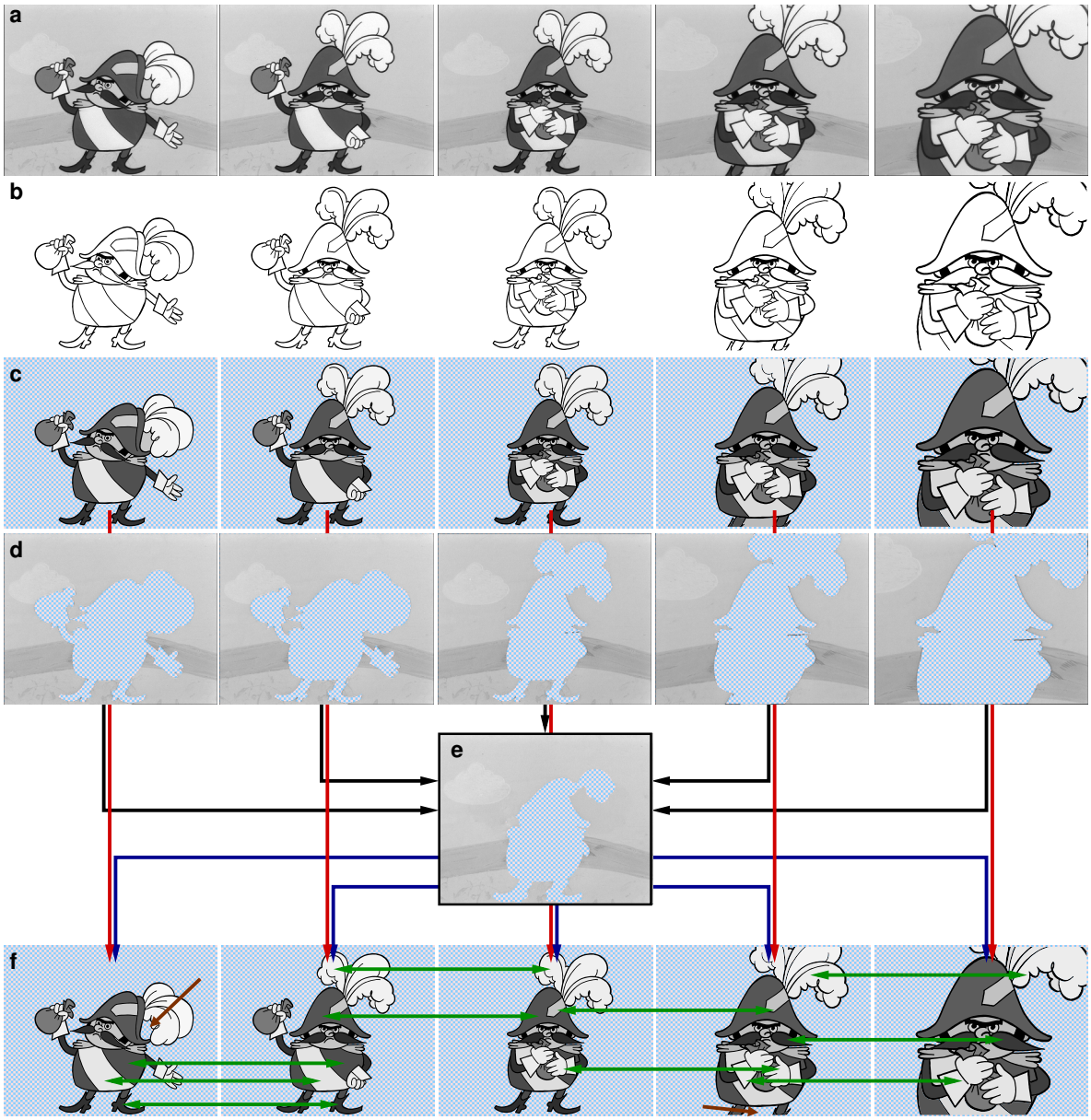

Figure 2.1: **Cartoon analysis framework in progress**: (a) input animation sequence, (b) outline detection, (c) pre-classified foreground layer, (d) visible portion of background layer, (e) reconstructed background layer (black arrows denote individual parts used for reconstruction), (f) correspondence retrieval in refined foreground layer (green arrows show several corresponding regions). Brown arrows in (f) show regions where the foreground/background classification was refined by combining information from pre-classified foreground layer (red arrows) and reconstructed background layer (blue arrows).

2.2.1 Previous work

The first aim of the proposed framework is to perform precise segmentation of cartoon images. At a first sight the task looks simple, however, several experiments demonstrate that

common approaches such as *single/multi-level thresholding* [34] (see Figure 2.2f,b) or *watersheds* [173] produce unacceptable over-segmentation. To alleviate it *region merging* techniques are used [64]. However, still in most cases the quality of the final segmentation is not acceptable since visually important details are typically omitted (see Figure 2.2c). Similar problem occurs even when more advanced techniques such as *mean-shift* [40] or *normalized cuts* [158]) are used (see Figure 2.2d,e).

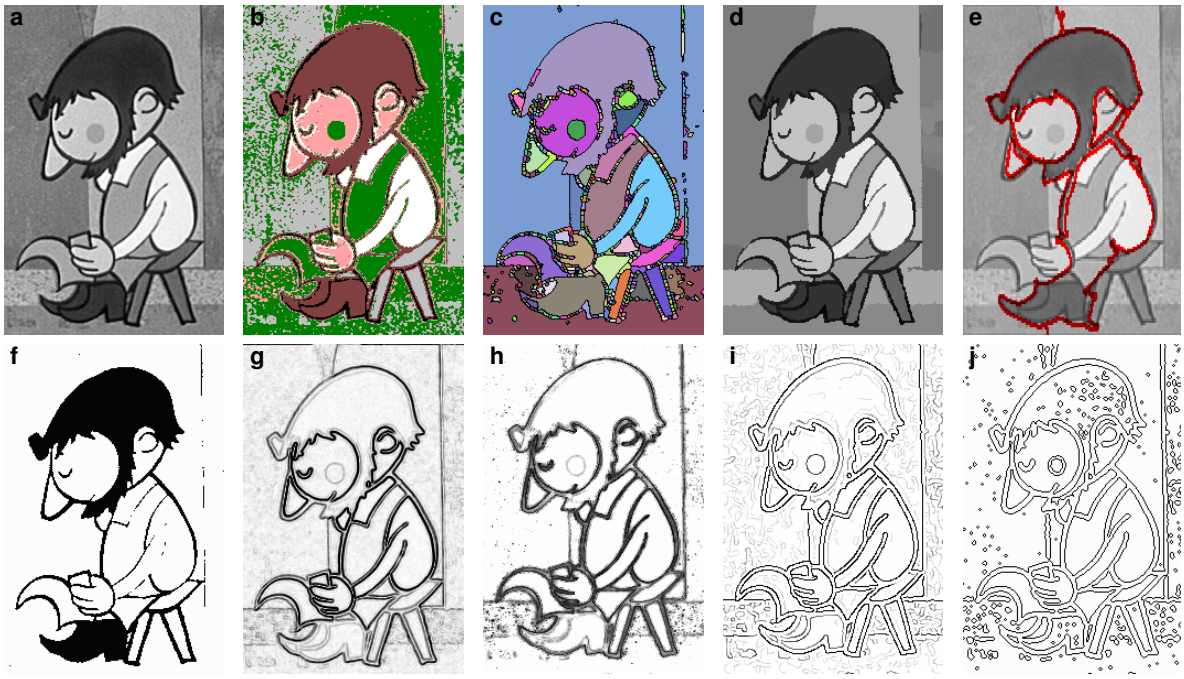

Figure 2.2: **An overview of image segmentation** (top) **and edge detection techniques** (bottom): (a) the original image, (b) multi-level thresholding, (c) watersheds with region merging, (d) mean-shift, (e) normalized cut, (f) simple thresholding, (g) Sobel, (h) SUSAN, (i) Canny, (j) $\mathbf{L} \circ \mathbf{G}$ zero-crossings.

Another large family of successful image segmentation techniques exploits *edge detection* to estimate locations of dominant step changes in the image luminance. The common assumption is that such locations usually represent region boundaries therefore can be used for segmentation.

In the literature various edge detection techniques have been proposed (for detailed survey see [65, 195]). *Sobel* edge detector is one of the simplest (see Figure 2.2g). It uses two convolution filters of size `3x3` to approximate first-order derivative of the image in the horizontal and vertical direction. Non-maxima suppression and/or thresholding with hysteresis is then used to extract salient edges. However, due to additive noise, false edges reveal and true edges disappear.

Canny [23] addressed this problem using variational approach and derived an optimal convolution filter able to extract edges in the presence of Gaussian noise. *Canny*'s filter can be approximated by directional first-order derivative of `2D` Gaussian. Comparable results to *Canny* edge detector produces also `SUSAN` (Smallest Univalue Segment Assimilating Nucleus) [159] – a popular edge detector that is not based on the principle of first-order derivatives thus

it does not suffer from additive noise. However, what is common for these (and other) edge detection techniques is the need of some thresholding mechanism that usually tends to omit important low-contrast details and break edge continuity (see Figure 2.2i,h). To alleviate such limitation complex edge connecting techniques have to be used [122].

Marr and *Hildreth* [114] shown that human visual system perceives shapes through the process that can be modelled using convolution with the Laplacian of Gaussian filter ($\mathbf{L} \circ \mathbf{G}$). This joint filter similarly to *Canny*'s performs two operations in one pass: Gaussian suppresses noise and Laplacian estimates second-order derivative of the noise-suppressed image. $\mathbf{L} \circ \mathbf{G}$ filter can be derived as follows:

$$\mathbf{G} = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.1}$$

and

$$\mathbf{L} = \nabla^2 = \nabla \circ \nabla = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}, \tag{2.2}$$

where $\mathbf{G}$ is Gaussian filter and $\mathbf{L}$ is Laplace operator (both in two dimensions). Due to linearity of convolution $\nabla^2(\mathbf{G} \circ \mathbf{I}) = (\nabla^2 \mathbf{G}) \circ \mathbf{I}$ symbolic derivation can be used to precalculate algebraic form of $\mathbf{L} \circ \mathbf{G}$(see Figure 2.3):

$$\mathbf{L} \circ \mathbf{G} = \nabla^2 \mathbf{G} = \frac{1}{\pi\sigma^4} \left(\frac{x^2+y^2}{2\sigma^2} - 1 \right) e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{2.3}$$

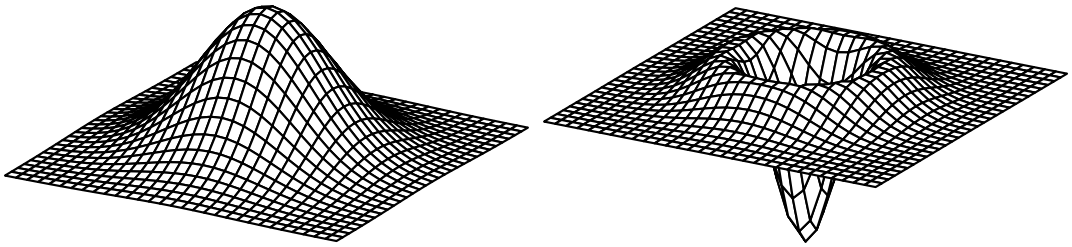

Figure 2.3: **Gaussian** (left) **and the Laplacian of Gaussian** (right).

Since $\mathbf{L} \circ \mathbf{G}$ response approximates second-order derivatives of the smoothed image, edges can be located at zero-crossings (local maxima or minima of the first-order derivative). On a discrete lattice edge detection can be approximated by a simple zero-crossing test: if two neighbor pixels differ in the sign then the edge is located in-between [38]. The main advantage in contrast to techniques based on the first-order derivatives is that no explicit thresholding is needed, all zero-crossings are equally important.

Another interesting feature of $\mathbf{L} \circ \mathbf{G}$ is that its zero-crossings form continuous closed curves (see Figure 2.2j) [105]. Consequently, $\mathbf{L} \circ \mathbf{G}$ response inside the region enclosed by zero-crossing should be only positive or negative. In the case of cartoon images, outlines are darker in contrast to neighbor regions, therefore $\mathbf{L} \circ \mathbf{G}$ response inside the outline is negative (see Figure 2.5b). These two important properties allow to design new outline detector that leads to an accurate cartoon segmentation algorithm. The overall concept of this algorithm is depicted in Figure 2.4 and its individual steps are described in the following sections.

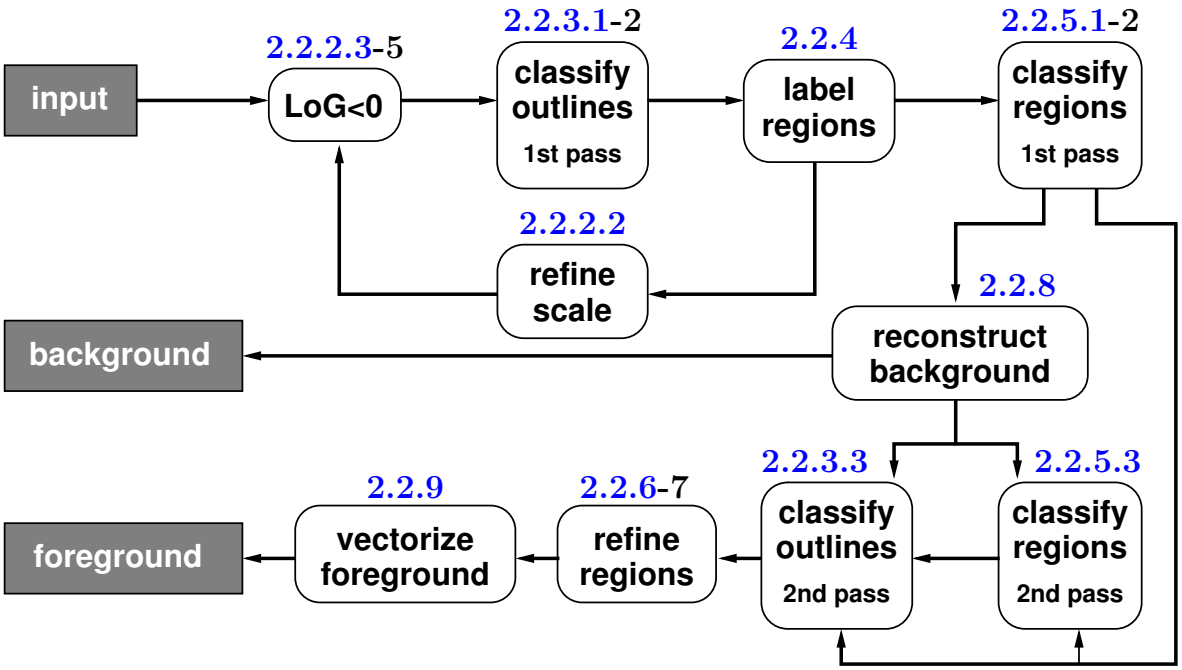

Figure 2.4: **Flowchart of the proposed segmentation algorithm**: numbers on the top of the individual boxes indicate sections where the corresponding operations are described in more detail.

2.2.2 Outline detector

The key idea of the proposed segmentation algorithm is to exploit $\mathbf{L} \circ \mathbf{G}$ response not for edge detection as is usual but directly for outline detection. Instead of performing zero-crossing test (see Figure 2.2j), $\mathbf{L} \circ \mathbf{G}$-negative response is used to mark out outline candidates, i.e. darker regions surrounded by brighter areas (see Figure 2.5b). It is important to stress the fact that (as in zero-crossing test) no additional thresholding is needed, each $\mathbf{L} \circ \mathbf{G}$-negative area has the same chance to be later classified as an outline despite of its actual contrast.

2.2.2.1 Algorithm overview

A brief overview of the proposed outline detector is depicted in Figure 2.5. First $\mathbf{L} \circ \mathbf{G}$-negative mask is generated (Figure 2.5b). In this step proper $\mathbf{L} \circ \mathbf{G}$ kernel size and scale is estimated. This is done by adaptive σ-fitting technique that removes need for tedious parameter tuning. Afterwards an adaptive classification algorithm is used to extract foreground outlines (Figure 2.5c), remaining $\mathbf{L} \circ \mathbf{G}$-negative areas are removed and the final mask of outlines is generated (Figure 2.5d).

2.2.2.2 Adaptive σ-fitting

The first important parameter that controls the response of $\mathbf{L} \circ \mathbf{G}$ is *standard deviation* σ (2.3). By varying σ edges at different scales are focused [185] (see Figure 2.6). When the thickness of outlines is small (< 6 pixels), constant $\sigma = 1.25$ is usually sufficient for most

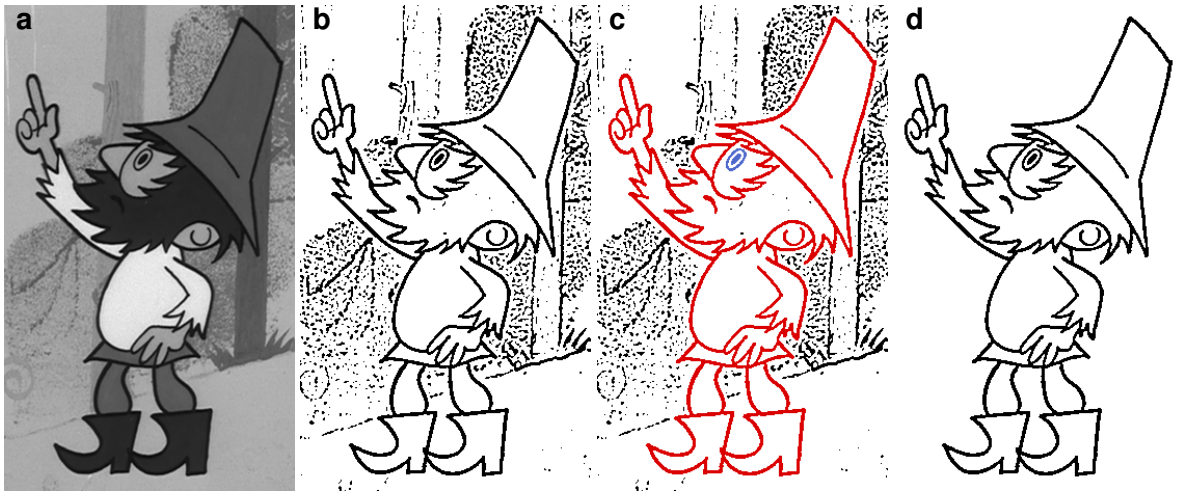

Figure 2.5: **Outline detector in progress**: (a) the original image, (b) $\mathbf{L} \circ \mathbf{G}$-negative mask, (c) adaptive outline classification, (d) extracted outlines.

images. However, when the thickness is greater (> 6 pixels) then an additive noise and/or so called *phantom edges* (minima of the first-order derivatives) [38] may produce small $\mathbf{L} \circ \mathbf{G}$-positive regions inside the outline (see Figure 2.7). In this case it is necessary to retrieve an optimal σ – large enough to eliminate these artifacts but also small enough to preserve important details.

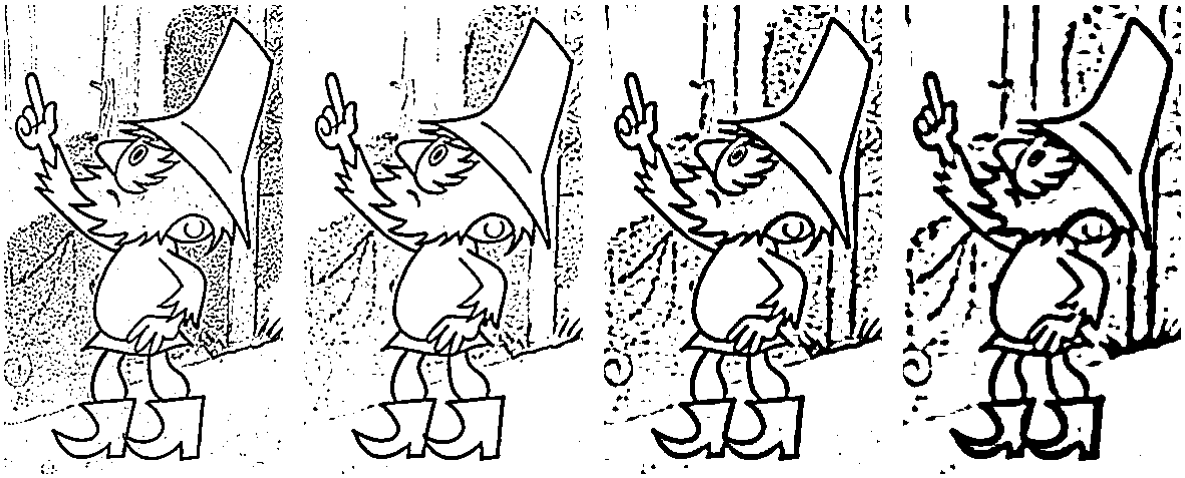

Figure 2.6: $\mathbf{L} \circ \mathbf{G}$**-negative mask with increasing** σ (from left to right): 1.0, 1.5, 2.0, 3.0.

To avoid labour-intensive manual tuning, coarse-to-fine strategy is proposed to retrieve an optimal σ automatically. The basic observation is that typically the number of small $\mathbf{L} \circ \mathbf{G}$-positive regions decreases monotonously with increasing σ and thus a simple iterative interval subdivision strategy can be used to retrieve optimal σ. Similar approach have been used also for edge focusing in [62].

The σ-fitting process works as follows. First $\mathbf{L} \circ \mathbf{G}$ scale-space is divided into a set of smaller intervals. In each interval a number of small $\mathbf{L} \circ \mathbf{G}$-positive regions is computed using flood-fill

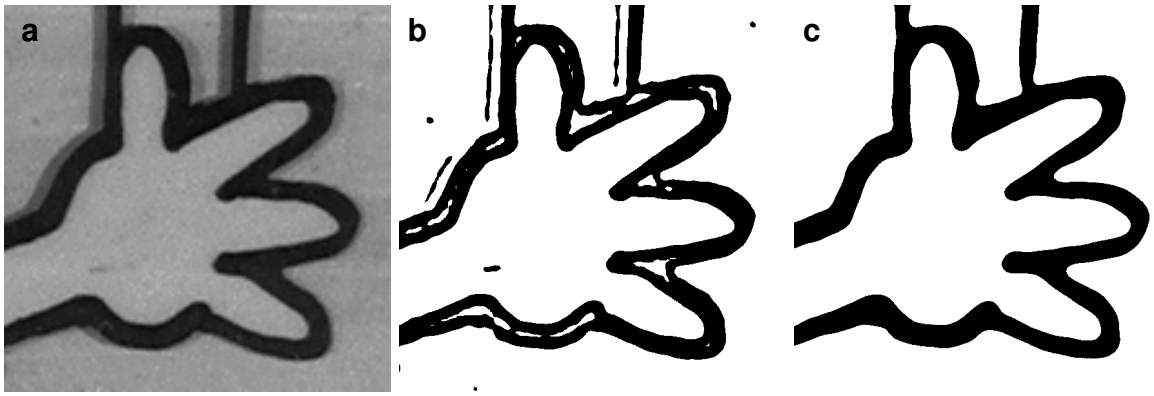

Figure 2.7: **An example of adaptive σ-fitting**: (a) the original image with thick outlines, (b) $\mathbf{L} \circ \mathbf{G}$-negative mask with small $\mathbf{L} \circ \mathbf{G}$-positive regions, (c) $\mathbf{L} \circ \mathbf{G}$-negative mask after σ-fitting.

with pixel counter started at each unfilled $\mathbf{L} \circ \mathbf{G}$-positive pixel. Small number of filled pixels indicates small region. Afterwards an interval I is selected where the lower bound σ produces $\mathbf{L} \circ \mathbf{G}$-positive regions but the upper bound σ guarantees that the number of $\mathbf{L} \circ \mathbf{G}$-positive regions falls under the specified limit. The same strategy is used recursively for I until the difference between the lower and upper bound σ becomes considerably small (see Figure 2.8). This full search algorithm is invoked only when the first animation frame is processed. During the animation the optimal σ is refined using only small initial interval around the optimal σ from the previous frame.

2.2.2.3 Allowable aliasing energy

Another important parameter that influences the quality of outline extraction is *allowable aliasing energy* p_a. Due to computational efficiency the infinite support of $\mathbf{L} \circ \mathbf{G}$ convolution kernel has to be truncated. However, truncation in the spatial domain causes periodical repetition in the frequency domain and vice versa. This repetition introduces aliasing energy expressed as follows:

$$\frac{100 - p_a}{100} = \frac{\sigma^6}{2\pi} \int_{-\alpha}^{\alpha} \int_{-\alpha}^{\alpha} \frac{\left(u^2 + v^2\right)^2}{\exp\left(\sigma^2 \left(u^2 + v^2\right)\right)} \mathrm{d}u\mathrm{d}v, \tag{2.4}$$

where α is aliasing frequency. For a given p_a and standard deviation σ the aliasing frequency α and consequently the size of $\mathbf{L} \circ \mathbf{G}$ kernel can be precalculated via numerical integration [161]. By increasing p_a the truncated convolution kernel increases its low-pass property and consequently produces additional smoothing in the filtered image (see Figure 2.9). According to many experiments performed on real cartoon images a constant value $p_a = 10\%$ produces optimal results.

2.2.2.4 Performance optimizations

Brute force filtering even with kernel truncation is still time consuming. In practice it is necessary to exploit several optimization techniques to avoid redundant computations. They

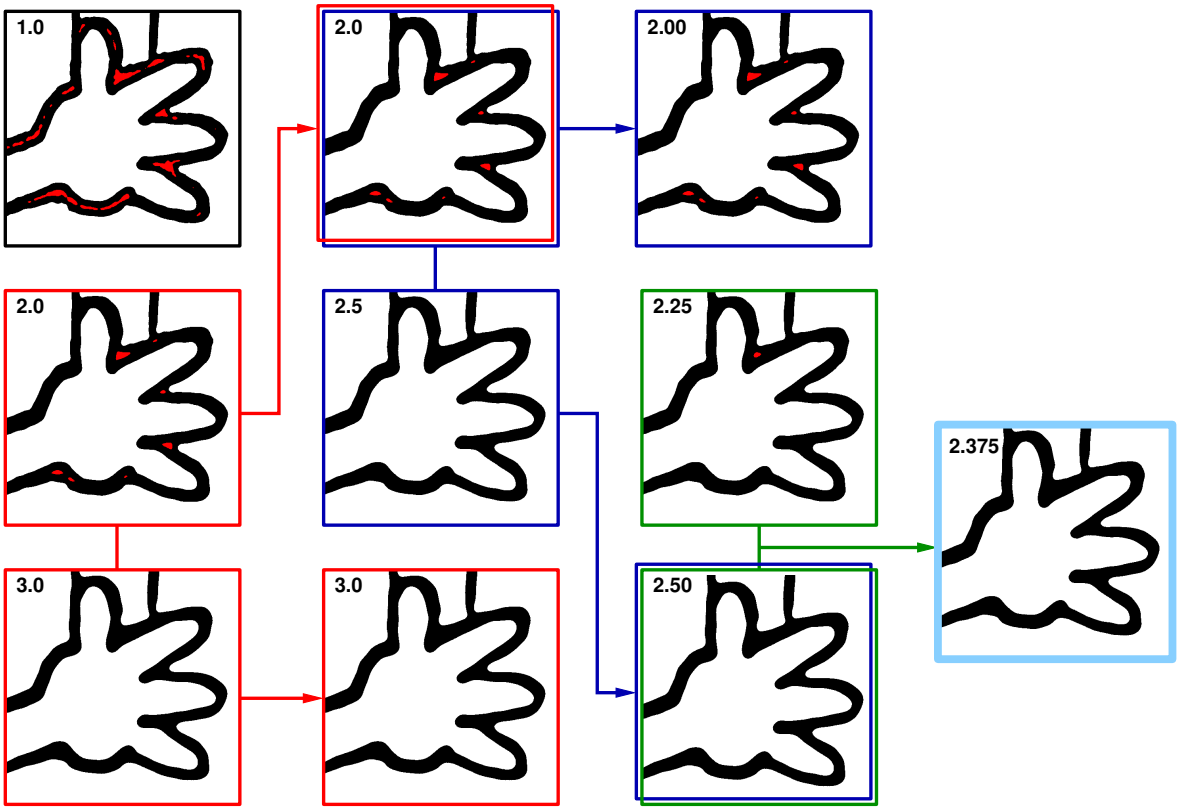

Figure 2.8: **An adaptive σ-fitting in progress**: an iterative elimination of small $\mathbf{L} \circ \mathbf{G}$-positive regions inside the outline. Two scale-space intervals are selected first: $\sigma \in (1.0, 2.0)$ and $\sigma \in (2.0, 3.0)$. Then it happens that $\sigma = 2.0$ does and $\sigma = 3.0$ does not produce small $\mathbf{L} \circ \mathbf{G}$-positive regions. Thus the interval $\sigma \in (2.0, 3.0)$ is refined by $\sigma \in (2.0, 2.5)$ and $\sigma \in (2.5, 3.0)$. The same approach is applied recursively until the difference between interval endpoints is less than 0.25.

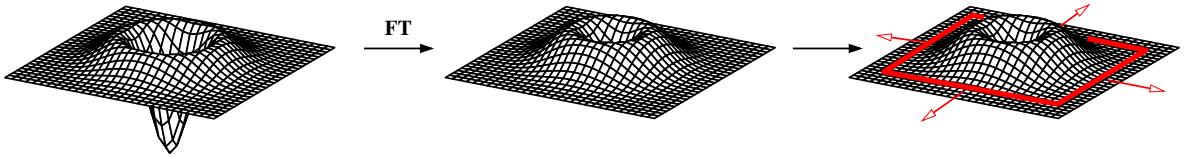

Figure 2.9: **Truncating $\mathbf{L} \circ \mathbf{G}$ convolution kernel in the frequency domain.**

allows to compute an accurate $\mathbf{L} \circ \mathbf{G}$ response more than ten times faster as compared to brute force approach.

The original image can be first pre-smoothed via separable version of Gaussian filter and afterwards $\mathbf{L} \circ \mathbf{G}$ convolution with smaller support can be used [30]. The $\mathbf{L} \circ \mathbf{G}$ itself is not separable but can be exactly decomposed into a summed multiplication of two `1D` filters [89]:

$$\nabla^2 \mathbf{G}(x, y) = \mathbf{h}_1(x) \cdot \mathbf{h}_2(y) + \mathbf{h}_2(x) \cdot \mathbf{h}_1(y), \tag{2.5}$$

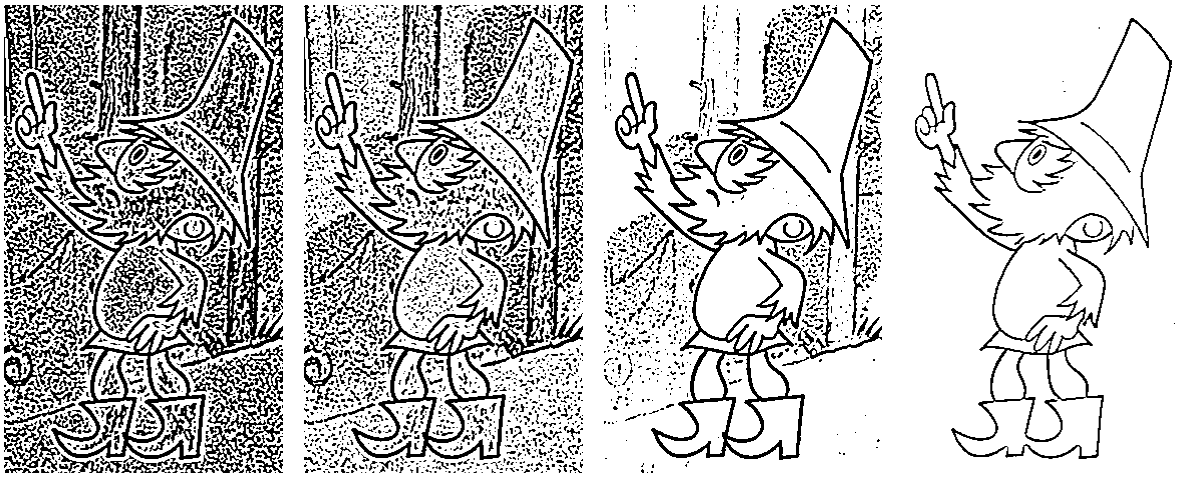

Figure 2.10: **$\mathbf{L}\circ\mathbf{G}$-negative mask with increasing allowable aliasing energy** (from left to right): $p_a = 1\%$, 5%, 10%, 25%.

where

$$\mathbf{h}_1(\rho) = \frac{1}{\sqrt{2\pi\sigma^2}}\left(1-\frac{\rho^2}{\sigma^2}\right)\exp\left(-\frac{\rho^2}{2\sigma^2}\right) \tag{2.6}$$

and

$$\mathbf{h}_2(\rho) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\rho^2}{2\sigma^2}\right). \tag{2.7}$$

Additionally when the thickness of outlines requires large σ, it is possible to down-sample the original image and then use smaller supports for $\mathbf{L}\circ\mathbf{G}$ kernel [161].

2.2.2.5 Sub-pixel accuracy

Another important advantage of $\mathbf{L}\circ\mathbf{G}$ is that its response can be simply up-sampled using bi-linear or bi-cubic interpolation and then $\mathbf{L}\circ\mathbf{G}$-negative areas can be extracted with sub-pixel precision (this idea has been first presented in the context of sub-pixel accurate zero-crossing test [75]). Thanks to noise suppression property of $\mathbf{L}\circ\mathbf{G}$, up-sampling in the $\mathbf{L}\circ\mathbf{G}$-domain produces considerably less artifacts as compared to common image domain approach. In practice it is possible to reach fourfold accuracy without noticeable artifacts (see Figure 2.11). This is valuable especially for applications where highly accurate segmentation is needed.

2.2.3 Outline classification

When the optimal $\mathbf{L}\circ\mathbf{G}$-negative mask is generated, the next important step is to estimate which $\mathbf{L}\circ\mathbf{G}$-negative area is actually an outline. This step can be seen as a variation of thresholding already rejected in Section 2.2.1. However, the key difference here is that the process is (1) adaptive (no fixed threshold value is set in advance) and (2) the classification is done on a coarse level in contrast to common local schemes where strong edges are detected first and then joined to form region boundary. In the proposed methodology no local joining is necessary, $\mathbf{L}\circ\mathbf{G}$-negative areas are classified entirely.

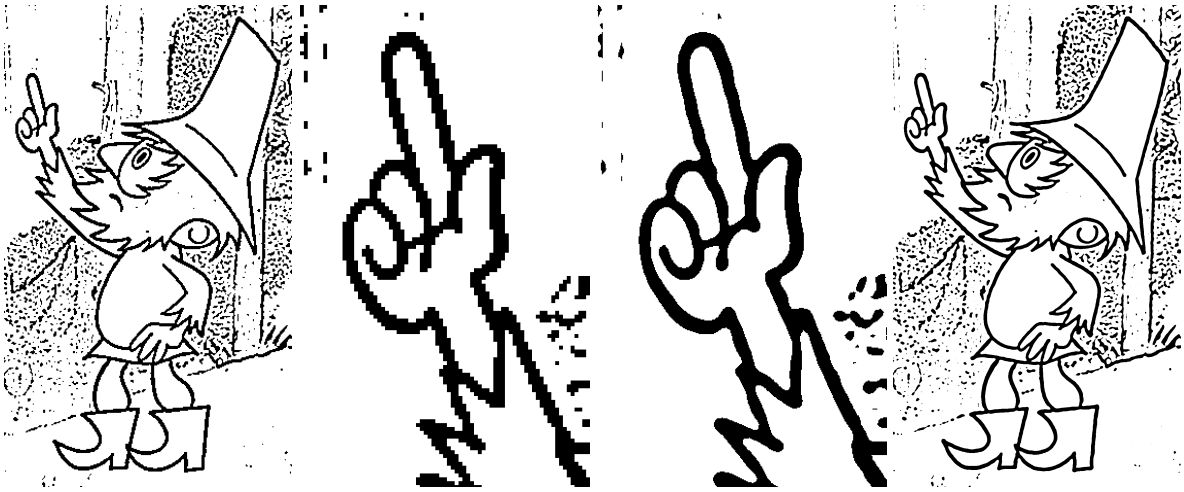

Figure 2.11: **Comparison of the $\mathbf{L} \circ \mathbf{G}$-negative mask generated with pixel** (left) **and with sub-pixel** (right) **accuracy using $\mathbf{L} \circ \mathbf{G}$-domain interpolation.**

2.2.3.1 Adaptive flood-filling

The prior assumption of the proposed approach is that outlines are dark thus pixels with minimal luminance over all $\mathbf{L} \circ \mathbf{G}$-negative areas can be classified as outlines first (red dots in Figure 2.12). This initial guess is then propagated to the remaining pixels inside the same $\mathbf{L} \circ \mathbf{G}$-negative area using standard flood-fill algorithm (red outlines in Figure 2.12). At this point it is easy to see why this technique outperforms local approaches: all pixels in the same $\mathbf{L} \circ \mathbf{G}$-negative area are classified equally despite of their actual local contrast.

However, such a simple one shot algorithm failed when some separated outlines do not contain pixels with minimal luminance. In this case an adaptive mechanism should be used to estimate additional brighter seed points (see eye in Figure 2.12). The process is iterative. In each iteration intensities of already filled $\mathbf{L} \circ \mathbf{G}$-negative pixels are used to estimate the luminance median $\tilde{L}_k$. Afterwards a set of new flood-fill seeds is generated at pixels p_i (blue points in Figure 2.12) where the luminance $L(p_i) < \tilde{L}_k/2$. This process is repeated until the current luminance median $\tilde{L}_k$ is lower or the same as the value from the previous step $\tilde{L}_{k-1}$. In most cases this situation occurs immediately after the first step as is depicted in Figure 2.12 where almost all outlines have been filled in the first step and only the eye in the second step (blue outlines in Figure 2.12).

2.2.3.2 Flood-fill with priority

Another rare problem that can reveal during the outline classification is outline leaking caused by coalescent $\mathbf{L} \circ \mathbf{G}$-negative areas in the background layer (see Figure 2.13). In such a problematic case previous flood-fill based adaptive classification produces unacceptable results (see red outlines in Figure 2.13) so it is necessary to incorporate pixel-wise classification. However, as will be discussed later the approach proposed here differs from the local thresholding scheme used in standard edge-detection techniques since it natively preserves continuity of outlines.

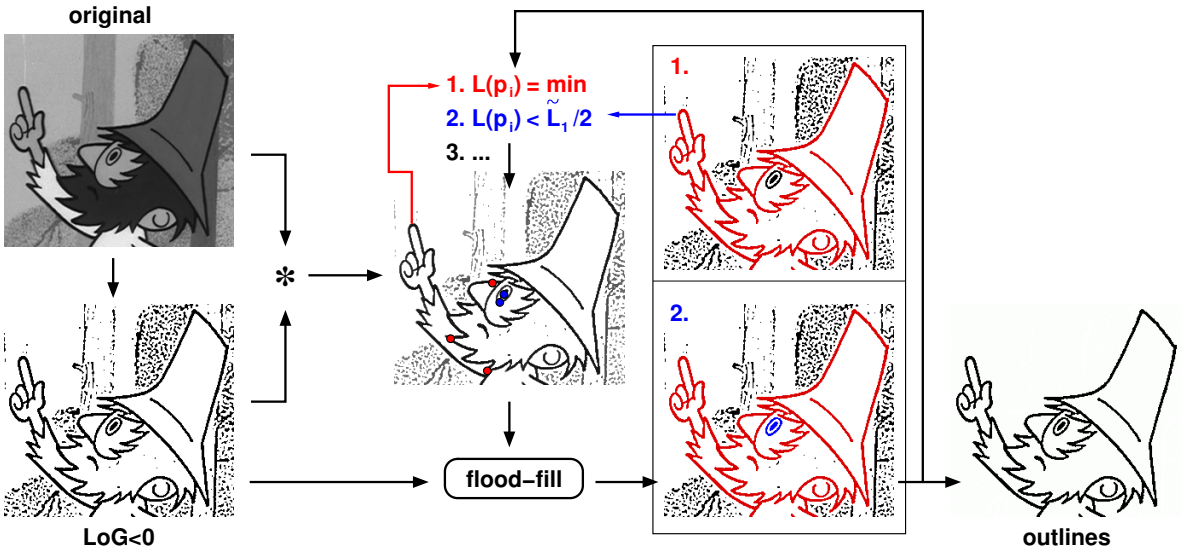

Figure 2.12: **An adaptive outline classification in progress (1)**: the original gray-scale image is masked with $\mathbf{L} \circ \mathbf{G}$-negative response, then pixels with the minimal intensity over the masked area (red dots) are selected as seed pixels for the first flood-filling step in the $\mathbf{L} \circ \mathbf{G}$-negative mask (results in red outlines). Luminance of already filled pixels is used to compute the luminance median $\tilde{L}_k$. Then unfilled pixels with luminance $L(p_i) < \tilde{L}_k/2$ are marked as new seed points (blue dots) and the next flood-filling step is applied (results in blue outlines). This operation is repeated until the luminance median does not increase $\tilde{L}_k \leq \tilde{L}_{k-1}$.

The first important modification that allows to locally detach dark outlines from brighter coalescent $\mathbf{L} \circ \mathbf{G}$-negative areas is scale adaptive contrast enhancement. Similar process is used in the human visual system [114] where proper scale changes considerably affects perceptual grouping mechanism [125]. Scale-sensitive contrast enhancement can be achieved by adding full $\mathbf{L} \circ \mathbf{G}$ response computed with proper scale σ (Section 2.2.2.2) to the original image $\mathbf{I}$. This operation produces new sharpened image $\mathbf{I}^\star = \mathbf{I} + \nabla^2 \mathbf{G}_\sigma \circ \mathbf{I}$ where the contrast between outlines and coalescent $\mathbf{L} \circ \mathbf{G}$-negative areas is enhanced (see Figure 2.13).

When the sharpened image $\mathbf{I}^\star$ is computed, the last flood-filling step from Section 2.2.3.1 is repeated, but instead of stack priority queue is used to store and expand seed pixels during the propagation. The priority is given from the inverted luminance of the sharpened image $\mathbf{I}^\star$ (in the middle of Figure 2.13). In contrast to simple stack-based flood-fill used in Section 2.2.3.1, this modification allows to visit dark pixels first and then gradually expand towards brighter values. A key issue here is to know where to stop the propagation, i.e. to select proper priority threshold. Fortunately, a good estimation for this value is the position of the first important valley (for robust valley detection see [34]) in the priority histogram computed over already filled $\mathbf{L} \circ \mathbf{G}$ zero-crossings (see Figure 2.13).

2.2.3.3 Background subtraction

Theoretically the most successful approach to removal of coalescent outlines is to reconstruct the background layer (see Section 2.2.8) and then subtract it from the original image (see Figure 2.14). When the sum of absolute differences is small enough the outline pixel can be

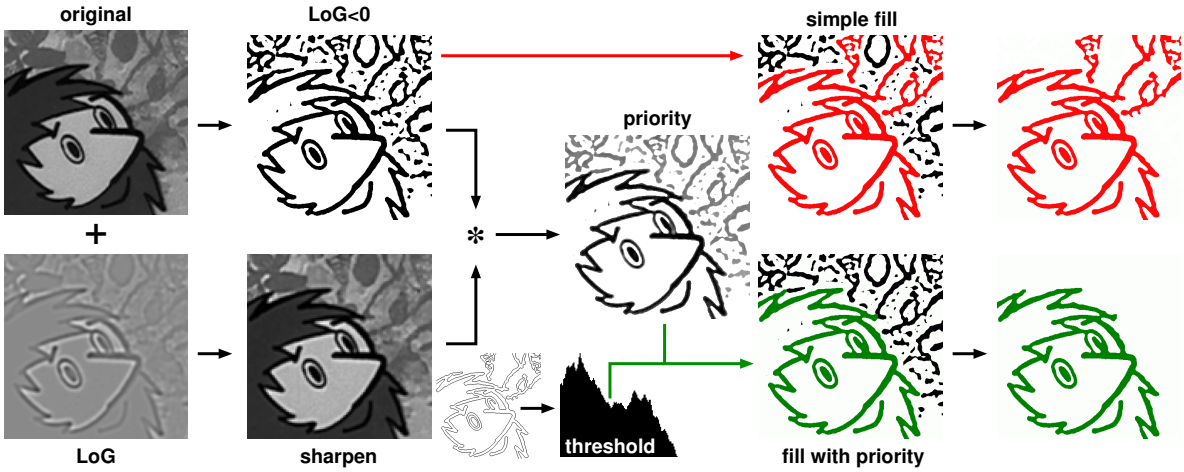

Figure 2.13: **An adaptive outline classification in progress (2)**: a problematic case when a simple adaptive outline flood-fill failed (red), outline classification using flood-fill with priority queue (green), in the middle is a mask of priorities derived from $\mathbf{L} \circ \mathbf{G}$-negative response and sharpened image (black means high and white means low priority), the stopping priority estimated from the histogram of priorities located at $\mathbf{L} \circ \mathbf{G}$ zero-crossings.

removed. Such technique is usable especially in cases when the flood-fill with priority tends to fail, i.e. when the contrast between background and foreground outlines is very low (see Figure 2.14).

However, there is one important issue. When outlines in the background layer coincide with the foreground layer, small gaps reveal at the point of intersection. To eliminate them, subtraction is avoided at boundaries belonging to foreground regions. They can be detected and masked by growing region boundaries (see Sections 2.2.5 and 2.2.6). When this masking is available only several outer outline recessions remain. They can be healed up by estimating local thickness of outlines as discussed in Section 2.2.7.

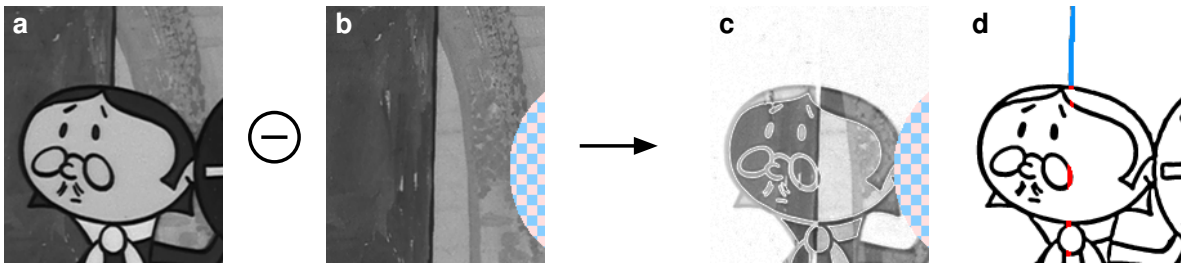

Figure 2.14: **Outline classification using background subtraction**: reconstructed background layer (b) is subtracted from the original image (a), resulting absolute difference image (c) is used for outline classification (d). Portions of outlines belonging to foreground regions are masked (red color). When the absolute difference falls under a specified limit then the corresponding unmasked outline pixel can be removed (blue color).

Although outline classification using background subtraction can produce compelling results, its main drawback is the need of an initial classification of regions. Fortunately, such a

classification does not need to be perfect. Conservative approaches described in Sections 2.2.3.1 and 2.2.5.1 can be utilized to produce it. However, still there should be a frame in the original animation sequence where the coalescent outlines are not detected as a part of the foreground layer, i.e. it is possible to reconstruct the corresponding part of the background layer.

2.2.4 Region labelling

When outlines are extracted it is easy to assign an unique label to each closed region. This can be done by scanning the mask of outlines pixel by pixel and running simple flood-fill algorithm at each unlabelled pixel to fill-in the region with a new label (see Figure 2.15b). The same result can be obtained using well known scan-line based region labelling algorithm [143]. The only trouble are imprecisely closed regions. For them a variant of edge or line joining algorithm can be used [122, 153].

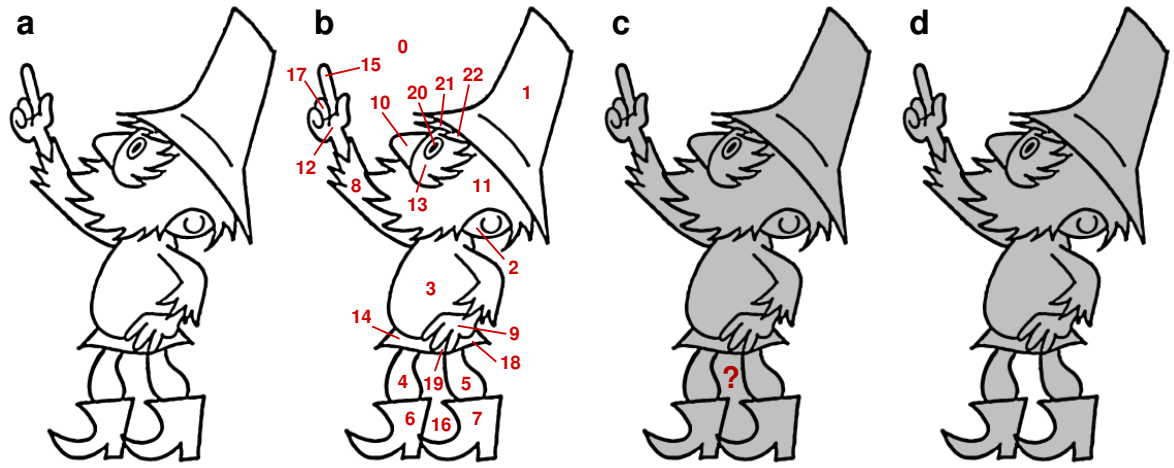

Figure 2.15: **Region classification in progress**: (a) extracted outlines, (b) region labelling, (c) initial classification using region area size thresholding, (d) improved classification using reconstructed background layer.

After the labelling, three important statistics are computed for each region: region area size, luminance histogram and luminance median (or mean color). Since mean or median of small regions can be significantly biased by outline anti-aliasing it is recommended to compute maximum luminance that is not robust but in most cases it better reflects the visual appearance of such regions.

2.2.5 Region classification

The final task of the segmentation algorithm is to decide whether the given region belongs to the background or to the foreground layer. In this case the same limitation reveals as in outline classification (Section 2.2.3), i.e. when a single image is considered only a rough approximation can be obtained. A more precise classification is possible as late as the background layer can be partially reconstructed (see Figure 2.15).

2.2.5.1 Area size thresholding

First good approximation to foreground/background classification is region area size thresholding. According to experiments performed on real cartoon images, critical size of the foreground region is usually 15% of the total image size. Larger regions are classified as background. It is obvious that such a simple approach can be very imprecise (for instance it is unable to recognize small hole between legs in Figure 2.15c), nevertheless, it is simple to compute and in most cases produces sufficient results.

2.2.5.2 Homogeneity

Another possibility how to classify background regions is to estimate region homogeneity by examining luminance histogram [34]. Two or more significant peaks may denote inhomogeneities caused by underlaying textural information. However, when the occluded part of background is also homogeneous it is still not possible to distinguish it.

2.2.5.3 Background subtraction

A more precise classification can be obtained when reconstructed background is available (see Section 2.2.8). In such a case (as in Section 2.2.3.3) background subtraction can be used. Again the normalized sum of absolute differences is computed over the region area between the original image and reconstructed background and if this sum falls under a specified limit then the region can be classified as background (see green arrow in Figure 2.16).

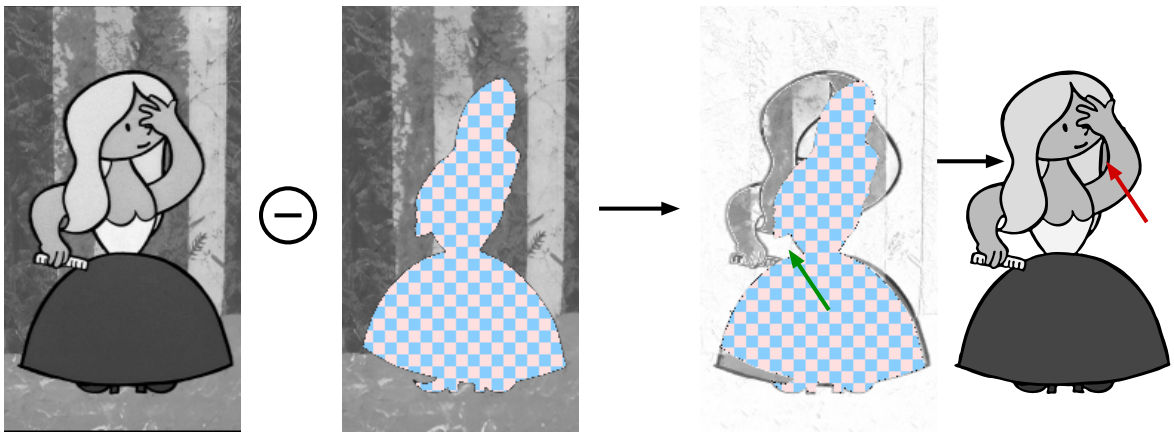

Figure 2.16: **Region classification using background subtraction**: reconstructed background is subtracted from the current animation frame (left). Region is classified as background when normalized sum of absolute differences is small (green arrow). Sometimes portions of background have never been pre-classified as background (chessboard pattern). In this case it is not possible to refine the initial classification (red arrow).

Similarly to median computation (Section 2.2.4) the background subtraction for small regions has low discriminative power since the normalized sum of absolute differences can be strongly biased by outline anti-aliasing. Fortunately such regions are entirely homogenous and thus it usually does not matter whether they remain in the foreground layer or not (see applications

in Sections 3 and 5). Much more problematic are larger homogenous parts of background that have the same mean luminance/color as the foreground layer or have never been visible or pre-classified as background (chessboard pattern in Figure 2.16). In such locations background subtraction cannot provide any kind of decision. When a perfect classification is needed they should be located and marked manually.

2.2.6 Region growing

In cartoon colorization (Section 3) it is necessary to leak the color inside the region outline to produce seamless colorization of anti-aliased boundaries. An issue here is where to stop the leaking. A naive approach to this problem is to compute medial axis of the detected outline (also known as *skeleton* [156]) (see Figure 2.17d) and then stop the flooding when the medial axis is reached. However, as is depicted in Figure 2.18c, visually disturbing artifacts may occur due to significant loss of information as compared to the original gray-scale image.

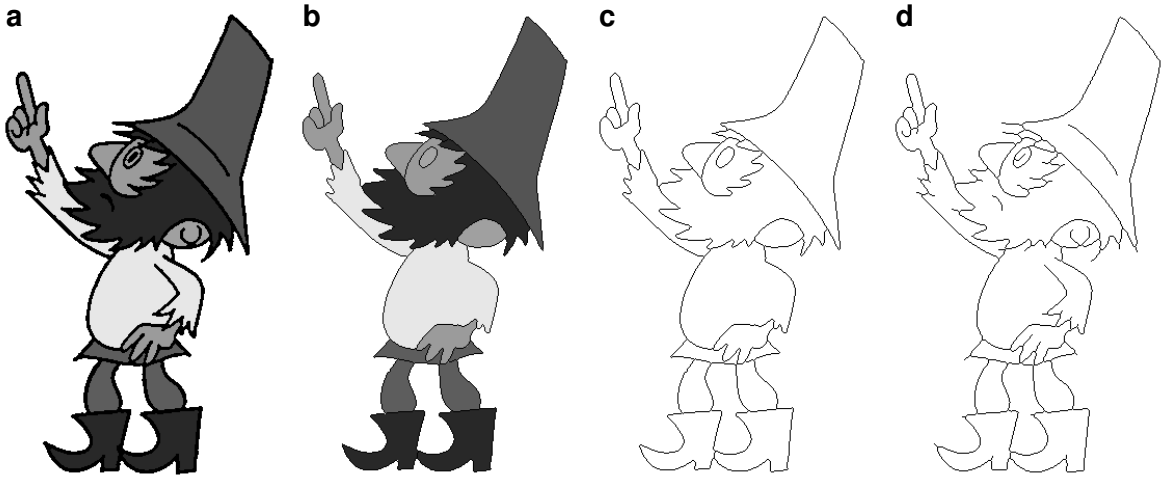

Figure 2.17: **Region growing**: (a) outline-based segmentation, (b) region growing using flood-filling with priority queue, (c) intrinsic region boundaries after region growing in contrast to (d) outline skeleton.

To produce correct results simple region growing algorithm is used instead of outline skeletonisation. For each outline pixel the nearest region is found using flood-filling with priority queue (as in Section 2.2.3). In contrast to outline classification the priority here is the original luminance and the stop condition is a reaching of the first region pixel. Since the algorithm expands brighter pixels first and since the `3D` luminance profile of an outline is conformable to a valley, a correct region boundary is retrieved (see Figures 2.17b and 2.18d).

2.2.7 Fragment extraction

In cartoon-by-example (Section 4) there is a need to extract user-defined subset of regions from the foreground layer. Since two neighboring regions share the same outline, an issue is to decide which part of the original outline belongs to the selected region. In general case this problem can be ambiguous (see e.g. [127]) and only additional constraints such as convexity

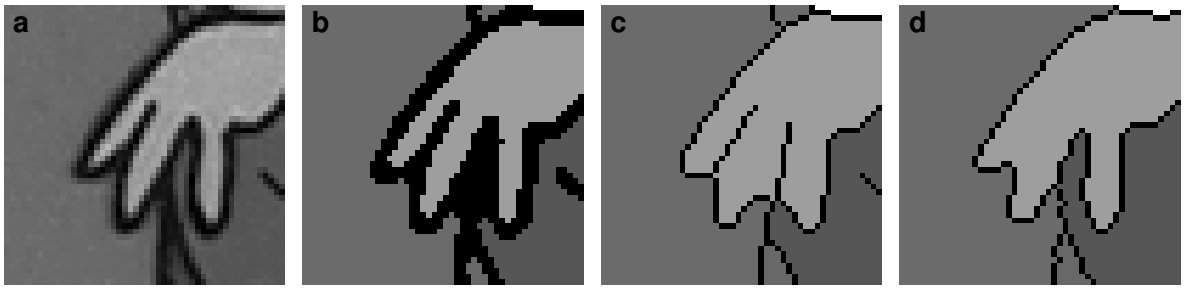

Figure 2.18: **Region growing (detail view)**: (a) the original image, (b) outline-based segmentation, (c) region growing using skeletonisation, (d) region growing using flood-filling with priority queue.

may help to resolve it. However, in cartoon animations outlines are usually smooth and have constant thickness that can be estimated using an approximation to Euclidean distance transform [13].

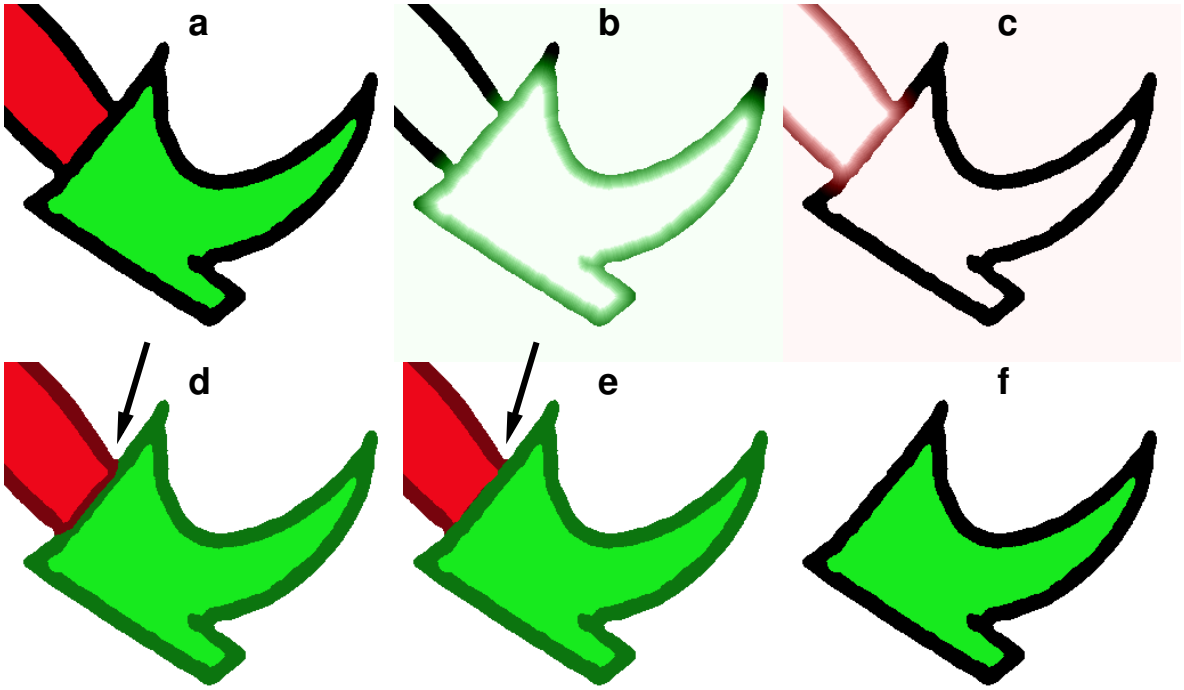

Figure 2.19: **Fragment extraction**: (a) the green shoe needs to be extracted from the red leg, (b) distance field for the shoe and (c) for the leg, (d) partition where the distance to the shoe and to the leg is the same, (e) refined partition, (f) final extraction.

Two distance fields are computed: one for selected and one for remaining regions (see Figure 2.19b,c). Then distances from both fields are compared in each outline pixel to decide whether it is closer to the desired fragment or to the remaining part of the foreground layer. Pixels with the same distance in both fields form medial axis (see Figure 2.19d). From distances assigned to those pixels median is computed and treated as a one half of the overall

outline thickness. Using this value the outline can be refined by adding remaining pixels that have distances from the selected regions smaller than the estimated thickness (see Figure 2.19e). Finally the region together with corresponding outline can be easily extracted (see Figure 2.19f).

2.2.8 Background reconstruction

In the proposed framework the key assumption is that the background layer is static and that the camera motion is restricted only to planar panning and zooming. Moreover, since the foreground layer is dynamic and usually does not appear on the same position during the sequence, it is possible to reconstruct a large portion of the background layer by combining visible fragments of background from different animation frames.

To accomplish this task, visible fragments have to be registered. Variety of image registration techniques can be used (for survey see [196]). According to experiments performed on real cartoon animations with textural backgrounds, feature-based approaches (namely `SIFT`-keys [16, 109]) proven to be useful. Their main advantage is robustness to occlusions (due to superimposed foreground layer), scale changes (due to camera zooming), luminance fluctuation and vignetting (due to inhomogeneous lighting conditions). The robustness can be further improved since only three parameter translation-scale model is necessary (Figure 2.20). However, when the background layer is nearly homogenous or contains repetitive patterns `SIFT`-keys tend to fail and it is better to use robust area-based methods such as [124] that are sensitive to occlusions but thanks to background pre-classification the foreground parts of the image can be omitted from estimation.

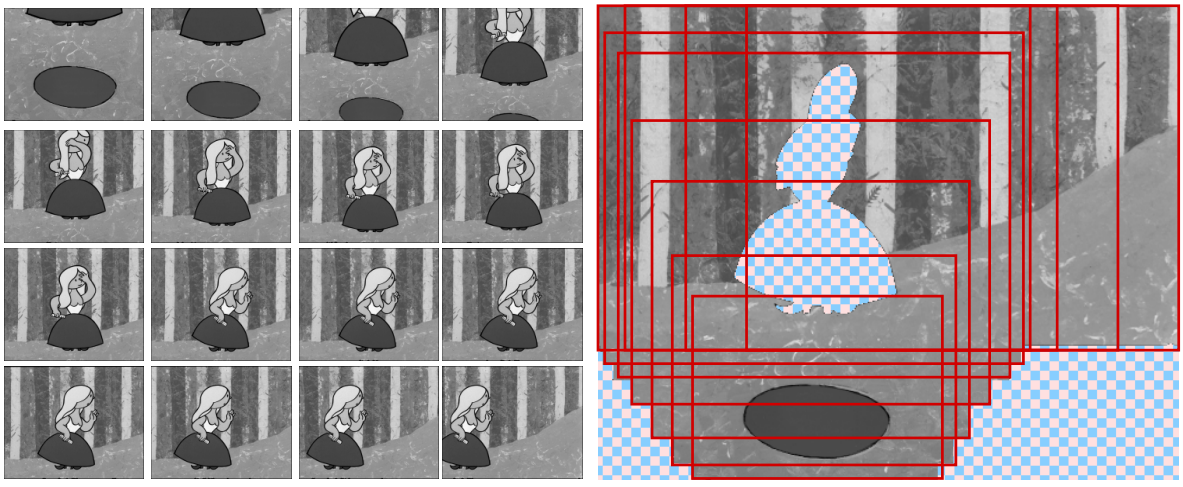

Figure 2.20: **Background reconstruction**: the original image sequence (left), reconstructed portion of background layer including original frame positions (right).

After the registration, pre-classified background regions are stitched together using *Poisson image editing* [128]. This technique allows to unify different levels of luminance caused by variable light conditions and image vignetting.

2.2.9 Vectorization

In example-based synthesis of cartoons (Section 4) the original foreground layer is rendered in different deformations/resolutions and thus the visual quality may suffer from sampling artifacts. This problem can be avoided by converting the shape of each region from raster to vector representation. Afterwards only resolution independent vector images are processed. Vector representation is also crucial for video compression (Section 5) where it allows to lower encoding bit-rates.

For the conversion an implementation of a standard *contour tracing algorithm* [181] is used. This technique extracts topology-based depth ordering of layers containing only simple regions without holes (see Figure 2.21). Afterwards a chain of *Bézier* cubics is fitted to the shape of each region using adaptive sub-division and least square fitting. By rendering *Bézier* regions in a correct depth order the original image can be easily reproduced (see Figure 2.21).

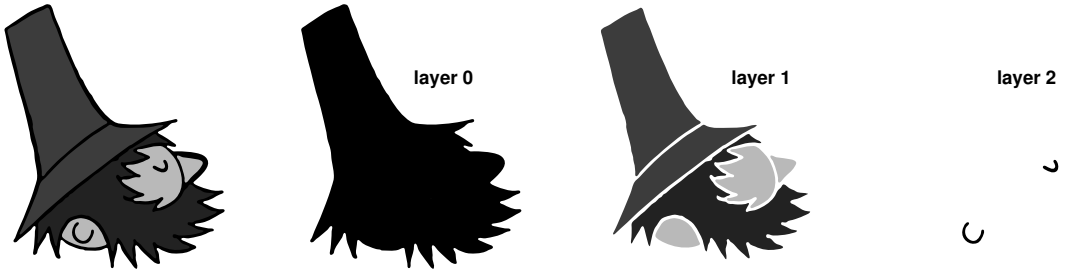

Figure 2.21: **Topology-based depth ordering of region layers**: the original image (left) is arranged into layers (right). Each layer contains several simple regions (without holes) that can be represented by single boundary curves. By drawing layers in a correct depth order the original image can be easily reproduced.

The key advantage is that the proposed framework performs precise preprocessing of the input image where all regions have constant color and boundaries are refined with sub-pixel accurate $\mathbf{L} \circ \mathbf{G}$-negative response (see Figure 2.22b). This allows to produce state-of-the-art vectorization (see Figure 2.22c) that significantly outperforms professional tools such as *VectorEye* [171] (see Figure 2.22d). Such output become useful especially when one wants to create high-quality prints from low-resolution footage (see Figure 2.23).

2.2.10 Experiments

To evaluate the proposed segmentation algorithm several experiments have been performed on real cartoon images from classical cartoon *O loupežníku Rumcajsovi* created by *Radek Pilař* in 1967. The original black-and-white negative has been scanned using *Spirit 2K DataCine* in the resolution of `720x576`.

The processing pipeline was the following: first the outline detector with flood-fill based classification has been used to extract outlines. Then regions where labelled and pre-classified to foreground and background layer using region area size thresholding and homogeneity test. Using pre-classified parts of background the background layer was reconstructed and then background subtraction applied to refine the classification of regions and outlines.

Figure 2.22: **High-quality vectorization**: (a) the original image, (b) result of the proposed sub-pixel accurate segmentation algorithm, (c) vectorization based on this segmentation, (d) vectorization using *Vector Eye*.

Final results are presented in Figures 2.30, 2.31, and 2.32. Each row represents one experiment where the input image sequence is foreshadowed by four thumbnails and currently processed frame. The output consists of the corresponding vectorized foreground layer and part of the reconstructed background layer. Results are divided into the three groups of five different image sequences to represent various levels of difficulty:

- The first group (Figure 2.30) stands for an easy set where the background layer is almost homogenous, contrast of outlines is high and all region shapes have only low frequency content that can be represented in the given resolution without aliasing. In such a favorable scenario resulting vectorization output is of high-quality without noticeable artifacts. Simple flood-fill based outline classification is sufficient for this type of images. The problem here is only wrong classification of regions. There are two typical examples when a large portion of background was not reconstructed from the given sequence and thus the background/foreground classification is not perfect (see small regions between legs in the 3rd and 5th row).
- The second group (Figure 2.31) contains more complicated backgrounds with visible textural information. The contrast of outlines is moderate and region shapes contain higher frequencies. Nevertheless segmentation and vectorization is still compelling. Sometimes outline waving and sharp-shaped artifacts arise due to coalescent textural information

in the background layer (see Rumcajs's leg in the 2nd row and Don Miracles's nose in the 5th row). A new artifact arising in this group is outline fusion at high-frequency shape details (Rumcajs's eye in the 1st row and soldier's curls in the 3rd row). It is also interesting to notice that even if the underlaying part of the background layer is reconstructed successfully the foreground/background classification is still not accurate. Several small regions are classified incorrectly due to significant bias of the sum of absolute differences caused by outline anti-aliasing (e.g. small regions between Manka's hands and skirt in the 2nd row).

- The third group (see Figure 2.32) represents images with highly textured background layer where the contrast between foreground and background is low. Outline waving and sharp-shaped artifacts are also visible (see e.g. Rumcajs's hat and Manka's body in the 1st row and dragon's neck in the 2nd row, etc.). In the last example background subtraction has been used to eliminate outlines in the background layer.

In the context of colorization (Section 3) discussed vectorization artifacts are not so important since only coarse pixel-level precision is needed to produce high-quality outputs. In the video compression application (Section 5) shape artifacts usually disappear since the foreground layer is superimposed back on the original background layer. Much more sensitive is example-based cartooning (Section 4) where parts of the original foreground layer are placed on a different background and possibly distorted. In this case smoothness of the curve can be locally edited by the user when required. The outline fusing artifact can be avoided by scanning the original film negative at higher resolution.

2.3 Correspondences

In this section correspondence retrieval between animation frames and regions is discussed. This additional information will be later useful especially for color-to-region assignment prediction (Section 3) and also for video compression (Section 5) to lower encoding bit-rate by retrieving redundant animation frames and reusing their shapes.

2.3.1 Frame-to-frame correspondences

In classical cartoon animations each frame is typically used more than once. To alleviate tedious manual inbetweening artists usually decide to reduce frame rate. Usually only each second frame is different and for long animations repetitive sequences of same frames are used (see Figure 2.24).

To retrieve redundant frames *phase correlation* [94] can be used first to quickly align down-sampled thumbnails of already processed outline-extracted frames. From this set the best matching candidate is selected by computing a map of *absolute differences* followed by *distance transform* [13] to obtain *weighted difference map* where large changes are emphasized against small global shifts (see Figure 2.25). Finally the frame with the lowest sum of weighted differences is selected. When this sum falls under a user-specified threshold the frame is considered as a copied instance of the frame in the database otherwise it is used as a best matching reference for the following region-to-region correspondence retrieval.

Figure 2.23: **Creating high-quality prints from low-resolution footage**: sub-pixel accurate vectorization allows to reproduce the original low-resolution artwork (left top) in professional print quality (right). In this example the output has been used to create the title page of the *Pixel* magazine (left bottom).

2.3.2 Region-to-region correspondences

Correspondence retrieval between regions of non-rigid cartoon drawings has been extensively studied in the context of *computer assisted auto-coloring* [111, 28, 154, 132, 134, 135, 133]. Here an issue is to reduce the amount of manual intervention in *cel painting* – a challenging variant of colorization where only hand-made outline drawings are available. For such

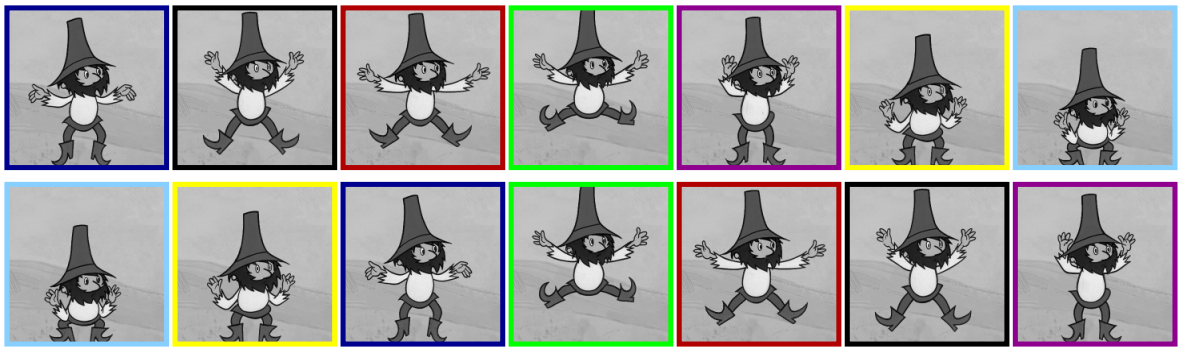

Figure 2.24: **An example of animation sequence where frames are used more than once.** The same frames are emphasized with the same color rectangle.

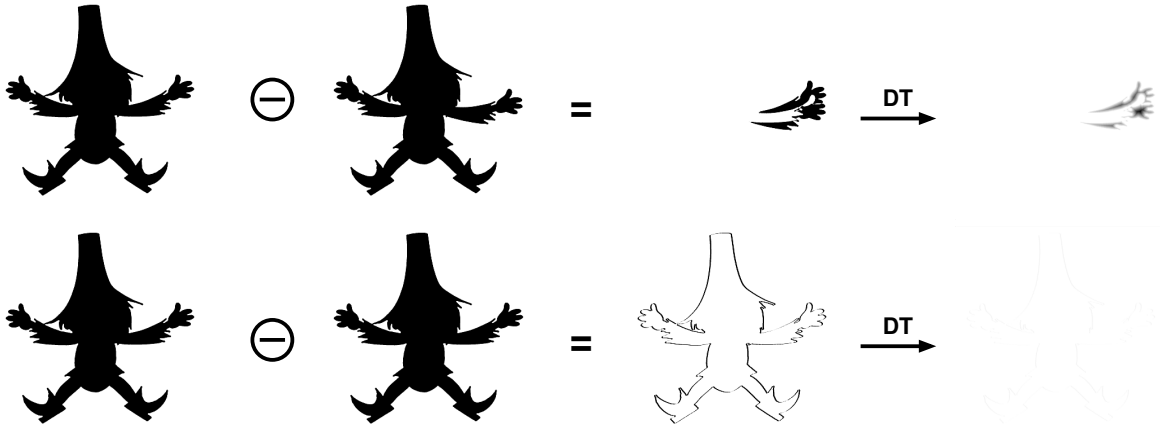

Figure 2.25: **Similarity metric for measuring frame-to-frame correspondences**: first a map of absolute differences is computed between outline-extracted frames (left), then the distance transform (DT) is used to emphasize large changes against small global shifts (right). By summing only the difference map it is usually not possible to distinguish small global distortions/shifts (bottom) from real movements (top).

scenario typical suggestion is to use region shape, size, position, and topology to estimate region correspondences. However, such features are sensitive to occlusions and topology variations imposed by virtual depth (see Figure 2.26), consequently the relative number of correct matches is moderate. Similar complications arise also in the context of general shape matching [36, 9, 59, 49]. Point correspondence retrieval on non-rigid shapes under occlusion is known to be difficult problem.

The situation becomes simpler when regions contain additional information such as luminance or color. A simple descriptor based on the median of luminance or on the mean color can dramatically decrease the number of false matches. However, in practice similar luminance/colors are used for many regions and thus the problem reduces to the cel painting scenario.

To address limitations of previous approaches a new structural matching scheme is introduced. Its main purpose is to avoid usage of global occlusion sensitive features such as region shape and topology. Only local structural similarity and local neighborhood relations are used in

Figure 2.26: **Examples of common structural differences between consecutive frames.** Red arrows indicate important shape and topology variations.

the estimation. This apparent restriction allows to estimate reasonable correspondences also for partially occluded or distorted regions.

2.3.2.1 Structural similarity

In order to compare local structural similarity of the input source and target animation frames, two sets of salient curvature points and junctions – *feature points* F_k^s and F_l^t (blue letters in Figure 2.27) are localized using *Kanade-Lucas-Tomasi* detector [169].

Structure of each source and target feature point is represented via square *patch* P_k^s and P_l^t (red letters in Figure 2.27) that cover local neighborhood of the corresponding feature point F_k^s and F_l^t.

For each source and target *region* $R_i^s \in \mathcal{R}^s$ and $R_j^t \in \mathcal{R}^t$ (white letters in Figure 2.27) a set of associated patches $\mathcal{P}_i^s = \{P_k^s : P_k^s \cap R_i^s \neq \emptyset\}$ and $\mathcal{P}_j^t = \{P_l^t : P_l^t \cap R_j^t \neq \emptyset\}$ is assigned (white lines in Figure 2.27, $P \cap R \neq \emptyset$ denotes non-empty spatial intersection between patch P and region R).

Using sets $\mathcal{P}_i^s$ and $\mathcal{P}_j^t$ the structural similarity $\texttt{Sim}(R_i^s, R_j^t)$ of any combination of the target and source region R_j^t and R_i^s can be estimated using the most similar mapping between corresponding sets of associated patches ($\mathcal{P}_i^s$ and $\mathcal{P}_j^t$):

$$\texttt{Sim}(R_i^s, R_j^t) = |\mathcal{P}_i^s| \cdot \left(\sum_{P_k^s \in \mathcal{P}_i^s} \min_{P_l^t \in \mathcal{P}_j^t} \texttt{Diff}(P_k^s, P_l^t) \right)^{-1}. \tag{2.8}$$

In (2.8) an issue is the implementation of function $\texttt{Diff}(A, B)$ which stands for the metric to compare patches A and B in terms of their structural similarity. This metric should be

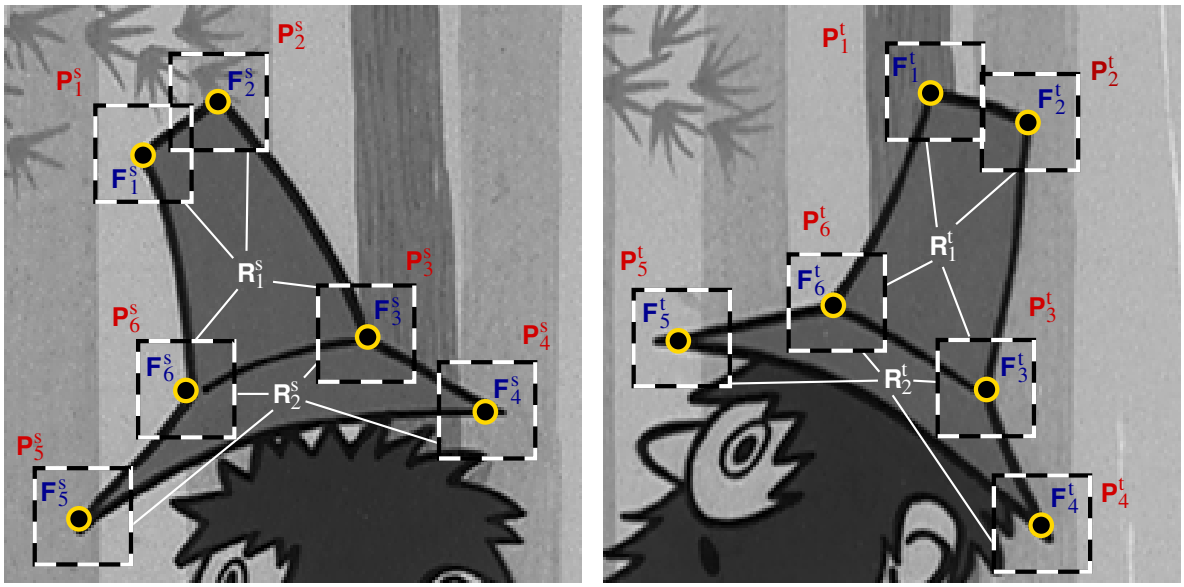

Figure 2.27: **Source and target regions** (R_i^s and R_j^t, white color) **with couple of associated patches** (P_k^s and P_l^t, red color) **covering structure of the corresponding feature points** (F_k^s and F_l^t, blue color).

discriminative, fast to compute, and also invariant to a reasonable subset of global deformations.

In case of classical cartoon animations rigid deformation model is sufficient since non-rigid changes can be modelled locally by translation, rotation and scale. Moreover, scale can be also omitted since it usually depends only on the camera field-of-view. Whenever camera zooming is detected during the background reconstruction phase it is later possible to resample the whole image and reach constant scale for all objects.

Following previous assumptions only translation and rotation invariant metric can be used:

$$\texttt{Diff}(A,B) = \min_{[x_0,y_0,\alpha]} \sum_x \sum_y \left(A(x+x_0, y+y_0) - \texttt{Rot}(B(x,y),\alpha)\right)^2 \tag{2.9}$$

where `Rot` denotes bitmap rotation.

2.3.2.2 Implementation issues

Although a brute force approach to evaluation of (2.9) guarantees exact value it is not tractable for interactive applications. To speed it up an approximation has to be used providing nearly exact value much faster.

A number of approximations have been tested, namely: *log-polar phase correlation* [137], *gradient descent* [110], *fast Fourier transform* [88] and *hierarchical block-matching* [119]. The best ratio between discriminability and computational overhead has been reached when each feature point was approximated by a set of rotation patches (see Figure 2.28, left). Over this set *hierarchical block-matching* together with *winner-update strategy* [33] has been used to found the best matching translation and rotation (see Figure 2.28, right).

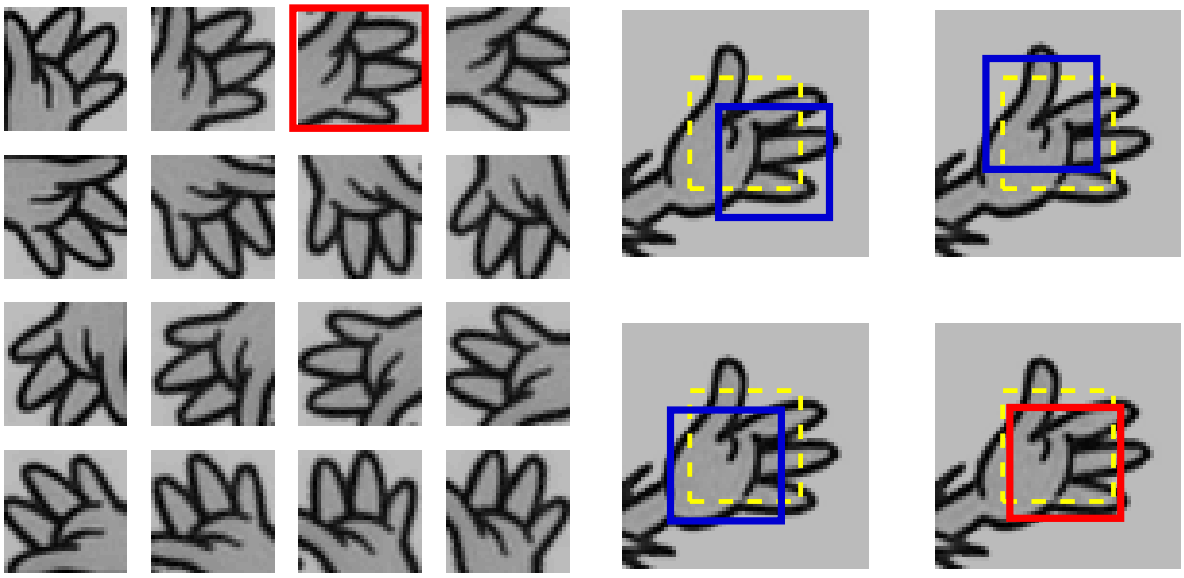

Figure 2.28: **Retrieving the best matching position and orientation**: a set of rotation patches generated from the neighborhood of the source feature point (left), each rotation patch is aligned using hierarchical block-matching (right), best matching rotation is selected and the corresponding sum of square differences is returned (red square).

Using optimized version of (2.9) it is still time consuming or even counterproductive to compute (2.8) for each source-target pair. When black-and-white or color cartoons are processed, it is reasonable to compare only regions that have similar median luminance or mean color. Also the region area size ratio is important. Experiments shown that it is undesirable to match regions when their potential counterparts are two times larger. Although small regions can represent visible parts of larger occluded regions they usually have not enough features to be classified correctly.

Another possibility how to speed up the computation of (2.8) is to exploit the fact that neighbor regions share one or more associated patches (see Figure 2.27). In such a case simple look-up table is used to store already computed structural differences (2.9). Anytime a comparison of the source and target patch is needed the look-up table is examined and when an appropriate value is found it is reused.

2.3.2.3 Neighborhood relations

When all feasible region-to-region similarities (2.8) are estimated it is possible to express the probability that the region R_i^s corresponds to the region R_j^t (denoted $R_i^s \triangleright R_j^t$) by normalizing the structural similarity $\texttt{Sim}(R_i^s, R_j^t)$ as follows:

$$\mathbf{P}_0(R_i^s \triangleright R_j^t) = \texttt{Sim}(R_i^s, R_j^t) \cdot \left(\sum_{R_k^s \in \mathcal{R}^s} \texttt{Sim}(R_k^s, R_j^t) \right)^{-1}. \tag{2.10}$$

To incorporate local neighborhood into the estimation the prior probability (2.10) can be refined iteratively using probabilistic relaxation scheme inspired by [2]. This simple approach

infers posterior probability using the following equation:

$$\mathbf{P}_{n+1}(R_i^s \triangleright R_j^t) = \frac{\mathbf{P}_n(R_i^s \triangleright R_j^t) \cdot \mathbf{Q}_n(R_i^s \triangleright R_j^t)}{\sum\limits_{R_k^s \in \mathcal{R}^s} \mathbf{P}_n(R_k^s \triangleright R_j^t) \cdot \mathbf{Q}_n(R_k^s \triangleright R_j^t)}, \tag{2.11}$$

where

$$\mathbf{Q}_n(R_i^s \triangleright R_j^t) = \frac{1}{|\mathcal{N}_i^s|} \sum_{R_k^s \in \mathcal{N}_i^s} \sum_{R_l^t \in \mathcal{N}_j^t} \mathbf{P}_n(R_k^s \triangleright R_l^t) \cdot \mathbf{P}_n(R_i^s \triangleright R_j^t) \tag{2.12}$$

is the region compatibility function which expresses how feasible is the assignment $R_i^s \triangleright R_j^t$ in the context of the local neighborhood $\mathcal{N}_i^s$ of the source region R_i^s and $\mathcal{N}_j^t$ of the target region R_j^t (regions are in local neighborhood if they share part of the boundary outline, see Figure 2.29). After a few iterations of (2.11) when a relative difference between prior $\mathbf{P}_n$ and posterior $\mathbf{P}_{n+1}$ falls under a specified limit the maximum a posteriori (`MAP`) solution is used to assign the most probable source region to each target region. Although this approach does not guarantee to find a global minimum (NP-complete problem [54]), in most cases it produces better results than a simple `MAP` solution based only on the local structural similarity.

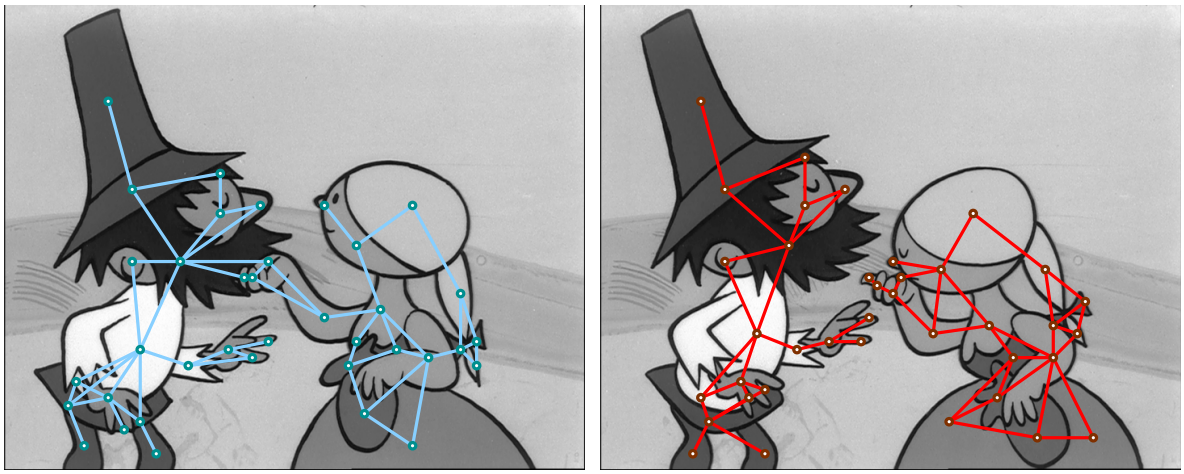

Figure 2.29: **Neighborhood relations between regions in the source** (left) **and in the target** (right) **image**. Regions are in local neighborhood if they share part of the boundary outline.

2.3.3 Experiments

In this section a performance of the proposed correspondence retrieval algorithm is evaluated. In all experiments scanned images from cartoon *O loupežníku Rumcajsovi* are used as in Section 2.2.10. Nine pairs of near consecutive frames were selected and the proposed image segmentation algorithm was used to locate and classify regions. For three selected pairs the original gray-scale information has been removed to simulate cel-painting scenario. In this case foreground/background classification based on the background subtraction is impossible and thus several background regions are considered as foreground.

Results are presented successively in Figures 2.33, 2.34, and 2.35. Each row represents one experiment. The source image is on the left and target image on the right side. In each

experiment 200 feature points have been detected (depicted as small red dots). Size of patches was `32x32` with `32` possible orientations and the luminance median threshold for pruning dissimilar regions was set to `16`. For the cel-painting scenario the allowable region area size ratio has been changed to `1.5x`. Each source region has assigned unique numerical label. Regions in the target frame are labelled according to estimated correspondences. By carefully examining experimental results it is possible to observe several interesting properties:

- When a local structure does not change significantly and when only a global transformations are applied, correspondences are almost perfect regardless region area size. However, when the local structure changes, smaller regions become sensitive to errors (e.g. regions 30, 31, 41, and 53 in Figure 2.33, middle row). By considering *Fitts'* law [53] from experimental psychology this phenomenon unfortunately complicate hand-driven corrections since hitting small regions is much more tedious than large ones.

- When two or more source regions have similar structure, one-to-many correspondences typically occur in the target image (e.g. Rumcajs's hands in Figure 2.34, bottom row and legs in top row). Such behavior can be advantageous for occluded regions. When they share local structural similarity with a fully visible region in the source image they can be labelled correctly thanks to possibility of one-to-many correspondences (e.g. boots in Figure 2.33, top row).

- In a challenging cel-painting scenario the tighter area size ratio provides a good preselection rule so the resulting classification is still usable (see Figure 2.35). However, when a large self-occlusion or region (dis)joining arises in the target image, several region pairs crosses the limit and wrongly estimated structural similarity can affect the posterior in a larger spatial context (e.g. Rumcajs's hand and gun in Figure 2.35, bottom row). In such cases correspondences based only on stand-alone local structural similarity may produce better results.

2.4 Summary

In this section a central part of the thesis – a novel cartoon analysis framework has been introduced. Its main purpose is to revert the process of layer composition and return the state before the final shot was taken. To accomplish this super-resolved outlines are extracted, homogeneous regions in the foreground layer are located, converted to vector representation, and the background layer is reconstructed from several observations. In the following phase global and local structural similarity with probabilistic reasoning over region neighborhood are used to estimate frame-to-frame and region-to-region correspondences. After that point the underlaying `2.5D` structure of the analyzed cartoon is partially known and opens a possibility to simplify various practical tasks. Some of them are discussed in next three sections.

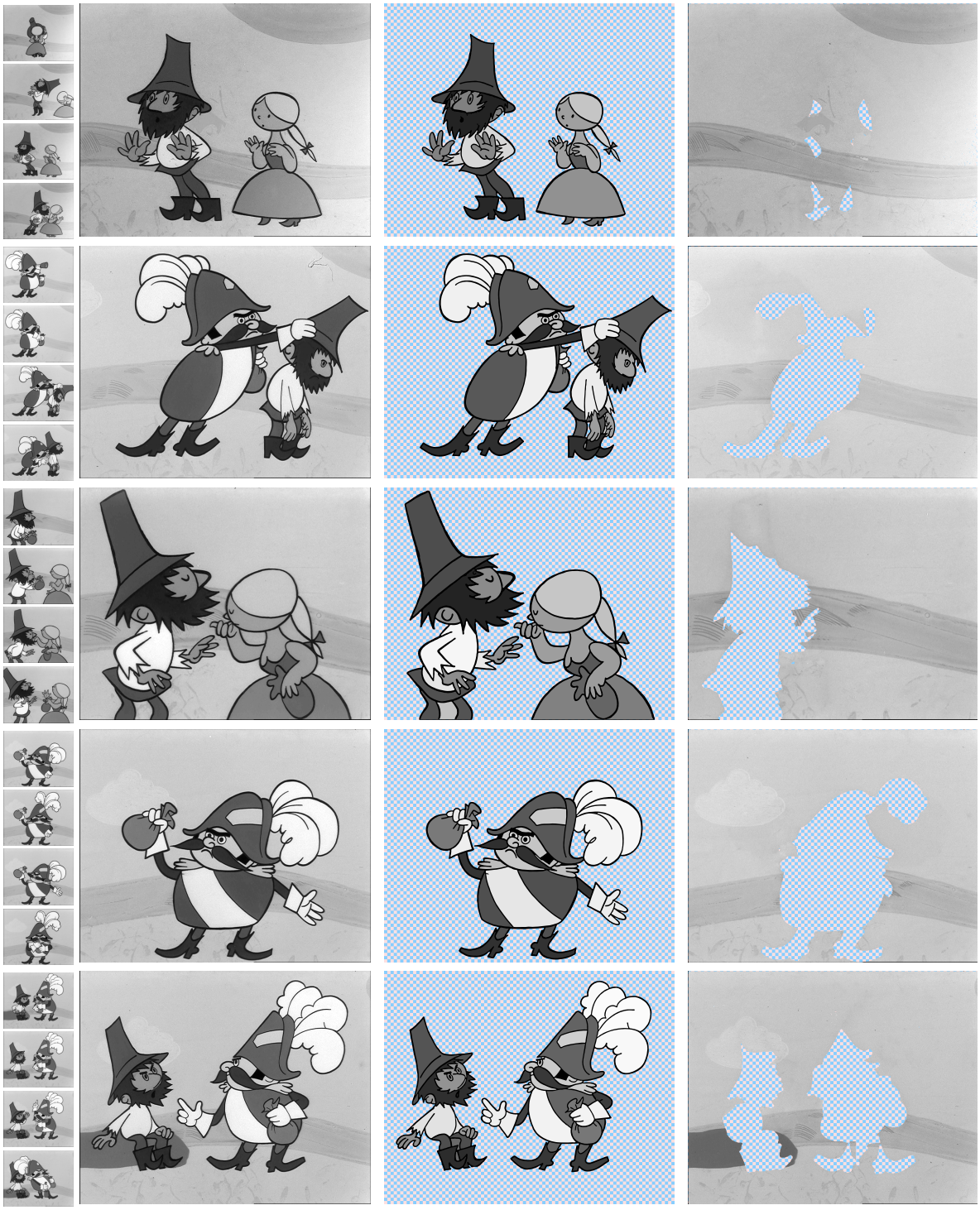

Figure 2.30: **Experiments – segmentation and vectorization (1)**: the original image sequence – four selected thumbnails and currently processed frame (left), vectorized foreground layer (middle), reconstructed background layer (right).

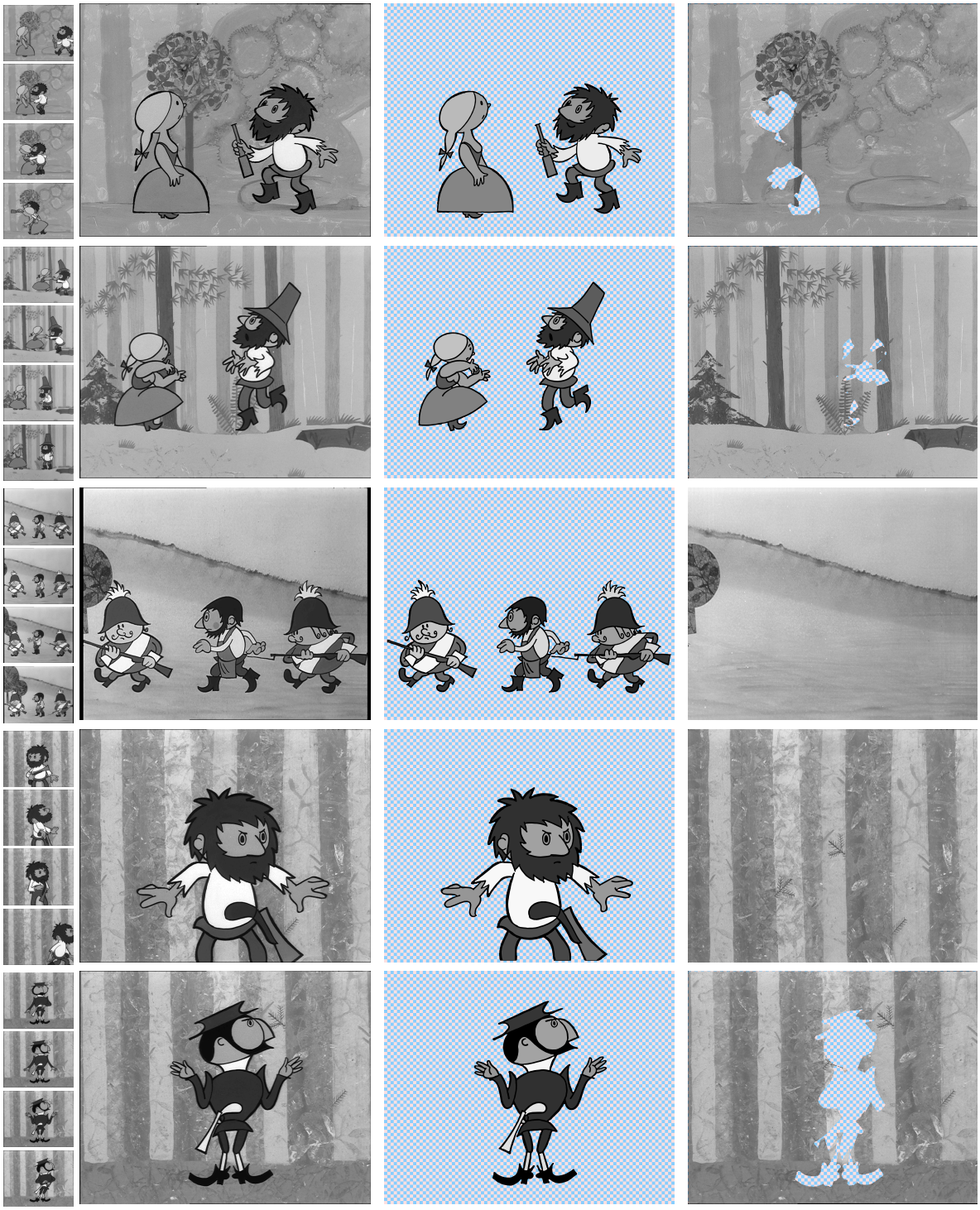

Figure 2.31: **Experiments – segmentation and vectorization (2)**: the original image sequence – four selected thumbnails and currently processed frame (left), vectorized foreground layer (middle), reconstructed background layer (right).

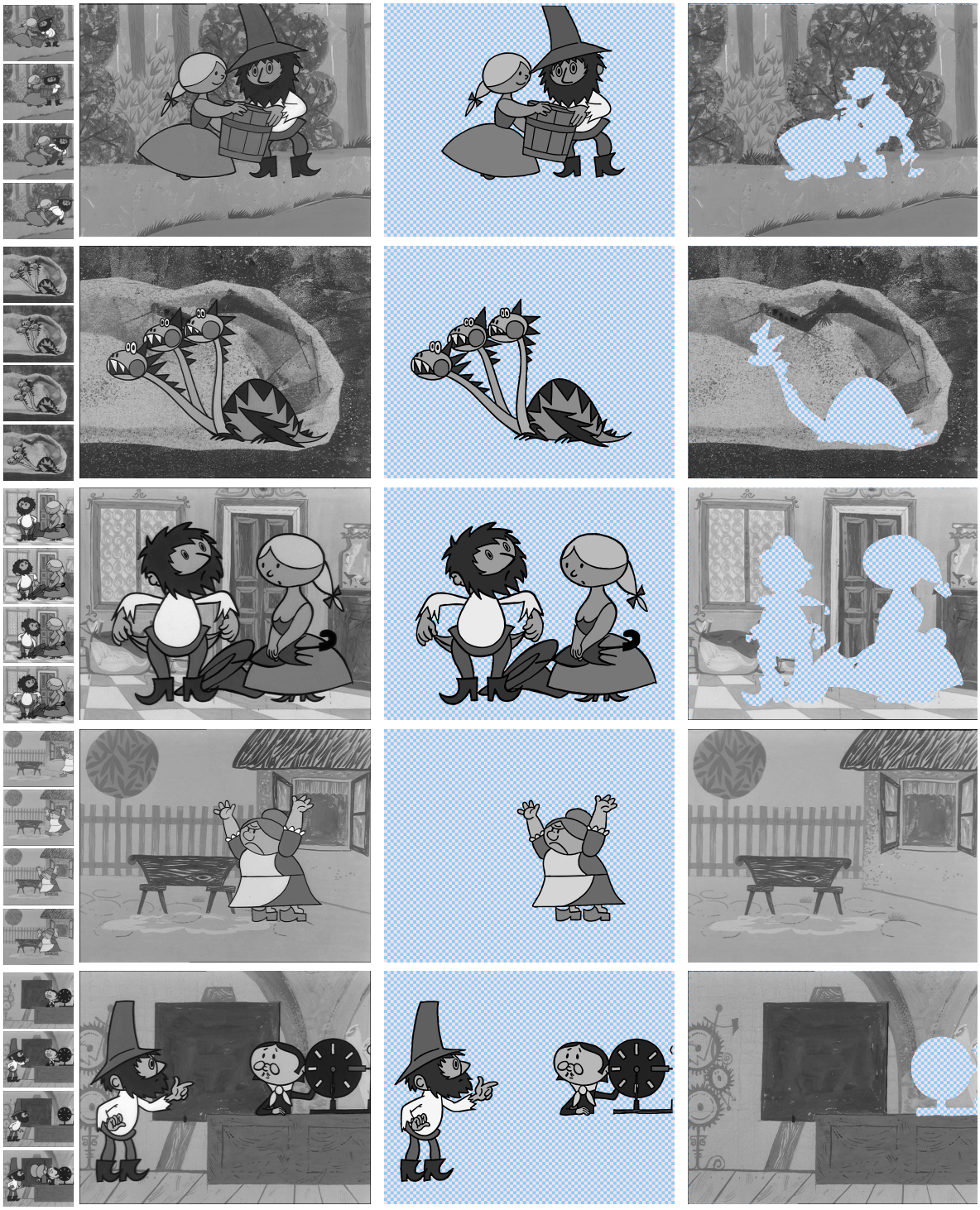

Figure 2.32: **Experiments – segmentation and vectorization (3)**: the original image sequence – four selected thumbnails and currently processed frame (left), vectorized foreground layer (middle), reconstructed background layer (right).

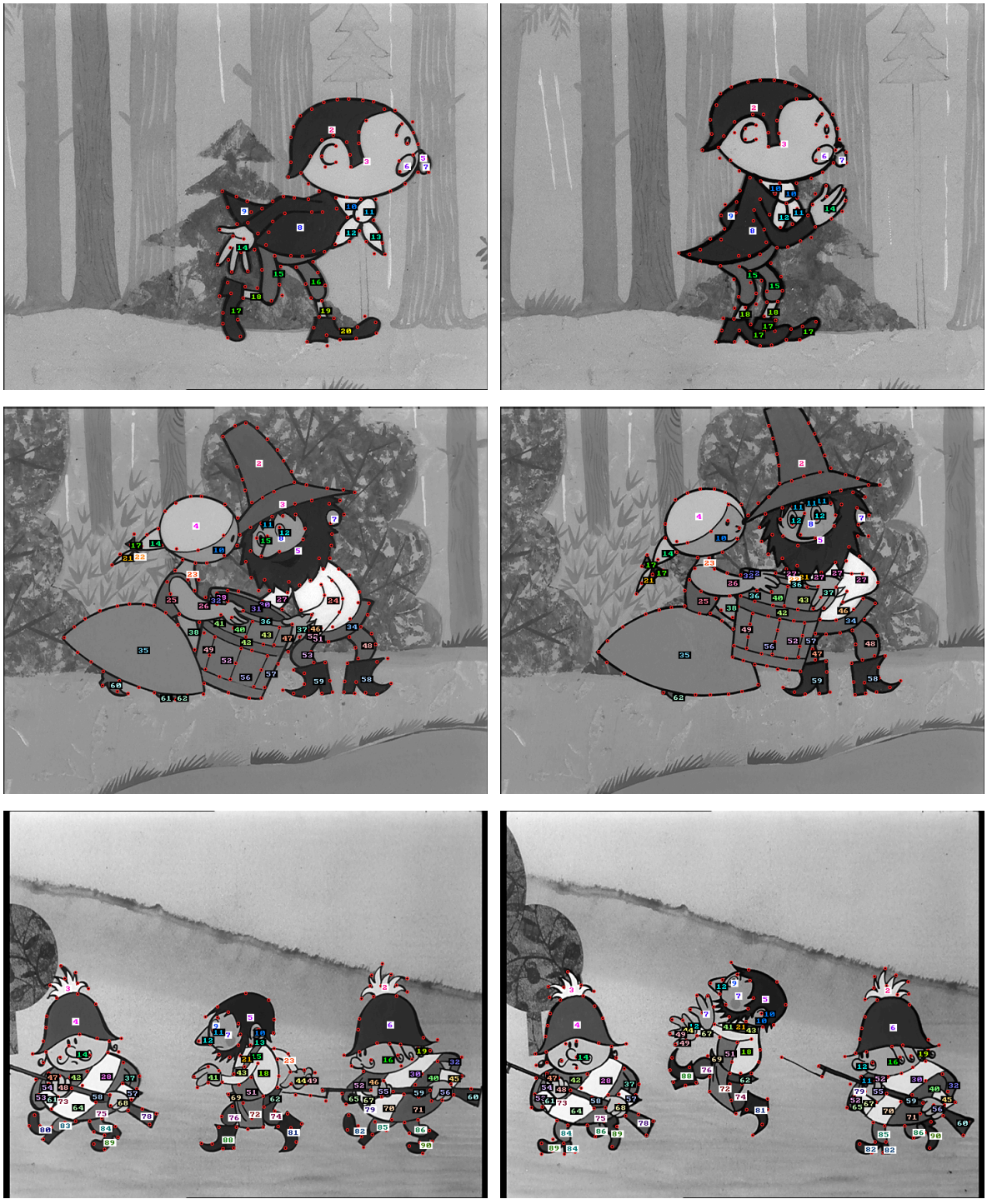

Figure 2.33: **Experiments – region correspondences (1)**: each region in the source image (left) has unique numerical label, regions in the target image (right) are labelled according to estimated correspondences. Features are depicted using red dots.

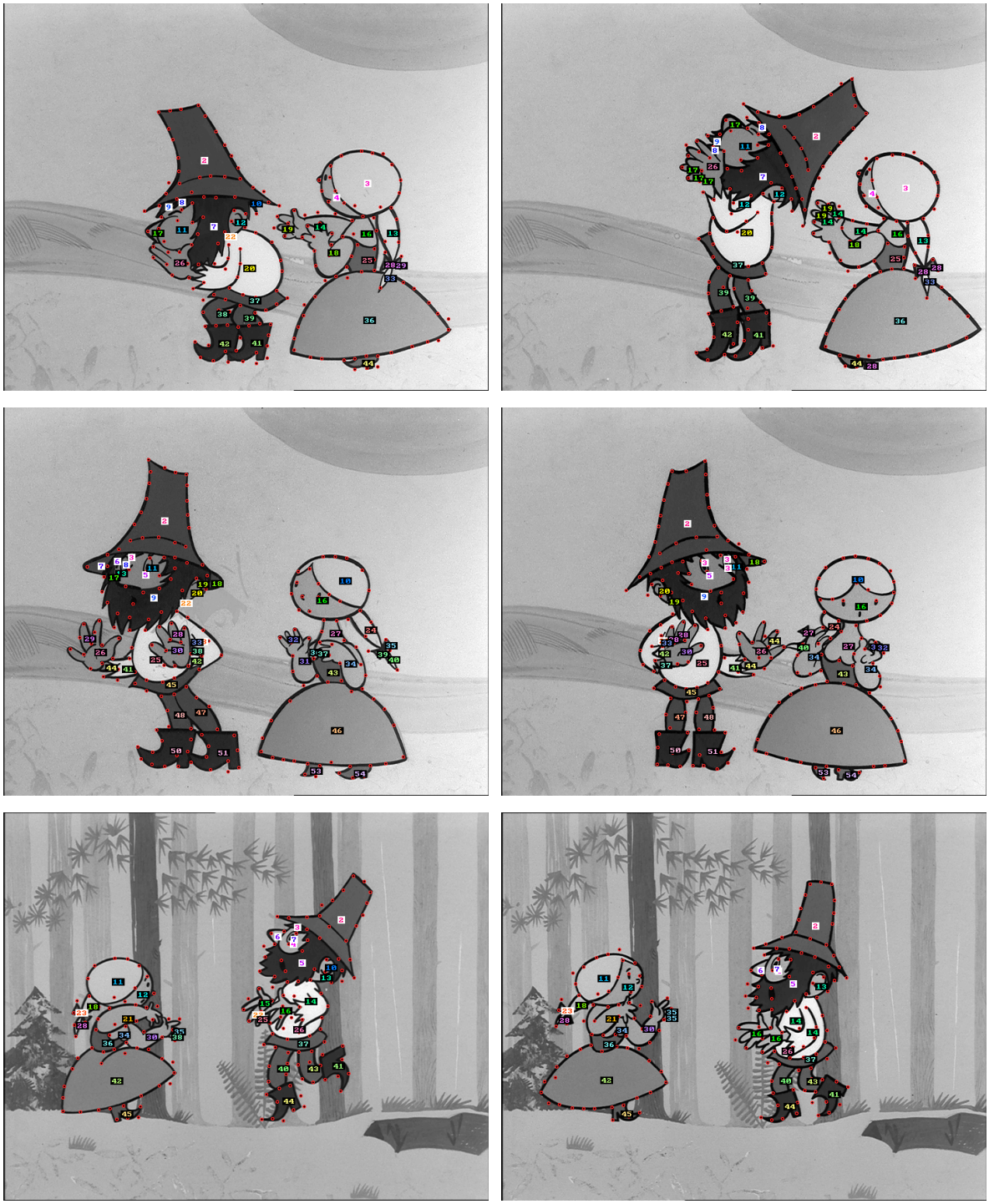

Figure 2.34: **Experiments – region correspondences (2)**: each region in the source image (left) has unique numerical label, regions in the target image (right) are labelled according to estimated correspondences. Features are depicted using red dots.

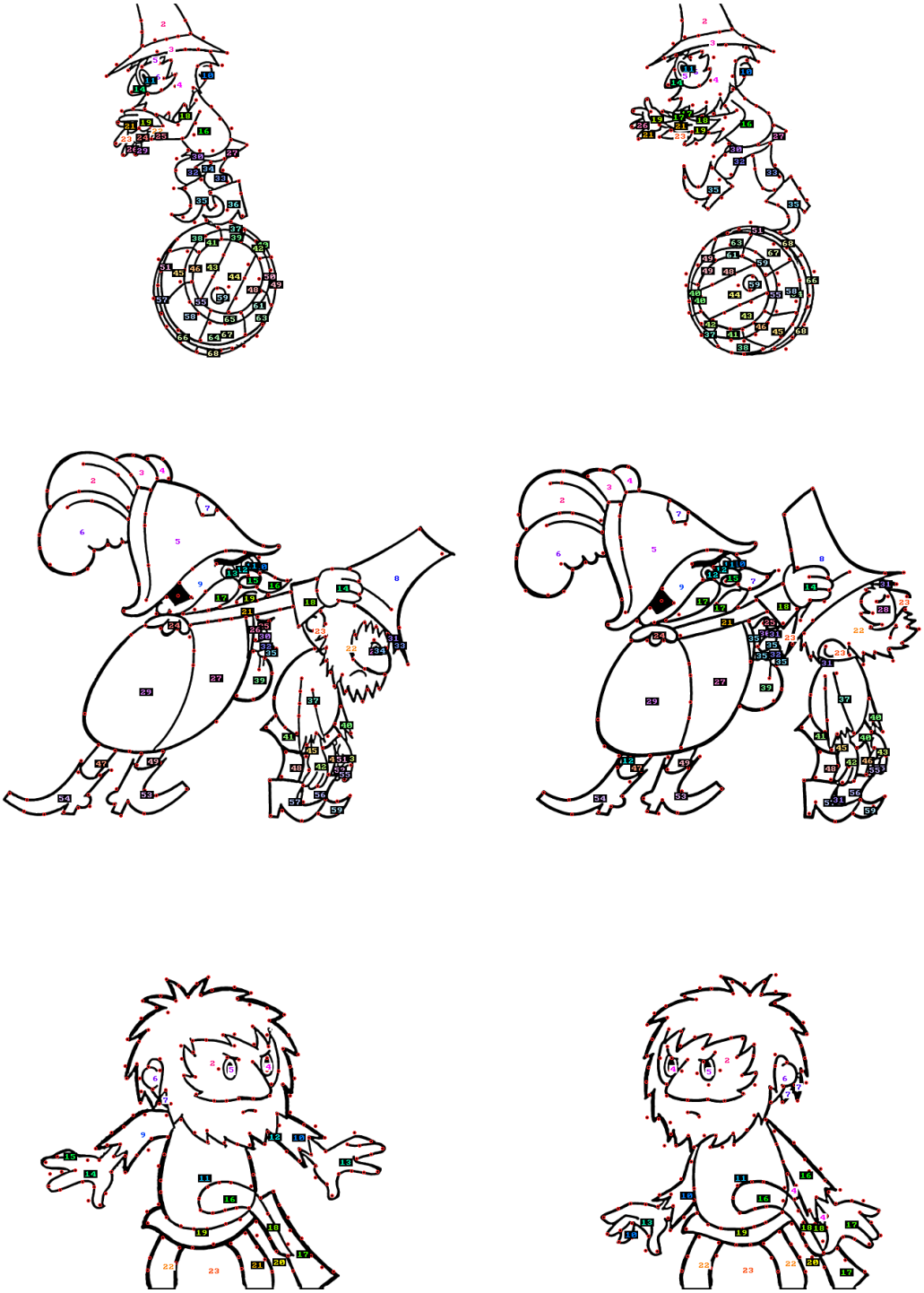

Figure 2.35: **Experiments – region correspondences (3)**: each region in the source image (left) has unique numerical label, regions in the target image (right) are labelled according to estimated correspondences. Features are depicted using red dots.

3 Application I: Colorization and restoration

In this section the proposed cartoon analysis framework is applied to the problem of colorization and color restoration of classical cartoon animations. First a detailed study of previous work on colorization is presented and later a novel approach is described together with experimental results.

3.1 Previous work

Colorization as a digital image processing problem has been studied since 1970 [112]. However, research in this field became popular until quite recently. During last five years number of semi-automatic approaches have been published. They exploit several heuristics to estimate color-to-luminance assignment automatically or with a partial user intervention. In this section these methods are described in more detail including description of main disadvantages to substantiate development of new approach.

3.1.1 Motion estimation

Historically *motion estimation* was the first published and patented approach to semi-automatic colorization [113]. Recently *Pan et al.* [126] and *Horiuchi et al.* [73] provided some improvements to this original method. The common workflow is to colorize key-frames manually and then propagate color information to the rest of the sequence using dense pixel correspondences given by the estimated optical flow.

Figure 3.1: **Colorization using motion estimation**: it is necessary to correct many pixels manually when the motion estimation failed.

However, usually lots of frames have to be colorized or corrected manually since motions and other rapid changes in the scene cause incorrect estimation of correspondences. This problem is common especially in the case of cartoon animation where the extent of motion is significantly large as compared to real-world movies (see Figure 2.26 and 3.1). Due to these circumstances optical flow based colorization remains tedious and labour intensive.

3.1.2 Luminance keying

Another well-known colorization technique called *luminance keying* (also known as *pseudo-coloring* [61, 99]) became popular for its simplicity. This method is a limited version of more

general technique – *chroma keying* frequently used for recoloring and color correction tasks in standard image manipulation or post-production tools. It utilizes user-defined look-up table to transform each level of luminance into a specified hue, saturation and optionally brightness (see Figure 3.2). In practice colors are blended in the table to reach smooth color transitions.

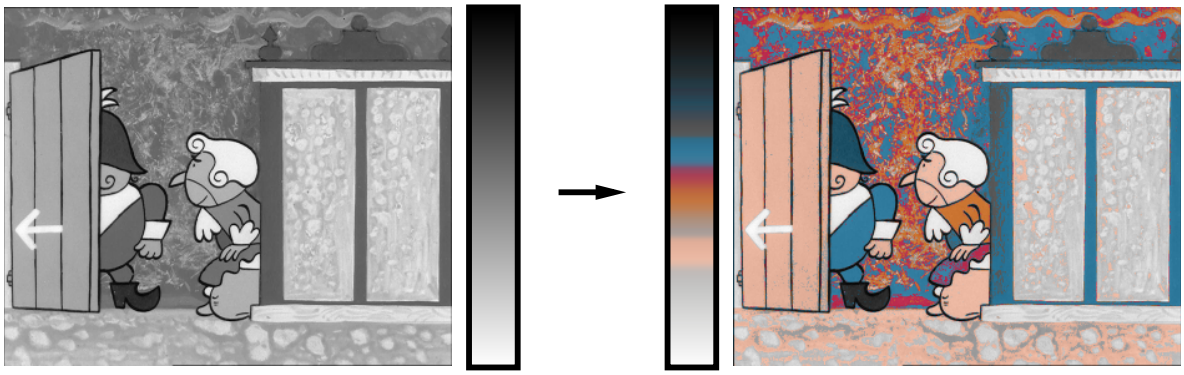

Figure 3.2: **An example of luminance keying**: the original gray-scale image (left), the look-up table that maps luminance into hue, saturation and brightness (middle), and the resulting colorization (right).

A crucial limitation of the luminance keying is inability to apply different colors on the pixels that have the same level of luminance but differ in a spatial location (see Figure 3.2). In practice tedious manual pre-segmentation is needed to overcome this limitation. Another problem arises when an adaptive noise, luminance fluctuation and/or image vignetting cause large derangement in the luminance. To address this issue the look-up table should be refined spatiotemporally which is rather tedious and time consuming.

3.1.3 Color-by-example

To partially overcome limitations of pure luminance keying *Welsh et al.* [184] proposed an example-based approach to colorization. Instead of pixel luminance they incorporates distribution of luminance from a local neighborhood of the given pixel. A similar idea has been used for color transfer between images [140] and in a popular framework of image analogies [67].

The method works as follows: first luminance and color components in the source and target image are separated via $l\alpha\beta$ color space [147] to minimize correlation between individual channels. Afterwards the luminance distribution from a small neighborhood of each target pixel is matched with a set of similar distributions in the source image. This set is selected either by jittered sampling or manually using predefined rectangular swatches. Finally chromatic information from the source pixel with the best matching distribution is transferred into the target pixel. The original luminance remains unchanged (see Figure 3.3).

Welsh's technique is surprisingly successful when applied to images that contain distinct textural information (see Figure 3.3). However, generally it still suffers from the same problems as luminance keying. When the target image contains several homogeneous regions then local luminance distributions collapse to a single peak with the same discriminative power as the look-up table used in the luminance keying. Color noise is another typical problem that arises

when two neighbor pixels in the target image are matched with two different source pixels that have similar luminance distribution but different color.

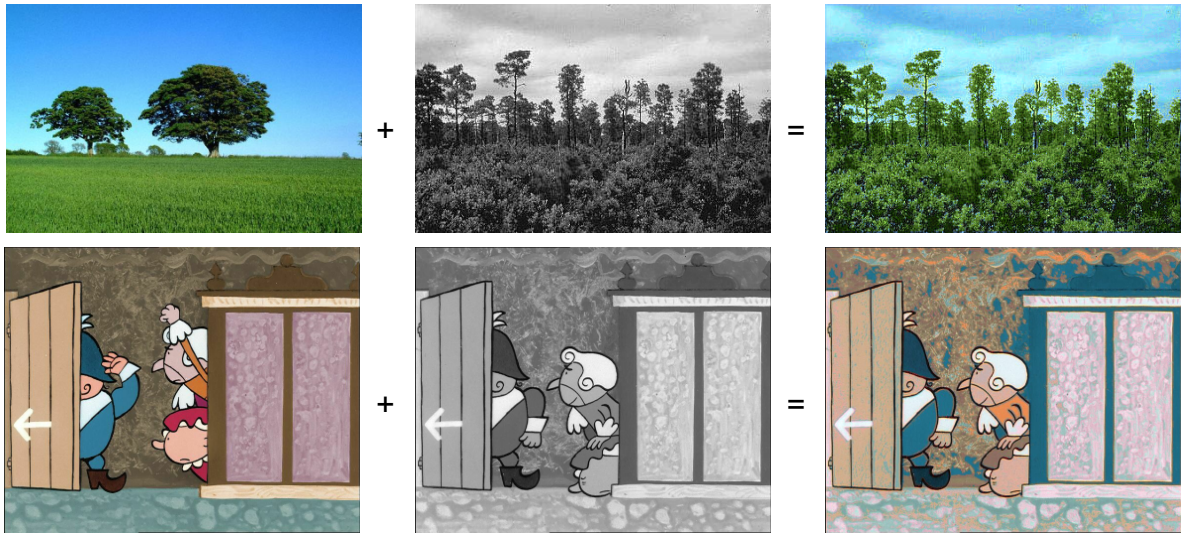

Figure 3.3: **Color-by-example in progress**: a natural scenario (top), cartoon image where several swatches have been used to perform the color transfer (bottom). Top images are from [184].

Several authors attempt to improve this technique. *Blasi* and *Recupero* [12] propose antipole method to retrieve best matching distribution. They report significant speed ups while the quality of colorization remain same or slightly worse as compared to the original technique. *Vieira et al.* [172] remove the need of manual selection of source images. For a given target image they automatically select an optimal source from a large database of annotated images. *Ji et al.* [81] suggest to use texture spectrum instead of luminance distribution to classify pixels. *Ying et al.* [191] added texture entropy to the feature vector to improve discriminability, `KD`-tree to decrease time complexity of retrieval of the best matching sample, median `k-NN` rule to reduce the influence of outliers, and median color filter to suppress color noise in the output image. Recently *Lipowezky* [106] incorporate advanced classification scheme based on textural features. His method is suitable especially for colorization of aerial images. However, despite of mentioned advances, the core idea remain unchanged therefore it is still impossible to distinguish large homogenous areas with similar luminance.

3.1.4 Segmentation

To overcome the fundamental limitation of locality, *Chen et al.* [32] used manual image segmentation to divide the original gray-scale image into a set of layers. Then for each layer the alpha channel is estimated using *Bayesian image matting* [35]. This decomposition allows to apply e.g. *Welsh*'s technique in each layer separately using different color sources. Final color image is reconstructed using simple alpha blending. However, this method brings no significant advantage as compared to tedious and labour intensive workflow commonly used in standard image manipulation tools.

Tai et al. [164] proposed different segmentation-based method that utilizes expectation maximization to automatically infer *smooth (probabilistic) segmentation* of the source and target image. Color information is then transferred directly between regions with similar luminance mean and variance. However, as in the luminance keying such a simple matching technique provide no possibility to distinguish regions with similar mean and variance.

3.1.5 Color inpainting

Another approach to colorization exploits well-known *channel dependency* property saying that homogeneity in the luminance indicates homogeneity in the color and vice versa [24].

Horiuchi [69, 70] was the first who adopted this premise. In his method color information is propagated from several seed pixels to the rest of the image. More formally, his aim is to minimize cost function E which penalizes color discontinuities over all pixels i in the target color image I:

$$E = \sum_{i \in I} \sum_{j \in \mathcal{N}_i} ||\mathbf{I}_i - \mathbf{I}_j||^2, \tag{3.1}$$

where $\mathcal{N}_i$ denotes neighborhood of a pixel i and $||\dots||^2$ is L_2-norm which measures color similarity in `RGB` color space. Proposed minimization is performed subject to following constraints: color of the seed pixel is fixed and other pixels must satisfy gray-scale conversion equation:

$$Y_i = (0.299, 0.587, 0.114) \cdot \mathbf{I}_i^t, \tag{3.2}$$

where Y_i denotes luminance of corresponding pixel in the original gray-scale image.

Horiuchi suggested to minimize E using *probabilistic relaxation* [90]. In this method each gray-scale pixel i has assigned a set of candidate colors $\mathcal{I}_i$ satisfying gray-scale conversion equation (3.2). Initially the probability of the color-to-luminance assignment $P_i^0(\mathbf{c})$ is equal for all colors in this set except in the seed pixels where it is set to 1 for the color defined by the user and 0 for other colors. Final solution is obtained by iterating over all pixels using the following relaxation formula:

$$P_i^{t+1}(\mathbf{c}) = \frac{P_i^t(\mathbf{c}) \cdot Q_i^t(\mathbf{c})}{\sum_{\mathbf{k} \in \mathcal{I}_i} P_i^t(\mathbf{k}) \cdot Q_i^t(\mathbf{k})} \tag{3.3}$$

where $Q_i^t(\mathbf{c})$ is a neighbor color compatibility function defined as follows:

$$Q_i^t(\mathbf{c}) = \sum_{j \in \mathcal{N}_i} \max_{\mathbf{l} \in \mathcal{I}_j} \left\{ P_j^t(\mathbf{l}) \cdot \left(2 - \frac{2||\mathbf{c} - \mathbf{l}||^2}{\sqrt{3}(|\mathcal{I}_i| - 1)} \right) \right\}. \tag{3.4}$$

Practical usability of *Horiuchi*'s method is strongly limited by the number of candidate colors $|\mathcal{I}_i|$. For common `8-bit` representation this number becomes extremely large. *Horiuchi* tried to overcome this using `4-bit` quantization, but also in this case the number of possible candidates can exceed 400, therefore still too much for practical application. Despite of this limitation *Horiuchi* performed several experiments on very small images using parallel implementation. They confirm that relatively small number of seed pixels is required to reconstruct the original color image (see Figure 3.4).

To overcome large computational overhead *Horiuchi* and *Hirano* [71] later proposed a much faster approximation similar to well-known flood-fill algorithm. In this method color information is propagated directly from already colorized pixels towards its color-free neighbors. During the propagation color is modified accordingly to the original luminance by selecting candidates which satisfy equation (3.2) and minimize following term:

$$\mathbf{I}_j = \arg\min_{\mathbf{c}\in\mathcal{I}_j} ||\mathbf{I}_i - \mathbf{c}||^2. \tag{3.5}$$

Although this propagation scheme is really fast, it introduces lots of visible artifacts that significantly reduce the final visual quality. *Takahama et al.* [165] tried to alleviate this issue by setting additional partitions that prevent unwanted propagation of color, however, their results are still not satisfactory.

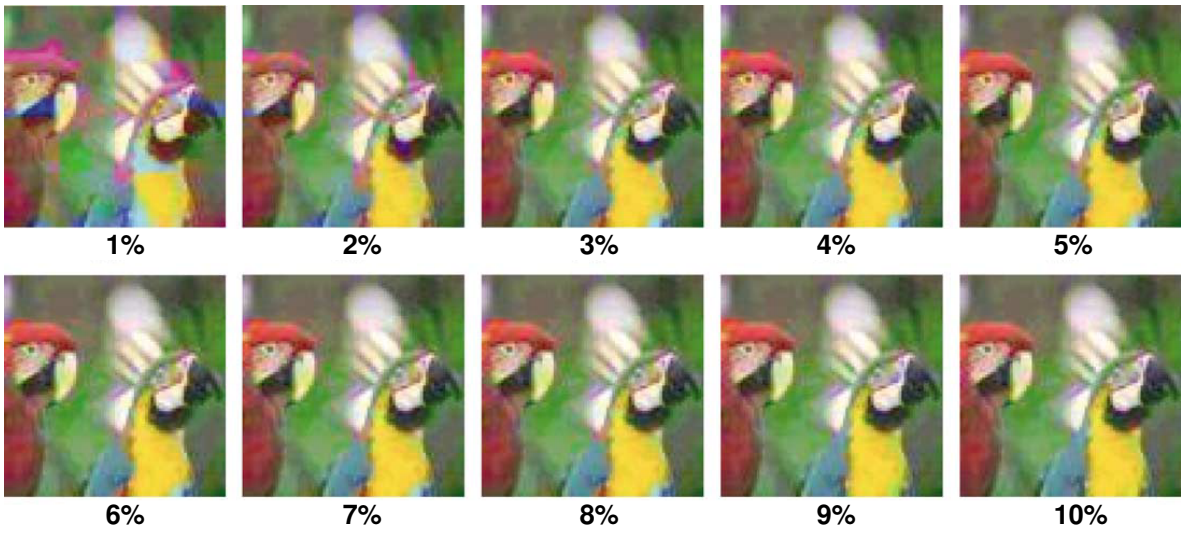

Figure 3.4: **Color propagation using probabilistic relaxation**: increasing visual quality as compared to increasing number of randomly selected seed pixels (from [70]).

Levin et al. [102] introduced another framework similar to *Horiuchi*'s. In their approach user-defined seed pixels are specified by a set of carefully placed color scribbles (see Figure 3.5). In contrast to *Horiuchi* they formulated the problem as a constrained weighted least squares optimization. The aim is to minimize difference between color components u and v (`YUV` color space is used here) in the pixel i and the weighted average of the same color components in the local neighborhood of the pixel i:

$$E = \sum_i ||u_i - \sum_{j\in\mathcal{N}_i} w_{ij}\cdot u_j||^2 + \sum_i ||v_i - \sum_{j\in\mathcal{N}_i} w_{ij}\cdot v_j||^2. \tag{3.6}$$

Accordingly to the assumption of channel dependency, two different *affinity functions* [182] are exploited to assign more weight to those pairs of pixels that have similar luminance:

$$w_{ij} \propto \exp\left(-\frac{(Y_i - Y_j)^2}{2\sigma_i^2}\right) \quad \text{or} \quad w_{ij} \propto 1 + \frac{1}{\sigma_i^2}(Y_i - \mu_i)(Y_j - \mu_i). \tag{3.7}$$

Mean μ_i and standard deviation σ_i are computed on a small neighborhood around the pixel i. These two affinity functions were later refined by local linear embedding [131], connectivity and distance factor [121], and by adaptive edge detection [74].

Resulting constrained optimization scheme yields a large sparse system of linear equations that is overdetermined therefore can be solved using number of standard least-squares solvers. *Levin et al.* used `Matlab`'s built-in solver which is good for profiling but very slow for real applications. Fortunately they also suggest to exploit fast multi-grid solver [130] that produces visually acceptable approximation in much shorter time frames (see Figure 3.5) though still not interactively.

Levin's framework is also suitable for image sequences. In this case standard motion estimation technique [110] is used to compensate movement between consecutive frames and then the color information is propagated both over the time and space.

Another possible extension of *Levin*'s framework is recolorization. Here the affinity function is computed using information from each color channel. It is only necessary to add a special type of scribble to mark pixels which should have the same color as in the original image.

Despite of the advances in the speed and precision, colorization using *Levin*'s method is still tedious and labour intensive by the reason that usually a large number of well-placed color scribbles is required to produce satisfactory results. *Irony et al.* [79] tried to overcome this limitation by combining ideas of *Welsh*'s approach with *Levin*'s. They developed new spatially coherent pixel classification scheme based on `DCT` features to transfer color information from pre-segmented source image. A plausibility of the resulting classification is deduced and pixels with high confidence are used as micro-scribbles. *Levin*'s framework is then used to produced the final colorization. Although authors present several interesting results, the usability of their approach is still strongly limited since the classification scheme is almost local and works only when the image contain well distinctive textural features.

Figure 3.5: **Color propagation using least-squares optimization**: user-defined color scribbles (left), *Levin*'s colorization (middle), original color image (right). Top images are from [102].

Sapiro [149] proposed another scribble-based color propagation scheme. In his approach in contrast to *Levin*'s framework the optimization is performed in the gradient domain. More formally, he minimizes following cost function:

$$E = \int_I ||\nabla Y_i - \nabla C_i||^2, \tag{3.8}$$

where $\nabla := (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$ is the *gradient operator*, Y is the original gray-scale image and C color component channel. This yields a large sparse system of *Poisson equations* (see e.g. [128]):

$$\triangle Y = \triangle C, \tag{3.9}$$

where $\triangle := (\frac{\partial^2}{\partial x^2}, \frac{\partial^2}{\partial y^2})$ is *Laplace operator* that can be approximated using simple forward differences as follows:

$$\triangle I(x,y) = I(x+1,y) + I(x-1,y) + I(x,y+1) + I(x,y-1) - 4I(x,y). \tag{3.10}$$

The solution of (3.9) can be obtained using number of known *rapid Poisson solvers* which exploit several speed up techniques to outperform classical *Gauss-Seidel* elimination or conjugate gradient approaches (see e.g. [55, 183]). Using rapid Poisson solver *Sapiro*'s method outperforms *Levin*'s framework in terms of computational overhead. The approach can be extended for image sequences using `3D` version of Laplace operator for which rapid Poisson solver also exists [142].

An interesting property of *Sapiro*'s method is that the minimization can be performed also in full `RGB` color space. This is possible since the solution of (3.9) preserves (in the least-squares sense) the original gray-scale gradient in each color channel. However, the problem is that the method is not based on discontinuity preserving energy function and so produces noticeable color bleeding artifacts (see Figure 3.6). This behavior is common for all pure least-squares approaches. Yet discussed extensions of *Levin*'s framework [121, 74] partially overcome this limitation using discontinuity preserving weighting factor allowing to change color more dramatically when the local magnitude of gradient in the original gray-scale image is large.

Noda et al. [123] formulates colorization as a maximum a posteriori estimation of a color image given a monochrome image modelled as *Markov* random field. They decomposed computationally prohibitive global inference into the several local `MAP` estimation problems leading to a constrained quadratic programming. *Zeng et al.* [193] presented another variational model that can be though as a combination of *Sapiro*'s and *Levin*'s frameworks. Unfortunately, authors of these two methods do not present sufficient number of experimental results to substantiate practical usability.

Recently, *Yatziv* and *Sapiro* [190] introduced a novel color inpainting scheme that outperforms previous approaches both in the speed and visual quality. In order to transfer color to the pixel i they compute *geodesic distance* [25] to each seed pixel j. This distance is defined as follows:

$$d_{i,j} = \min_{P_{i,j}} \int_{t=0}^{1} |\nabla Y \cdot P'(t)| dt \tag{3.11}$$

where $P_{i,j}$ denotes curve following one of possible paths between pixels i and j. The final color for the pixel i is computed as a weighted sum of colors stored in the seed pixel j:

$$C_i := \frac{\sum_j w_{i,j} \cdot C_j}{\sum_j w_{i,j}}. \tag{3.12}$$

Figure 3.6: **Color propagation in the gradient domain**: user-defined scribbles (left), *Sapiro*'s colorization (middle), original color image (right). Top images are from [149].

The weight $w_{i,j}$ is a function of $d_{i,j}$ computed as $w_{i,j} := d_{i,j}^{-r}$, where r is blending factor that controls smoothness of color transition. To speed up computation *Yatziv* and *Saprio* suggested greedy *best first* approach to approximate geodesic distance. The process starts at seed pixels and gradually expands towards neighbor pixels while in each turn the shortest geodesic is expanded first. Authors reported that such an approximation is sufficient and produces compelling results (see Figure 3.7). Nevertheless it is also possible to use more sophisticated algorithms to compute geodesic distance [170]. Another issue is connected with the number of different color scribbles. For each stand-alone scribble it is necessary to propagate information to the whole image. However, experiments show that it is only necessary to propagate color information from a few closest scribbles. This approximation does not produce noticeable artifacts but significantly reduces both time and memory complexity.

Similarly to other color propagation schemes also *Yatziv-Saprio*'s approach can be easily extended for image sequences. In this case the color is propagated both in space and time using `3D` geodesic distance. Moreover, *Yatziv* and *Sapiro* introduced simple extension that allows the user to change the original luminance. Using two different scribbles marked as background and foreground they annotate image and compute color propagation. This process results in an approximate alpha-matte that can be used for various image editing tasks such as color brightness manipulation.

Although *Yatziv-Sapiro*'s color propagation scheme produces visually compelling results in much shorter time frames, it is still necessary to draw many well-placed scribbles to produce good-looking colorization. This problem become really tedious when two neighbor regions have similar texture and share weakly contrasted boundary. In this problematic case scribbles have to be drawn extremely carefully since the visible color transition occurs actually near

Figure 3.7: **Color propagation using geodesic distance**: user-defined color scribbles (left), *Yatziv-Sapiro*'s colorization (middle), original color image (right). Top images are from [190].

the medial axis between scribbles. This is due to underlaying geodesics which in this case simply reduces to the Euclidian distance.

3.2 Colorization and restoration framework

In this section a novel approach to colorization and color restoration of classical cartoon animations is described. It is fully based on the cartoon analysis framework proposed in Section 2 that allows to significantly reduce the amount of manual intervention as compared to previous approaches. First an overview of the colorization pipeline is presented and later, in successive sections, details of each step are discussed including specific implementation issues and experimental results.

3.2.1 Framework overview

The key assumption in the proposed cartoon analysis framework is that animation frames are created as planar compositions of two layers (foreground and background) with considerably different visual appearance. Each layer has its pros and cons regarding colorization. The foreground layer contains only homogenous regions that seem to be easy to locate and colorize but the problem is that they are dynamic and so require frame-by-frame care. On the other side background is usually more complicated but the advantage is that it is static during the

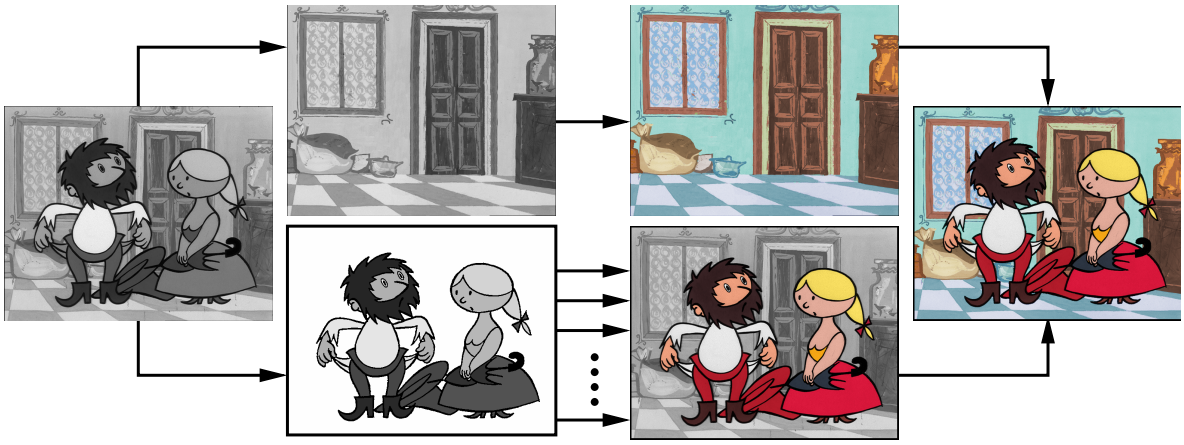

Figure 3.8: **Colorization pipeline**: a static background (top) is reconstructed from several animation frames. Color is then applied on the whole background layer at one snap using standard image manipulation tool. The dynamic foreground (bottom) is colorized semi-automatically frame-by-frame.

animation and so it can be processed only once and then reused in the whole sequence (see Figure 3.8). The aim of the proposed colorization pipeline is to exploit pros and alleviate cons listed above. To do so layers have to be detached and processed separately using custom tailored technique.

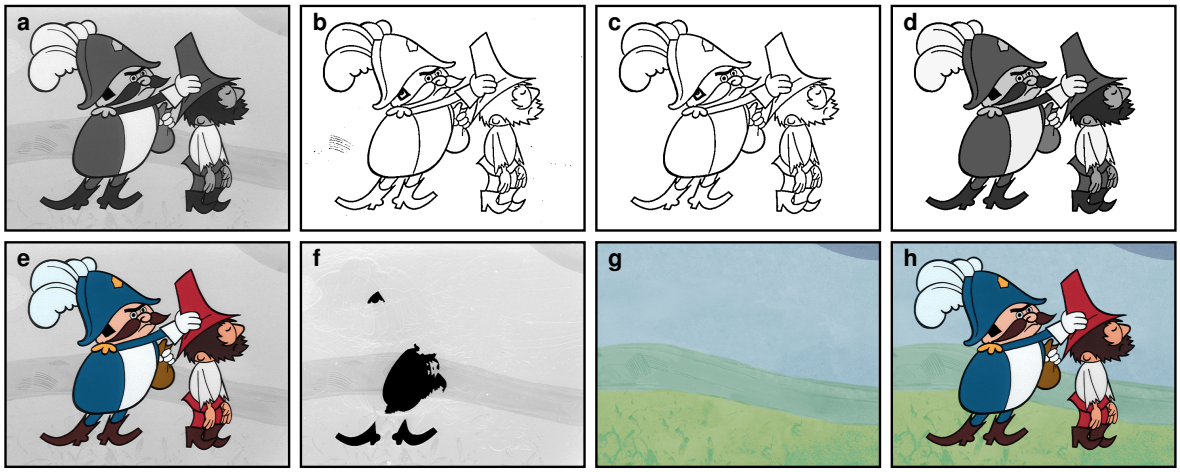

Figure 3.9: **Colorization in progress**: (a) the original gray-scale image, (b) outline detection, (c) outline extraction, (d) foreground layer segmentation, (e) foreground layer colorization, (f) background layer reconstruction, (g) background layer colorization, and (h) final composition.

In the first stage unsupervised image segmentation (Section 2.2) is used to divide the input frame (Figure 3.9a) into a set of regions (Figure 3.9b,c,d). In the next phase camera motion is tracked through the sequence and the visible portion of the background layer is reconstructed (Section 2.2.8, Figure 3.9f). Such an image is then colorized at one snap using color inpainting technique or any standard image manipulation tool (see Figure 3.9g and 3.10).

The dynamic foreground layer need to be colorized in the frame-by-frame manner. To speed up this tedious process frame-to-frame and region-to-region structural correspondences (Section 2.3) are exploited to predict proper color-to-region assignment for uncolored frames using already colorized frames as an example. During this step limited manual intervention avoids propagation of prediction errors into the consecutive frames and consequently guarantee the final visual quality.

Color image synthesis phase follows (Section 3.2.3). In this step hue and saturation from a user-defined palette are applied automatically on each pixel in the foreground layer. Brightness of the final color is modulated using the original luminance (see Figure 3.9e). Also dust spots and band scratches are removed in this phase automatically by exploiting the assumption on region homogeneity.

Finally, the knowledge about the camera movement allows to extract visible portion of already colorized background and make the final composition with colorized foreground layer (see Figure 3.8 and 3.9h).

3.2.2 Foreground layer colorization

After the pre-processing (Section 2.2) the structure of each animation frame is represented by a reference map where each pixel contains pointer to the underlaying region. Using this map the user can exploit standard point-and-click approach to assign desired color labels to regions.

When the first animation frame is processed it is necessary to mark out all regions. However, in proceeding frames frame-to-frame and region-to-region correspondences (Section 2.3) can be used to reduce the number of point-and-click operations. The advantage is that the color prediction can be helpful even if the region correspondences are inexact.

As was discussed in Section 2.3.3 small regions are very sensitive to errors. One of the reasons of this phenomena is that for small regions no distinct set of patches can be associated and so the similarity metric (2.8) does not provide enough discriminative power to distinguish them. To reduce the number of wrong predictions region-to-region correspondences are used only to estimate the spatial context. For more precise color transfer technique called *patch pasting* is proposed.

3.2.2.1 Patch pasting

The idea of patch pasting is to transfer color information pixel-by-pixel using small color patches extracted around the salient points in the source image and pasted on the best matching location in the target image (see Figure 3.11). Such a location can be predetermined using already estimated region-to-region correspondences therefore only local matching between the sets of associated patches can be performed. Such hierarchical approach provide better robustness as compared to global feature matching since it incorporates larger spatial context and neighborhood relations to prune set of possible candidates.

In more detail patch pasting is implemented as follows: from each example feature point (see Figure 3.12a) two rectangular patches are extracted. The first one contains the original luminance and second one the associated color information. After the extraction the best

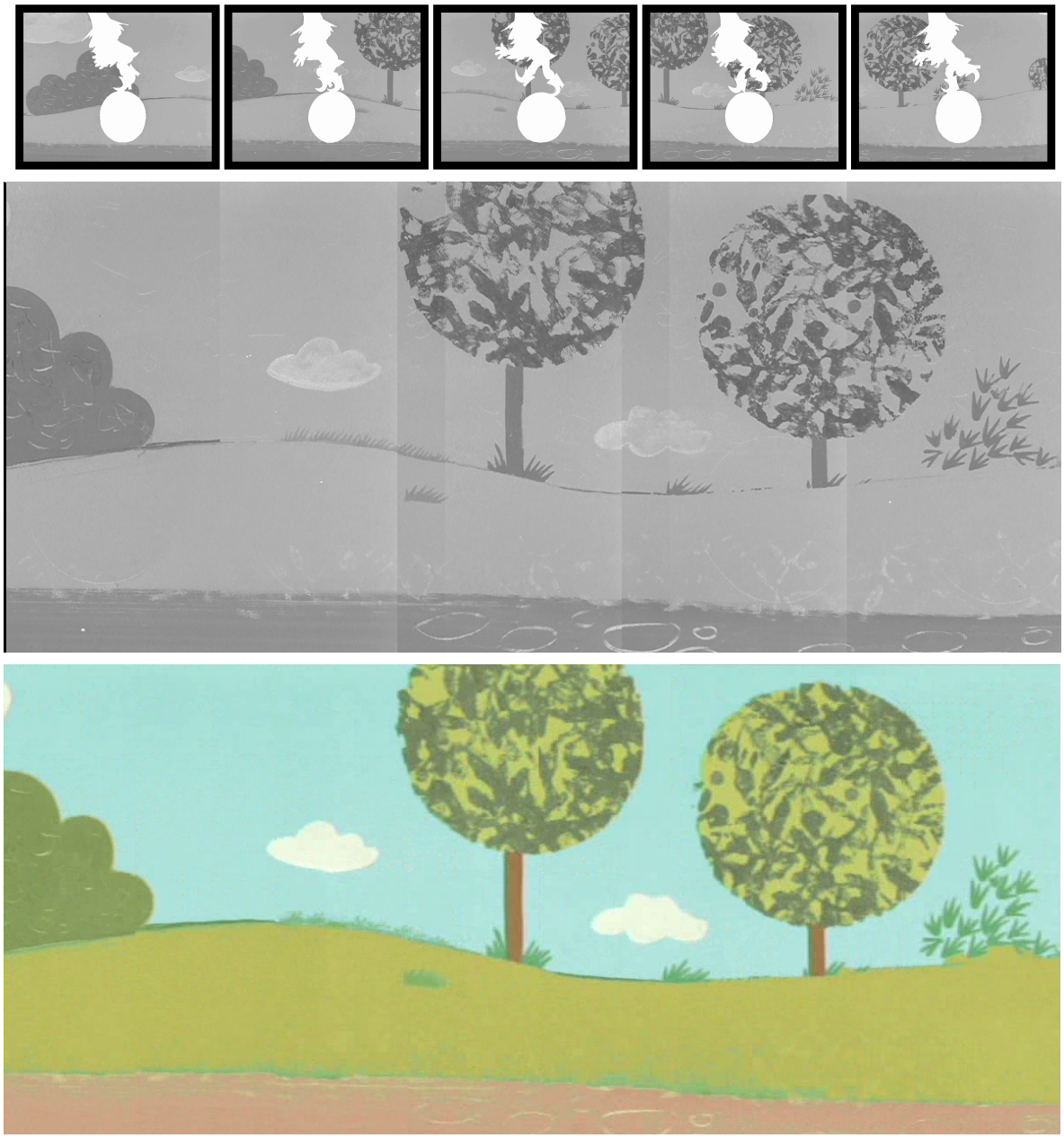

Figure 3.10: **Background layer colorization**: camera motion estimation through the original sequence (top), intermediate gray-scale background reconstruction (middle), final background reconstruction followed by manual colorization (bottom).

matching translation and rotation is searched using the luminance patch as in Section 2.3.2.1. Such transformation is then applied to the example color patch that is afterwards pasted on a proper position in the target color buffer (see Figure 3.12b). To avoid possible random overlapping, associated patches are sorted in lexicographic order according to non-decreasing structural similarities (2.8) and (2.9) so that more suitable patches are pasted after inferior

ones. Ordering is also performed on the pixel level. In this case an analogy of well-known *z-buffer* called *quality buffer* (see Figure 3.12e) is used:

$$\begin{array}{lll} \textbf{if} & |I_e(x',y') - P_t(x,y)| < Q_t(x,y) & \textbf{then} \\ \{ & Q_t(x,y) = |I_e(x',y') - P_t(x,y)| & \\ & C_t(x,y) = C_e(x',y') & \} \end{array} \tag{3.13}$$

Before the pixel (x', y') (from the transformed color patch C_e) is pasted on the proper position (x, y) in the target color buffer C_t, the absolute luminance difference between the corresponding pixel in the transformed target patch P_t and in the example image I_e is computed. If such a difference is smaller than the actual value stored in the quality buffer Q_t, color label is pasted on the target color buffer and then the value in quality buffer is updated. Otherwise, the original color label and difference remain unchanged. Finally, when all patches are pasted, non-maxima suppression over each region is used to select the most frequent color label (see Figure 3.12c).

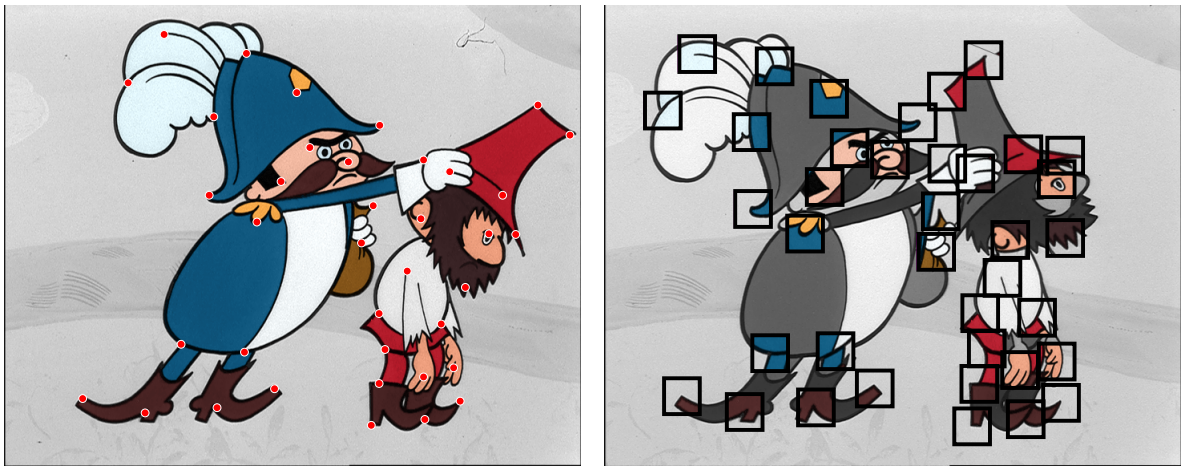

Figure 3.11: **Patch pasting (main idea)**: yet colorized foreground layer with superimposed feature points (left), target frame after patch pasting (right).

3.2.3 Color image synthesis and restoration

In this section the final stage of the colorization pipeline is presented. It will be shown how to apply color information on the gray-scale image and how to seamlessly compose foreground and background layers. Additionally a simple dust spot removal technique and other restoration steps are discussed.

3.2.3.1 Color brightness modulation

Previous works on colorization assume that only the color components are modified and the brightness remain unchanged. However, the problem is that especially in cartoon animations the brightness defined by an artist usually does not correspond to the luminance in the original gray-scale image. When a user wants to apply brighter color on a dark region and vice versa,

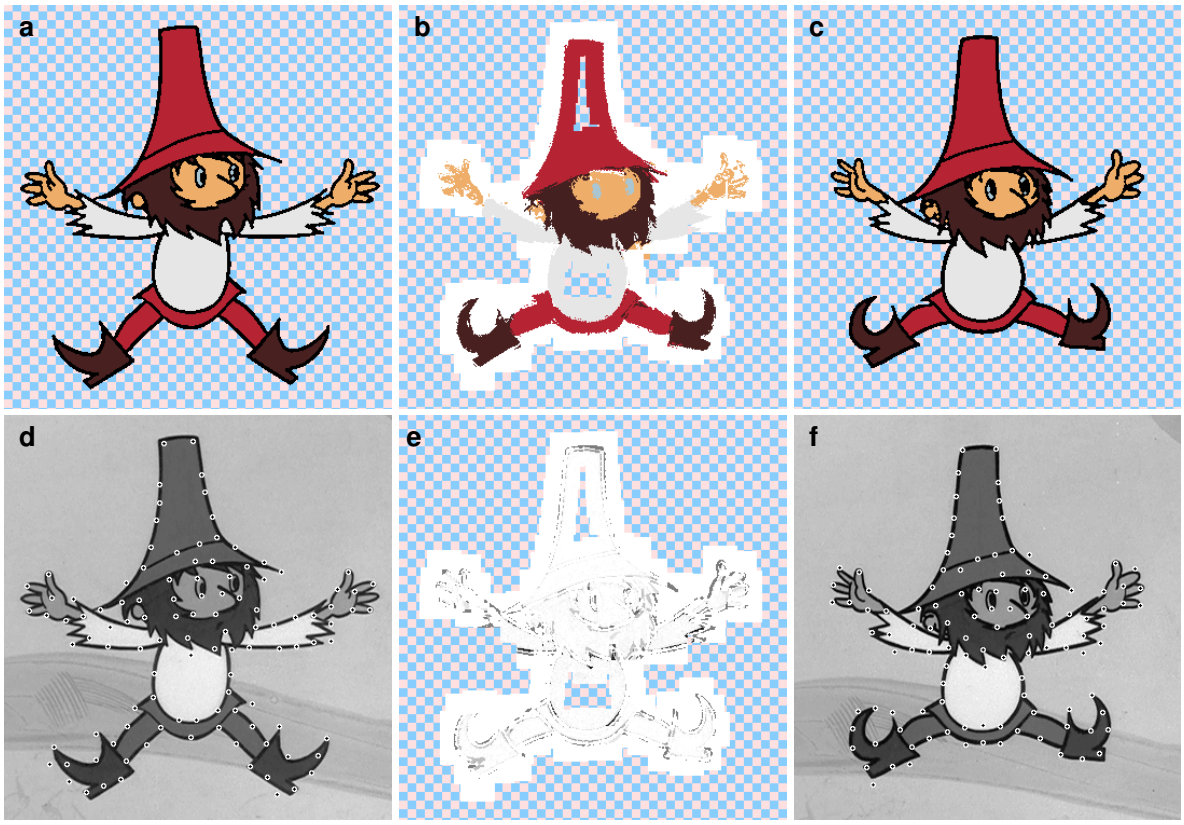

Figure 3.12: **Patch pasting (in detail)**: (a) source color buffer, (b) target color buffer after patch pasting, (c) target color buffer after non-maxima suppression, (d) source image with superimposed feature points, (e) quality buffer, (f) target image with superimposed feature points.

it is necessary to either shift or scale the original luminance to reach the required level of brightness. However, luminance shifting produces step changes and scaling empowers noise scattering inside the region. Both artifacts are visually disturbing (see Figure 3.13). One possible solution to this problem is to estimate the alpha channel of the region boundaries as in [32, 190], manipulate the brightness and then reconstruct the color image through alpha blending. However, the problem is that alpha estimation is usually slow and produces lots of artifacts when the contrast between the region and outline is negligible.

In the proposed framework an approximation to alpha estimation is used to produce acceptable results much faster. The main idea is to use smooth transition between additive and multiplicative brightness modulation. Brightness of pixels near the region boundaries are multiplied and interior pixels vary color brightness via additive modulation. This technique can be formulated as follows:

$$C = (1 - \alpha)\left(C_0 + L - \hat{L}\right) + \alpha C_0 L/\hat{L}, \tag{3.14}$$

where C_0 is the user-defined color, $\hat{L}$ represents the region luminance median, L is luminance of the current pixel in the original gray-scale image, and α is spatially driven blending factor

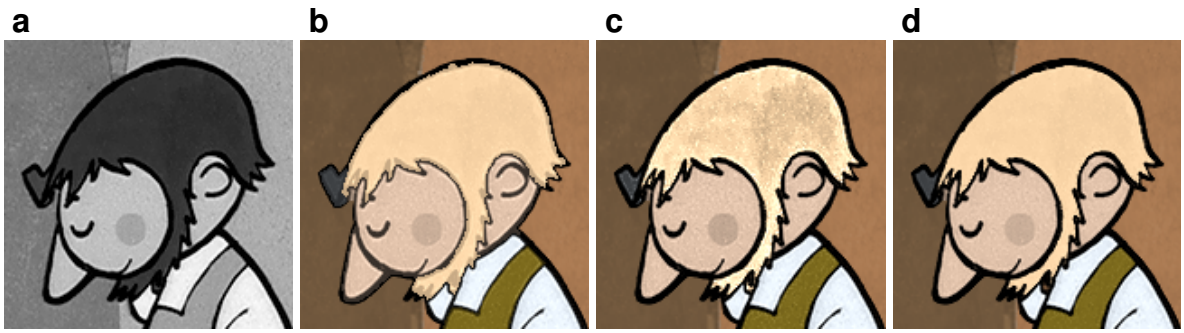

Figure 3.13: **Changing color brightness**: (a) the original gray-scale image, (b) brightness shifting produces step changes, (c) brightness scaling empowers noise, (d) proposed approach.

that allows to smoothly combine additive and multiplicative modulation. It can be estimated for each pixel by slightly blurring original outlines (see Figure 3.14).

3.2.3.2 Layer composition

Similar technique is used for seamless compositions of foreground and background layers. In general, the original background layer is not homogeneous, therefore it is necessary to compute luminance median in a small neighborhood (e.g. `7x7`) around the current pixel. For median computation only pixels that do not belong to detected outlines and foreground regions are considered. In contrast to (3.14) the original color in the background layer is blended with its scaled version using the following formula:

$$C = (1 - \alpha)\, C_b + \alpha C_b L / \hat{L}_b, \tag{3.15}$$

where C_b denotes the color of the actual pixel in the background layer, L is the luminance of the current pixel in the original gray-scale image, and $\hat{L}_b$ is the luminance median of pixels from a local neighborhood of the current pixel. See Figure 3.14 for results of the proposed background and foreground color modulation.

3.2.3.3 Restoration

Thanks to the background reconstruction and region homogeneity assumption, well-known degradations such as dust spots, band scratches, vignetting, and luminance fluctuation can be easily detected and suppressed. In the case of reconstructed background layer all degradations are removed at one snap in a single image. In the foreground layer luminance fluctuation and vignetting are removed automatically since mean of color brightness remain fixed as results from (3.14). Also dust spots and band scratches can be removed automatically by assigning region luminance median $\hat{L}$ to all interior pixels. However, such simple technique produces well-known flat appearance that can be undesirable from an aesthetic point of view.

To preserve natural grain as much as possible, dust spots and band scratches are detected at pixels that have significantly different luminance as compared to region luminance median $\hat{L}$: $(L - \hat{L})^2 > \sigma_L^2$ (L is pixel luminance and σ_L^2 standard deviation of luminance inside the region). These outliers are restored using new luminance taken as a random sample from the

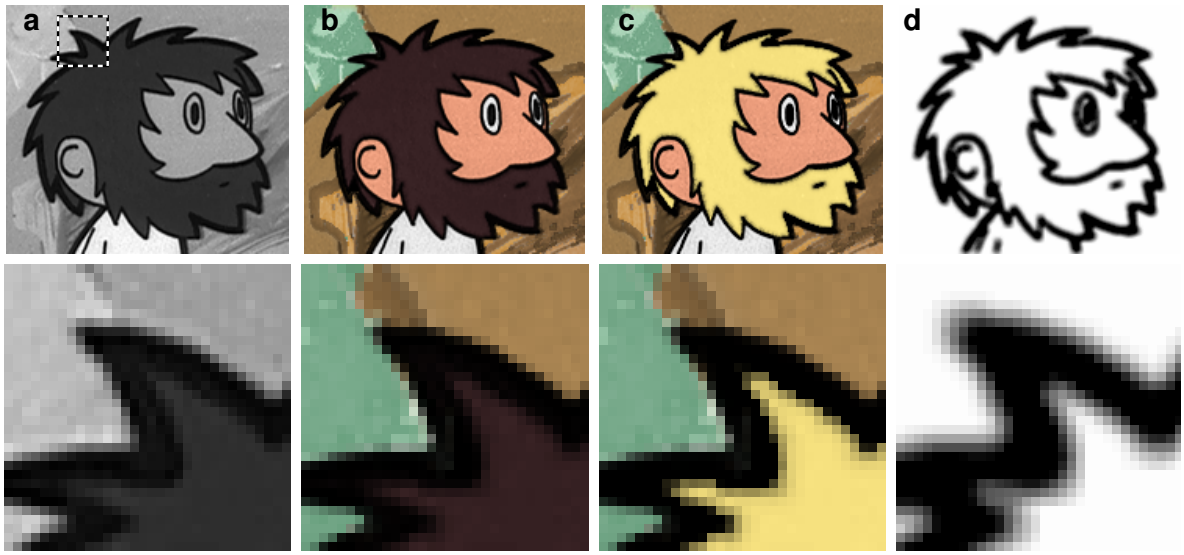

Figure 3.14: **Color brightness modulation and layer composition**: (a) the original gray-scale image, (b) an example of the final color image, (c) the case where a bright color is applied on a dark region, (d) blending factor: white color denotes $\alpha = 0$ and black $\alpha = 1$.

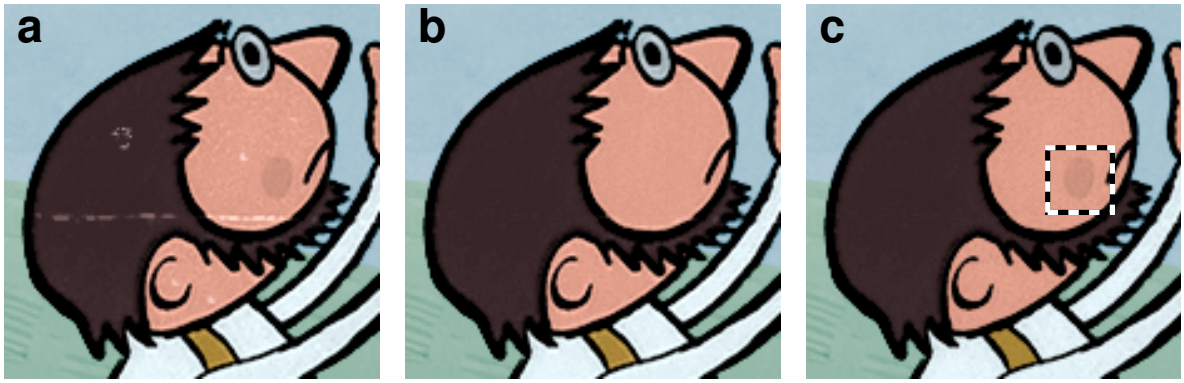

Figure 3.15: **Unsupervised dust spot removal**: (a) the original color image, (b) dust spot removal, (c) white dust spot removal.

Gaussian distribution (see Figure 3.15b) with the following probability density function:

$$\mathrm{pdf}(x) = \exp\left(-(x - \hat{I})^2/\sigma_I^2\right). \tag{3.16}$$

Such a simple dust spot removal technique is not suitable for regions where some intended inhomogeneities exist (e.g. cheek in Figure 3.15a). Usually the digital conversion is performed from the original celluloid negative where dust spots are black (white after inversion), therefor whenever the local inhomogeneities are darker it is possible to perform only white dust spot removal (see Figure 3.15c).

3.3 Results

In this section several colorization results and experiments are presented. They confirm superiority of the proposed framework both in the reduction of manual intervention and in the improvement of the visual quality. Images used in these experiments (as in Section 2.2.10) are scanned in the resolution of `720x576` from the original celluloid negative of classical cartoon *O loupežníku Rumcajsovi*.

3.3.1 Overall results

The proposed colorization framework has been implemented as a stand-alone application in which the colorization pipeline consists of three independent stages. In the first stage off-line precalculations are performed to prepare raw input data for interactive processing. This phase covers outline detection, image segmentation and structural similarity computations. It takes in average about ten seconds per frame on commodity hardware. In the second phase the camera motion estimation is performed and visible parts of background are extracted to form one large image which is colorized manually using standard image manipulation software. Then starts the interactive phase where the user colorizes the foreground layer frame-by-frame. Mouse or tablet is used to select predefined color labels from a predefined palette and correct color markers when the proposed prediction scheme fails. The colorization pipeline is finished by the color image synthesis phase. Here already colorized foreground and background layers are automatically putted together to produce the final color image.

Selected colorization results of several still images are presented in Figure 3.16, 3.17 and animations in Figure 3.18. At average two trained users were able to colorize one episode (roughly 30 animation sequences consisting of 10 000 frames) during one week (80 man-hours) in contrast to previous approaches which takes more than two months to process the same amount of frames. Using the proposed framework most of the time was spent on background colorization which is manual therefore requires large amount of hand-driven interventions. In the case of foreground colorization the user interaction takes in average 5 seconds per frame. However, the real processing speed strongly depends on the complexity of currently processed sequence.

3.3.2 Prediction performance

To verify the performance of the proposed color prediction scheme (Section 3.2.2) two experiments have been performed. In each experiment four representative pairs of nearly consecutive frames were selected. In the first group (Figure 3.19) it is possible to observe how the proposed approach behaves in a typical situation. In the second group (Figure 3.20) difficult cases are presented where the proposed approach tend to fail, however, still produced helpful results.

The proposed color prediction scheme is compared to a simple luminance-based technique where the color transfer is done between regions that have similar luminance (as in luminance keying). The amount of needed user intervention is expressed by the number of prediction errors. Each error requires to move mouse or table cursor over the region and place a marker

Figure 3.16: **Results – still image colorization (1)**: original gray-scale image (left) and the final color image (right). See how the color brightness varies as compared to the original luminance and how the image looks when a slightly different background layer is used (bottom).

Figure 3.17: **Results – still image colorization (2)**: original gray-scale image (left) and the final color image (right). See how the color brightness varies as compared to the original luminance and how the image looks like a slightly different background layer is used (bottom).

Figure 3.18: **Results – colorization of animation sequences**: selected frames with dominant structural changes. Slightly different color version of the background layer has been used in the first sequence.

with correct color label. This action takes usually several seconds per error depending on the region size and relative position (according to *Fitts'* law [53]).

Numbers of errors in Figure 3.19 confirm that in contrast to the simple luminance-based approach the proposed technique significantly reduces the amount of hand-driven interventions. Results are also interesting in the cel painting scenario where the crucial gray-scale information is not available (Figure 3.20). Nevertheless, the method usually fails to predict correct color labelling for some small or partially occluded regions. The fundamental problem is also how to assign appropriate color labels when a new structural pattern arises. Due to these circumstances limited manual interaction is still required to produce errorless results.

3.3.3 Visual quality

In this section the proposed color transferring technique is compared to *Levin*'s framework in the sense of visual quality (see Figure 3.21). Besides the visual quality it is also interesting to compare the amount of user intervention needed to produce the final colorization.

The most significant limitation of *Levin*'s framework is that only U and V components of YUV color space are modified. As was discussed in Section 3.2.3 artists prefer to prepare colors in the full color space by the reason that color components provide only a limited variance of appearance. Consequently in Figure 3.21 soldier's epaulets have slightly darker color in contrast to the real luminance in the original image. Similar situation is visible on the soldier's hand where the same color is preserved for the uniform although the hand is slightly darker in the original image. Also low contrast of outlines is observable due to $YUV \Rightarrow RGB$ conversion which does not preserve contrast as compared to proposed multiplicative color brightness modulation. The another artifact is color bleeding that arises since the least-square fashion of the *Levin*'s optimization problem tends to redistribute error smoothly along discontinuities.

3.4 Summary

This section introduced a novel approach to colorization of black-and-white cartoon animations. The proposed cartoon analysis framework has been extended by the technique called patch pasting that allows to transfer color information locally while still maintaining the global context of region correspondences. Another important steps toward were automatic color brightness modulation, layer composition and dust spot removal techniques allowing to produce final color images in broadcast quality without additional user intervention.

Numerous practical experiments confirm that using the proposed framework considerably less manual effort is required as compared to previous approaches. The colorization pipeline reduces from laborious frame-by-frame scribbling to a simple one-click correction workflow that makes the overall process tractable and cost effective. Moreover, the introduced framework is not limited only to the colorization of black-and-white cartoon animations. It can be directly applied to semi-automatic cel painting (see Figure 3.20) or to the restoration and enhancement of aged color cartoons (see Figure 3.22). In this case the additional color information simplifies the retrieval of region correspondences.

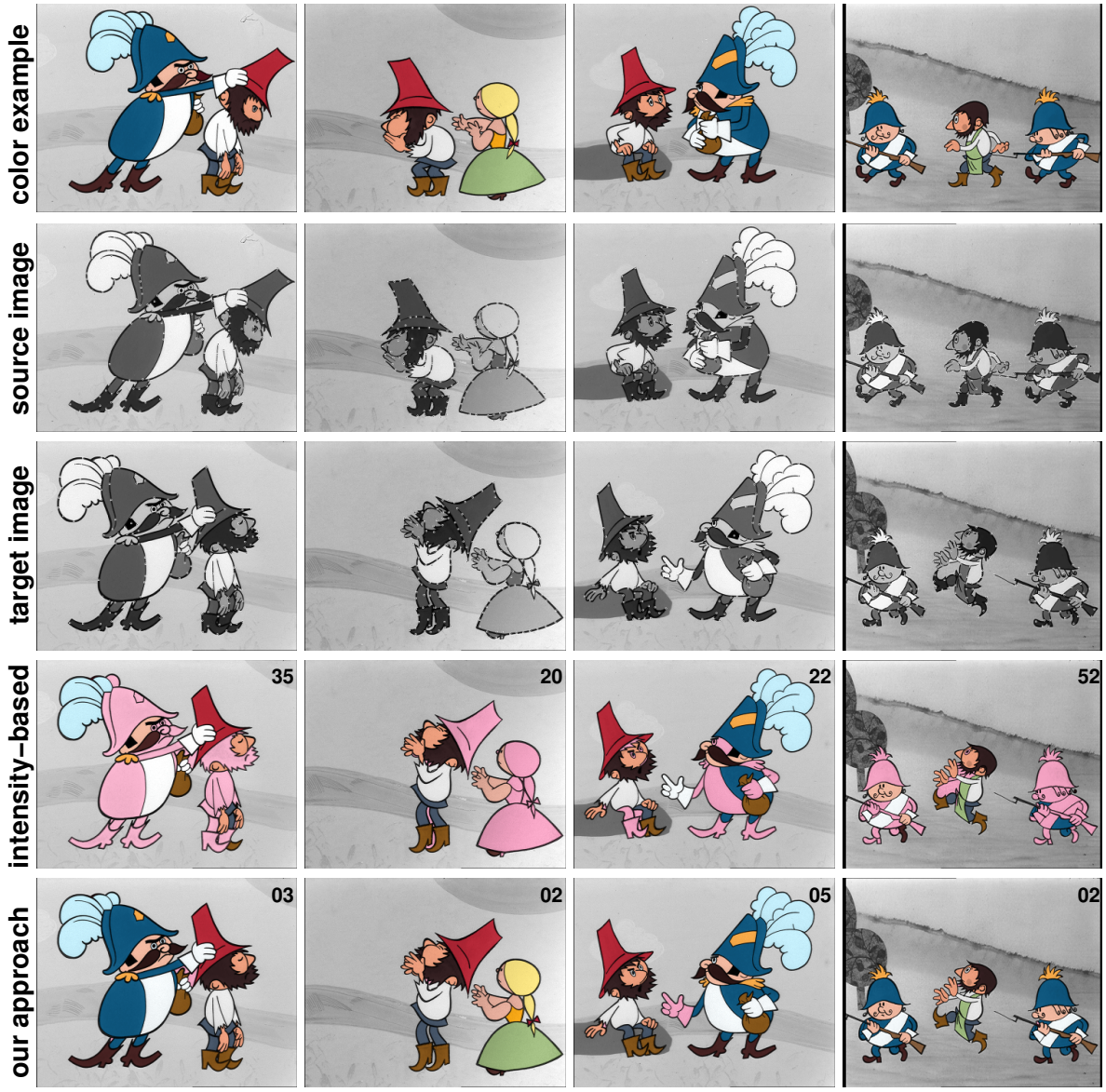

Figure 3.19: **Experiments – color prediction performance (common case)**: luminance-based prediction versus proposed color prediction scheme. Pink regions denote prediction errors and numbers in images represent the overall number of prediction errors. White dots in the source and target image denote extracted features.

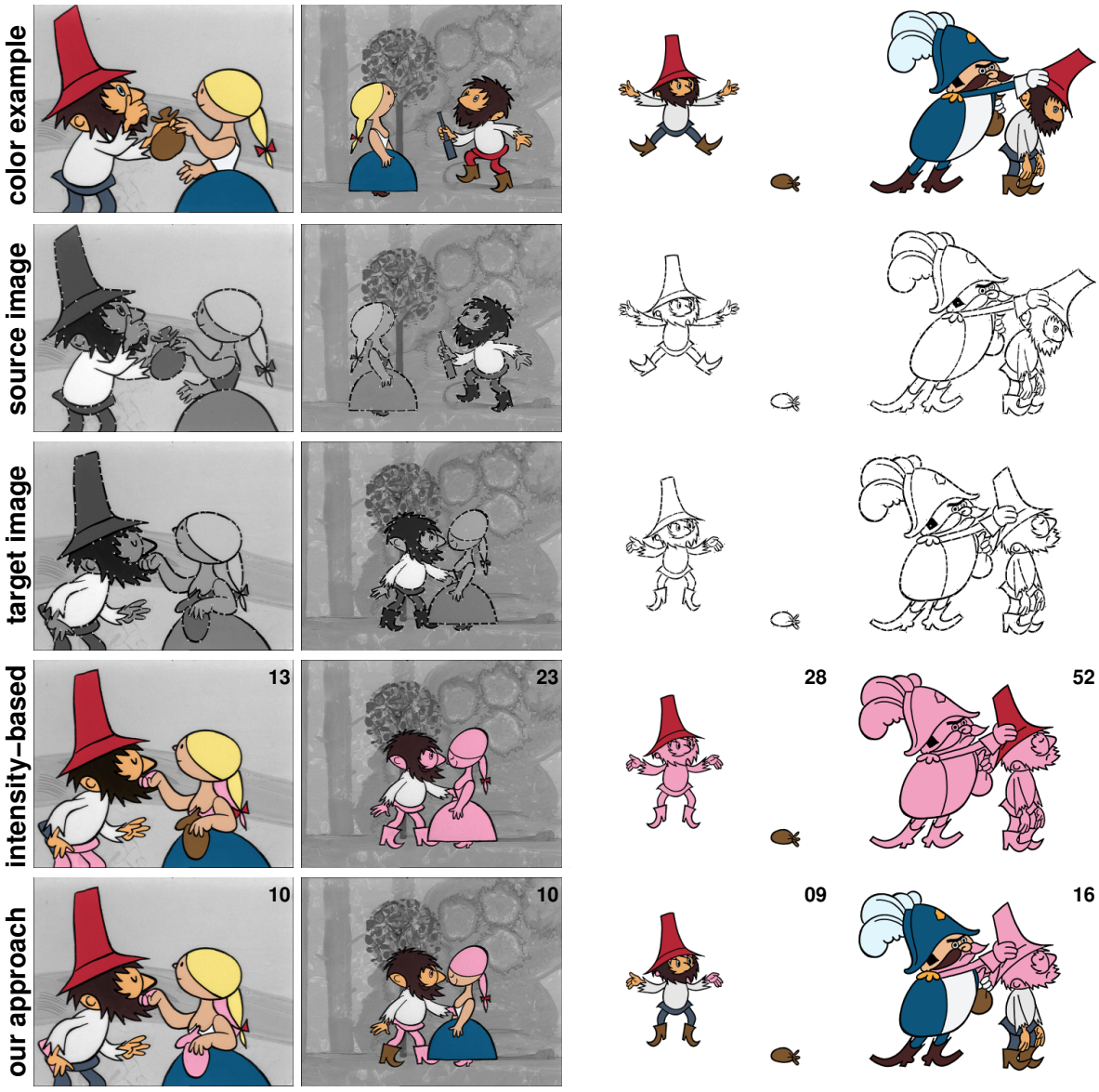

Figure 3.20: **Experiments – color prediction performance (difficult case)**: the same ordering as in Figure 3.19. Two difficult cases where many regions arise or change position/shape (left side) and color-to-region assignment for two cel paintings where no gray-scale information is available (right side).

Figure 3.21: **Experiments – visual quality comparison**: colorization produced by *Levin*'s framework (left) in contrast to the proposed approach (right): color markers (top), foreground layer colorization (middle), detail views (bottom).

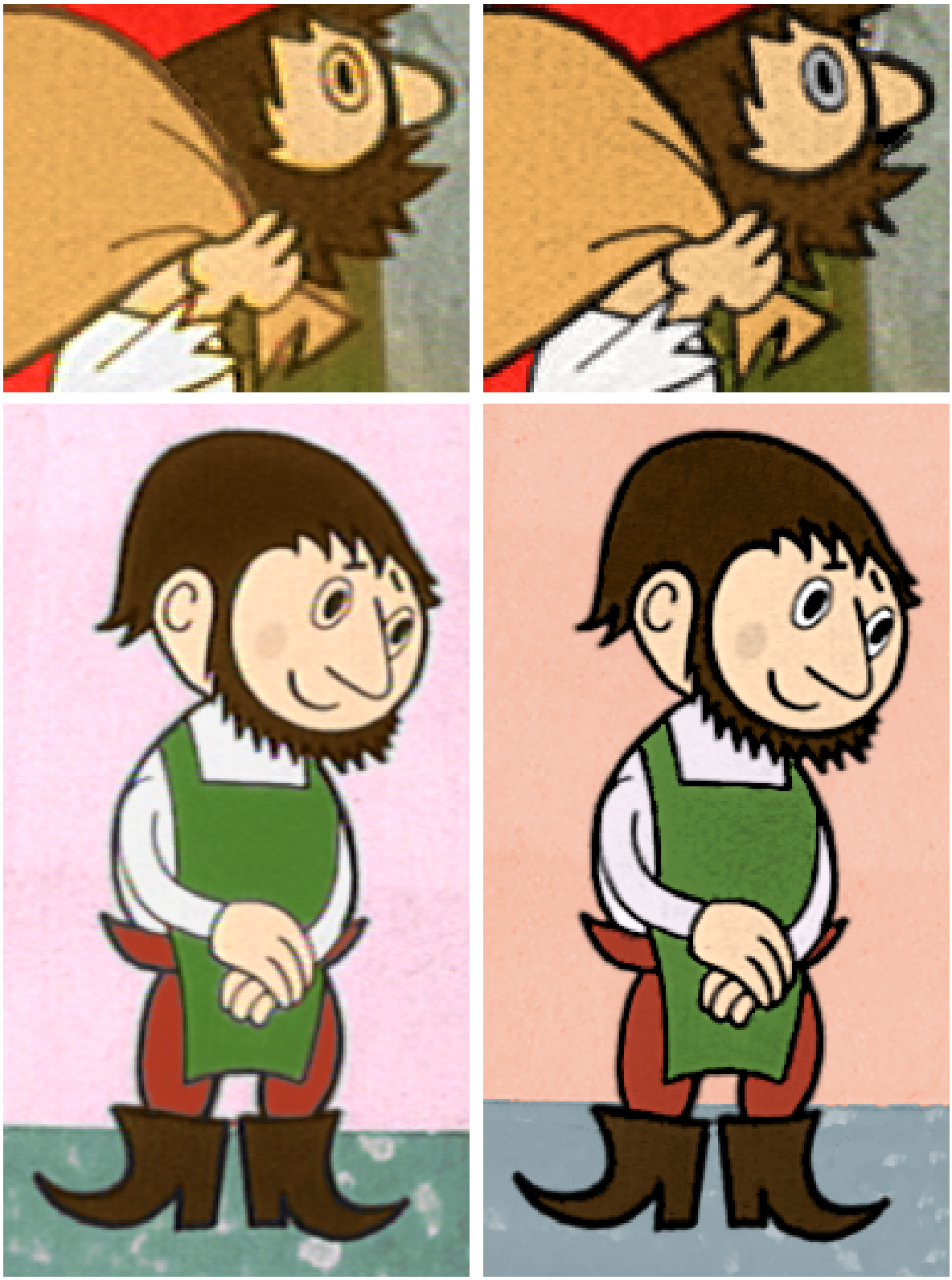

Figure 3.22: **Experiments – color cartoon enhancement**: the proposed colorization framework is not limited only to black-and-white cartoons, it can be exploited to enhance aged color cartoon animations. Detail views of degraded images (left) and the same images after enhancement (right).

4 Application II: Cartoon-by-example

Traditional cartoon drawings and animations contain visual style which is unique and typical for their authors [101]. Generations of children and adults enjoy these classical works and wish to watch new stories in the same style. However, when a classical cartoonist is deceased it is usually very tedious and time consuming to mimic his or her style in order to create new poses and characters undistinguishable from the original.

In this application the proposed cartoon analysis framework is used to reduce the amount of manual interventions connected with creation of new cartoon compositions and animations from the existing footage. First main limitations of previous image extraction and composition techniques are discussed and later in successive sections a novel scribble-based approach is described.

4.1 Previous work

The aim is to create novel cartoon poses and characters by extracting and composing fragments of the original artwork. Using standard image manipulation tools this task is tedious and time consuming. The proposed approach allows the user to simply (1) extract an interesting part in the original image and then (2) adjust it in a new composition using only few selection and deformation scribbles.

4.1.1 Fragment extraction

Interactive image segmentation (or fragment extraction) is one of the central problems in computer vision. First important group of methods exploit a coarse selection lasso (also called wire, lane or snake) drawn by the user [86, 60, 117] to encompass the desired fragment. After the coarse selection some sort of refinement is performed to augment lasso's shape so that it fits the real object boundaries. Such technique is suitable especially for extracting salient objects from almost homogenous backgrounds which is not the case of cartoons where the selection lasso typically crosses outlines of the neighboring regions.

The second approach exploits free-from strokes called scribbles to select desired fragments directly. Research in this filed became popular since *Boykov* and *Jolly* [14] proposed their revolutionary graph-cut based technique. This approach allows to infer binary segmentation of an arbitrarily gray-scale image from user-defined foreground/background scribbles. *Boykov* and *Jolly* formulate this variant of image segmentation as a min-cut problem and show that the optimal solution can be found quickly using a slightly modified version of max-flow algorithm. Later, several authors improve this technique to process color images [104, 146]. They additionally present several post-processing steps to obtain smooth alpha-channel from hard segmentation. Recently, *Wang* and *Cohen* [177] proposed joint interactive segmentation and alpha estimation using fast version of belief propagation algorithm [50]. In Figure 4.1 it is possible to compare interactive segmentation based on *Boykov*'s and *Wang*'s approaches in contrast to the technique proposed in this section. Against common expectations (likewise in Figure 2.2) and despite of large amount of precise manual selection also these modern techniques fail to produce clean and accurate segmentation for cartoon images.

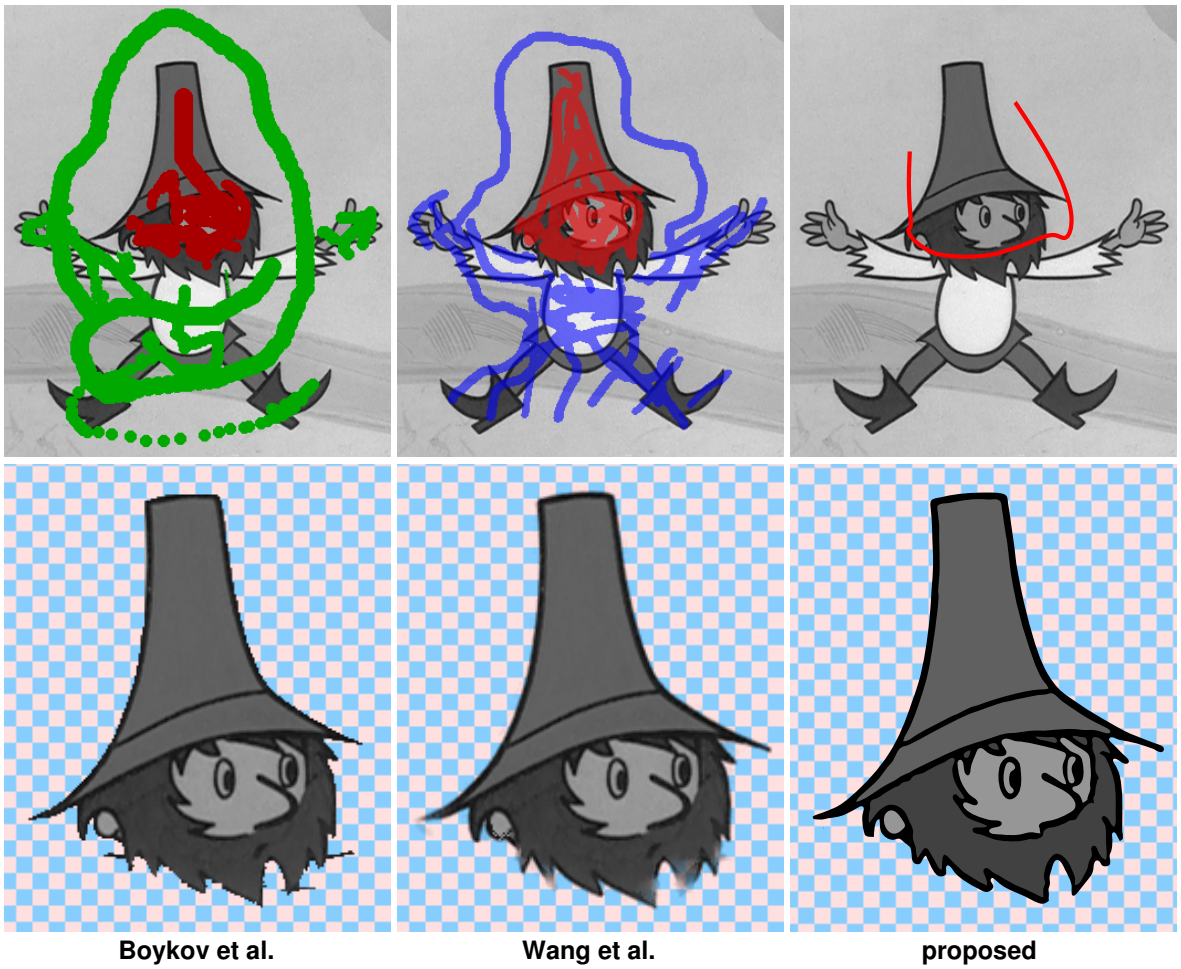

Figure 4.1: **Comparison of scribble-based fragment extraction techniques**: *Boykov*'s hard segmentation [14] (left), *Wang*'s joint segmentation and alpha estimation [177] (middle), and the proposed approach (right). See also the amount of user intervention (top) needed to produce the final segmentation (bottom).

The most conformable to the framework proposed here are works of *Barret* and *Cheney* [6] and *Saund et al.* [150] that share similar motivation, i.e. to simplify the selection and manipulation of objects in images. *Barret* and *Cheney* call this workflow *object-based image editing*. For the fragment extraction they exploit a set of pre-segmented watershed basins that can be selected one by one or by intersection with selection scribble. However, this is tractable only for homogenous parts e.g. interiors of regions in cartoons. For thin outlines this technique produces lots of small basins. The similar limitation occurs also in *Saund*'s system *ScanScribe* that works effectively only with simple blueprints on homogeneous background. There is no built-in mechanism able to simplify the extraction of region together with its corresponding outline.

4.1.2 Fragment composition

When a desired fragment is selected it is necessary to define its new pose in a target composition. This typically requires from the user to define proper translation, rotation, scale and also some amount of a free-form deformation. Using standard vector manipulation tools such task is typically nonintuitive and tedious since it is necessary to combine a number of basic transformations including translation, rotation, scale, bend, shear, etc.

The task of fragment composition can be also seen as a special form of image warping (for nice overview see e.g. [116]) where the common workflow is to first select a subset of salient features such as points [6], lines [8] or free-form curves [43] and interactively adjust their new positions in the target composition. Proper warping algorithm then produces the final deformation by considering these user-defined constraints.

The framework proposed here adopts the concept of popular intuitive technique called *warp* [43] that has been successful used in *Teddy* [76] and in other 3D interactive systems [87, 120]. The technique exploits two curves to define the free-form deformation. In the context of cartoon-by-example framework they can be called *composition scribbles*. The first scribble is drawn in the source image and the second in the target composition (see Figure 4.2). The basic assumption here is that scribbles are drawn in a constant speed so they have the same parametric length therefore point correspondences can be estimated by exploiting simple arc length parameterization.

This technique allows to define proper position and free-form deformation simultaneously. The another advantage is that typically the selection scribble can be also used for composition. However, the problem is that in previous works used *warp* only for scale preserving local deformation. In cartoon-by-example scenario the most prominent task is to define proper scale for fragments to match the scale of the final composition that is typically different. The original *warp* technique does not handle this requirement well since scale changes are propagated only in the direction parallel to the composition scribble (see Figure 4.3).

4.2 Cartoon-by-example framework

In this section a novel framework for interactive synthesis of cartoons from existing footage is described in detail. As in colorization and restoration application (Section 3) also in this framework most of components built upon the techniques presented in the proposed cartoon analysis framework (Section 2). They allows to reduce the amount of manual intervention connected with fragment extraction and manipulation in the target composition. First an overview of the cartoon-by-example pipeline is presented and later implementation details and results are discussed in detail.

4.2.1 Framework overview

The aim of the cartoon-by-example framework is to (1) alleviate shortcomings of previous scribble-based approaches, i.e. to simplify fragment extraction and allow scale invariant manipulation in the final composition, and (2) to increase visual quality of the final composition by exploiting the proposed raster-to-vector conversion scheme (Section 2.2.9) that allows to

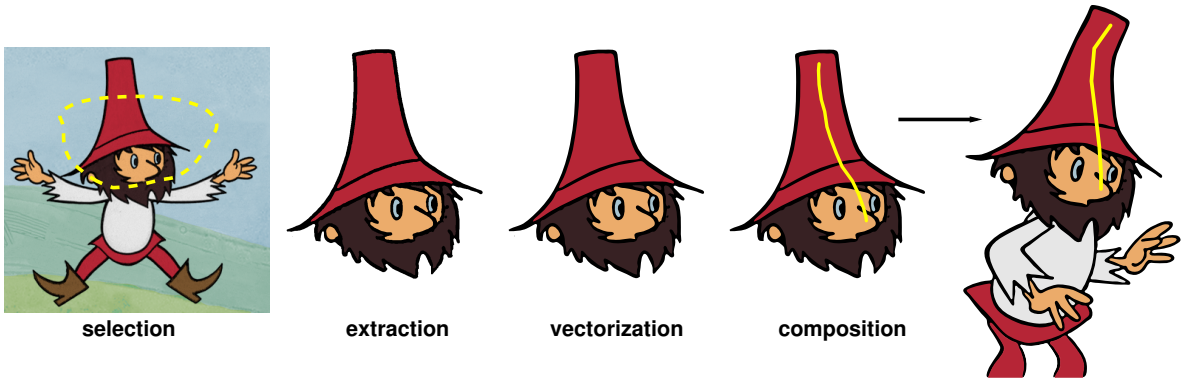

Figure 4.2: **Framework overview**: the user first selects a desired fragment in the original image (left), then the system automatically extracts it and perform vectorization (middle), finally the fragment is arranged in a new position using two composition scribbles (right).

avoid sampling artifacts and produce compelling compositions in the resolution independent manner.

In the first step (follow Figure 4.2) an unsupervised image segmentation (Section 2.2) is used to partition the input image into a set of regions. Each region is then classified (Section 2.2.5) whether it belongs to the background or to the foreground layer. An interactive phase follows. In this step the user simply selects a subset of regions forming the desired fragment. The selection process is remarkably simplified since the underlaying structure of the original image is known (see Figure 4.1). Afterwards, the system extracts the fragment together with the corresponding outlines (Section 2.2.7) and performs vectorization (Section 2.2.9). Finally the user arranges the fragment in a new position by drawing two composition scribbles that allow to simultaneously define a combination of uniform scale and free-form deformation.

4.2.2 Fragment selection

After the pre-processing (Section 2.2) the user is allowed to select an interesting part in the pre-segmented image. Since in this phase it is already known which pixel belongs to which region and vice versa, it is possible to use various selection tools, e.g. to simply click on the desired regions or to draw selection scribbles over them.

Preferred approach is to draw the selection scribbles in such a way that they can be later reused as composition scribbles. Whenever this approach does not allow to cover all important regions, additional scribbles are used to add them to the selection. To further simplify the selection, regions that have non-zero spatial intersection with an area of which the selection scribble is boundary curve are also selected (first and last points of the selection scribble are connected to form closed curve). Similar approach has been used also in *ScanScribe* [150], however, in the framework proposed here the selection scribble can be rather sloppy (see Figure 4.1) since background regions are automatically excluded from the selection process.

When the desired regions are selected the fragment extraction technique (Section 2.2.7) is used to extract corresponding outlines and form the final selection (see Figure 2.19). Afterwards

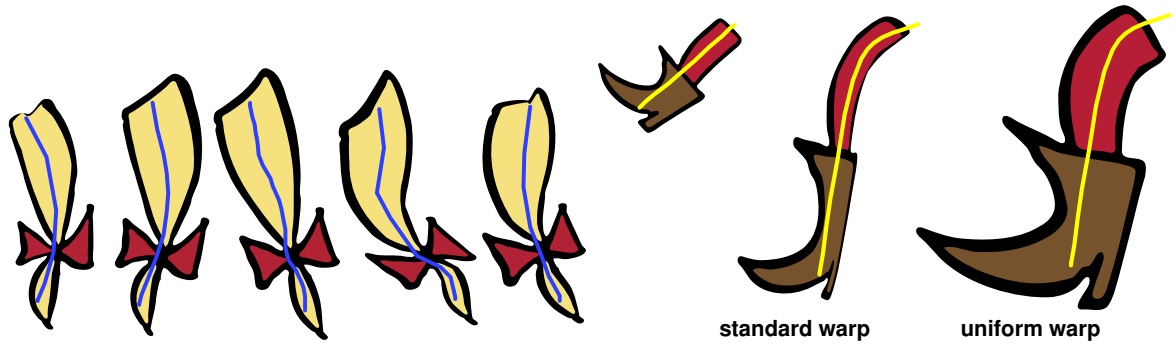

Figure 4.3: **Standard vs. uniform warp**: free-form deformation using standard *warp* (left), when the scale of target fragment position is different, the user intuition is to change it uniformly (right).

Figure 4.4: **Visual feedback for composition scribbling:** the user can see the composition scribble as well as the corresponding fragment deformation during scribbling. This is useful especially when it is necessary to get the correct proportions of fragments in the final composition.

super-resolved raster-to-vector conversion scheme is used (Section 2.2.9) to obtain resolution independent representation of regions and outlines.

4.2.3 Fragment composition

As was discussed in Section 4.1.2 the aim is to adopt the idea of *warp* [43], where the source fragment is deformed according to changes in positions and tangent orientations of the corresponding points on the composition scribbles (see Figure 4.3, left).

However, the problem is that the original *warp* produces non-uniform scale distortion that can be sometimes useful but in cartoon-by-example workflow usually does not fit the user's intuition since the source fragments have typically different scale and it is necessary to adjust them to meet the scale of the final composition (see Figure 4.3, right).

To address this issue a novel composition tool called *uniform warp* is proposed. It utilizes length ratios of source and target scribbles to obtain proper scale normalization. The source fragment is first scaled together with its composition scribble to meet the actual length of the target scribble and then the standard *warp* is applied to perform free-form deformation. Moreover, since the uniform warping can be implemented in real-time, it is possible to visualize

Figure 4.5: **Scribbling key-frame animation**: the user scribbles start (left) and stop (right) position of the selected fragment and the system interpolates these two positions to obtain smooth inbetweening.

the composition process by displaying intermediate warps of the fragment interactively to bring invaluable feedback to the user (see Figure 4.4).

Although users are not limited to use only *uniform warp* (the system offers common manipulators such as translation, rotation, scale, flipping, mirroring and also standard *warp*) experiments prove that they often prefer *uniform warp* since it allows to define a proper scale and deformation at one snap. Such simplification is useful especially when the composition serves as a key-frame for a novel animation. The user can quickly scribble new key-frame positions and the system automatically interpolates them (see Figure 4.5). This intuitive workflow is attractive especially for young children who wish to create own cartoon animations but are not familiar with professional tools.

4.2.4 Implementation issues

The original *warp* operates with smooth curves. In the framework proposed here mouse or tablet is used to acquire composition scribbles therefore it is necessary to implement *warp* in a discrete domain. To accomplish this a set of drawn pixels is approximated via piecewise linear curve (line strip). The length of each scribble is computed as a sum of individual lengths of line segments. Then uniform arc length parameterization allows to estimate new positions of source control points on the target strip (see Figure 4.6). When the shape of the source strip does not fit the target shape well it is necessary to perform subdivision in a problematic area and recompute new locations of the corresponding points.

The another implementation issue is a depth ordering of fragments. When the fragment is selected the user has to assign proper depth value to obtain a correct visibility of fragments in the final composition. This task is simple for planar layers. However, when the fragment is not strictly planar, i.e. when a part of it should be in front and the rest behind some different fragment (e.g. hand and body, see Figure 4.7a,b), it is necessary to assign two different values and then perform smooth spatial transition between them (see Figure 4.7d). To be able to render such compositions correctly *z-buffer* technique should be used instead of simple depth ordering. Another possibility is to cut the fragment via special cutting scribble and then assign two different depth values for each individual part (see Figure 4.7c). This solution is much more practical when formats like `PDF` or `SVG` are used for the final output since they do not support per-pixel depth tests. Cutting scribbles are also useful when the user wants to extract only a part of the homogenous region regardless of existing outlines.

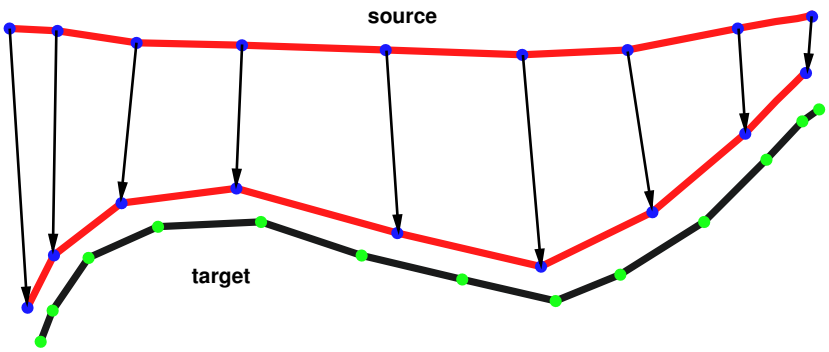

Figure 4.6: **Linear mapping between composition scribbles**: the source strip is subdivided to match the shape of the target strip and then a simple arc length parameterization is used to map source control points on the target strip.

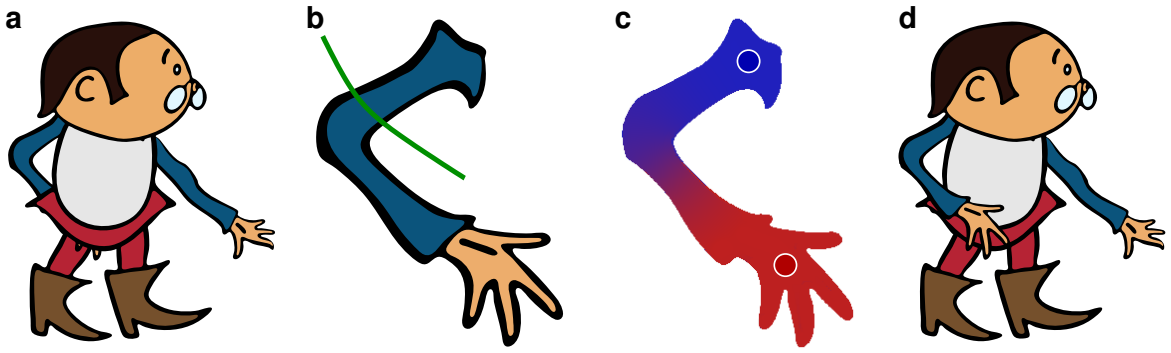

Figure 4.7: **A solution to the depth ordering problem**: one part of the hand should be behind and the second part in front of the body (a). Fragment cut using special cutting scribble (b) or smooth depth assignment (c) allow to produce the correct composition (d).

4.3 Results

This section presents several cartoon-by-example experiments. Similarly to previous experimental sections source cartoon images are scanned in the resolution of `720x576` from the original celluloid of cartoon *O loupežníku Rumcajsovi* and additionally colorized using framework proposed in Section 3.

In the first experiment non-experienced user scribbled five new characters in the style of *Radek Pilař* from a set of twelve different images (see Figure 4.8). Using the proposed framework it takes only tens of seconds to complete them. For the same task experienced user needs minutes to obtain comparable results with a standard image manipulation tool.

In the second experiment a simple cartoon animation has been created by scribbling six new key-frames (see Figure 4.9). Simple linear interpolation of scribbles has been used to produce smooth inbetweening. Although such a simple approach does not preserve rigidity of fragments the final animation looks still compelling. Nevertheless, more advanced rigidity preserving techniques [4, 77] can produce similar results with lower number of key-frames.

4.4 Summary

In this section a novel framework for example-based synthesis of cartoons has been presented. Practical experiments performed on real cartoon images confirm that it allows one to synthesize new cartoon drawings within much shorter time frames as compared to standard image manipulation tools. Namely it has been shown how the proposed cartoon analysis framework simplifies the fragment extraction phase and increases the visual quality of the final composition. Also a new intuitive scribble-based composition tool – uniform warp has been presented. It successfully extends toolbox of existing vector manipulation tools since it allows to define relative change of scale and free-form deformation of the selected fragment at one snap.

Figure 4.8: **Results – five new cartoon characters created by example**: desired fragments have been selected with several selection scribbles marked as dashed curves (top). Pairs of composition scribbles marked as solid curves have been used to sketch new compositions (bottom). Note that some composition scribbles serve also as selection scribbles.

Figure 4.9: **Results – simple cartoon animation created by example**: six new key-frames have been scribbled by the user (top). Proposed system interpolates them to produce smooth in-betweens (bottom).

5 Application III: Video compression

In this section a novel approach to video compression suitable for outline-based cartoon animations is presented. Similarly to previous two applications also here the assumption is that foreground and background layers have considerably different appearance. The aim is to fully exploit this prior knowledge and provide better visual quality for the same storage requirements as compared to standard video compression techniques. Moreover, the aim is not only to compress the animation efficiently but also to replay it in real-time and reach partial spatial scalability by exploiting commonly available graphics hardware.

5.1 Previous work

For decades video compression was and still is very popular research topic [139, 39]. However, despite of everlasting demand on cartoon oriented video codec only a few methods directly focus on cartoon animations.

Kwatra and *Rosignac* [95] presented approach where pre-segmented `2D` shapes are swept through the time axis to create `3D` volume. Edge-breaker compression [144] is then used to encode volume geometry. Decoding is performed by intersecting compressed volume with the image plane using `GPU`-based capping technique [145]. This approach is suitable only for regions that change their shape and/or position slightly in time. Experiments were performed only on pre-segmented real-world image sequences which are far from classical cartoon animations since the motion is much coarse and region shapes suffer from occlusion. Authors also did not address the key problem of region shape extraction for more complicated image sequences.

Concolato et al. [41] assume that the input sequence is already stored in some common vector format such as `SVG`. For this type of data they suggest to exploit existing *Binary Format for Scenes* (`BIFS`) from `MPEG-4` standard [46] that can be think of as `VRML` [180] with compression and streaming capabilities. This standard has been published in 1998 by `ISO/IEC` but it still waits for full implementation. *Concolato et al.* shown that in the context of cartoon animation proper implementation of `BIFS` allows for significant savings in contrast to standard vector formats. However, the practical usability of `BIFS` is still doubtful since it cannot be directly applied to existing cartoon animations stored as a sequence of bitmap images.

Lee and *Kassim* [98] were probably the first who tried to develop practically usable video codec for cartoon animations. Their main idea was (1) to use standard wavelet coding for the whole image but with lower encoding bit-rate, (2) use *mean-shift* segmentation [40] to filter the image so that it consists only of homogenous regions with step boundaries. Such image is then coded using a new set of basis functions called *wedgelets* and *beamlets*. These are designed to capture main characteristic of cartoon image: long linear discontinuities (wedgelets) and thin outlines (beamlets). During the playback low bit-rate wavelet layer is overlapped with wedgelet/beamlet layer to emphasize discontinuities. Authors compare this approach with `JPEG-2000` [78] and report superior visual quality for comparable bit-rates. However, the main drawback of this technique is the use of mean-shift segmentation that tends to produce inaccurate results (as was discussed in Section 2.2) and so the image quality can be notably reduced before the proposed encoding scheme is applied.

5.2 Video compression framework

In this section a novel approach to video compression of classical cartoon animations is proposed. Its aim is to overcome shortcomings of previous approaches by exploiting cartoon analysis framework proposed in this thesis. First a brief overview of the whole pipeline is presented and later, in successive sections, implementation details are discussed.

5.2.1 Framework overview

The proposed video compression pipeline seamlessly plugs to the output of the cartoon analysis framework. The assumption is that the foreground and background layers are already detached so that the input is: (1) frame-by-frame vectorized foreground layers, and (2) several reconstructed background layers together with corresponding camera pan and zoom information. Similarly to [98] the key idea is to use different coding scheme for each layer.

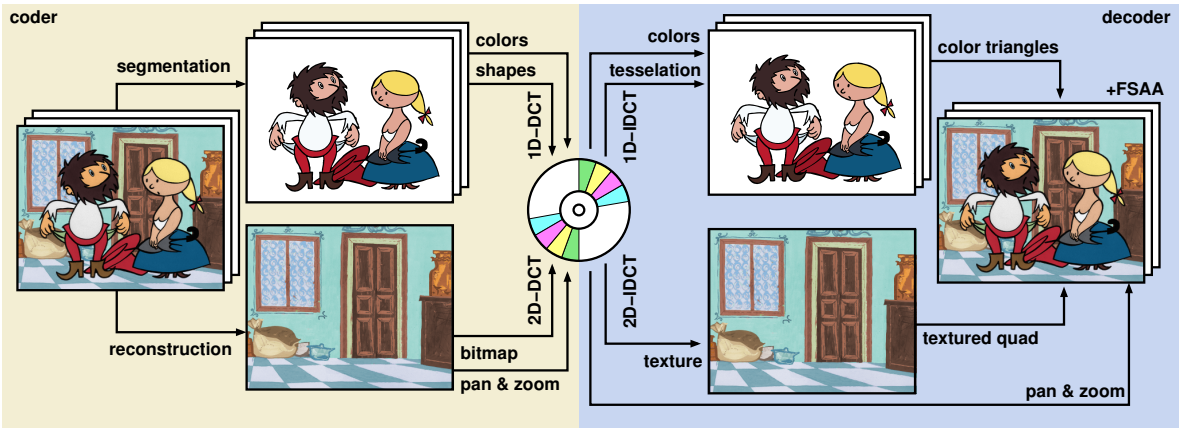

Figure 5.1: **Video compression pipeline**: first foreground and background layers are detached using the proposed cartoon analysis framework (left). Foreground layer is vectorized and compressed using `1D DCT` (top left). Background layers are reconstructed and stored as static images using standard `JPEG` compression (bottom left). Camera pan and zoom parameters are stored per frame. During the decompression hardware accelerated full screen anti-aliasing is activated and the background images are decoded and uploaded to the texture memory (bottom right). Foreground shapes are decoded per frame using `1D IDCT` and tesselated to triangles (top right). Finally the background layer is rendered as textured rectangle using proper camera pan/zoom and foreground triangles are superimposed to produce the final composite (right).

Since the reconstructed background layer is static textural image it can be simply stored as a standard `JPEG` image [175]. Optionally, when the camera moves during the animation, per frame information about pan and zoom are included (see Figure 5.1, left bottom).

On the other side the dynamic foreground layer is almost homogenous and contains sparse discontinuities (edges). For this type of `2D` signal standard image compression techniques like `2D DCT` or `DWT` are not feasible. The important information here is not stored in the area of region but along its shape, therefore `1D` shape coding techniques can produce better results.

To perform the transition from 2D to 1D foreground region shapes are first adaptively sampled to produce piecewise linear representation that can be easily transformed to the frequency domain using complex 1D DCT. Finally, a simple binary quantization is used to retain proper number of significant DCT coefficients. To further lower storage requirements lossless *Burrows-Wheeler* block-sorting transform [19] and *Huffman* coder [148] are used for the final compression (see Figure 5.1, left top). As a supplement to the shape information the layer depth (see Figure 2.21) and mean color are stored as well.

During the decompression/playback phase (see Figure 5.1 right) the aim is to render high-quality composition of layers frame-by-frame in real-time. For this time critical task GPU-based approach is proposed. To ensure high-quality output hardware supported full screen anti-aliasing is switched on. Then the background image is decoded and uploaded into the texture memory. During the animation the background texture rectangle is properly shifted and zoomed and region shapes are decoded via complex 1D IDCT, tesselated and superimposed over the background layer.

5.2.2 Shape compression

After the segmentation inferred region shapes need to be converted from raster to vector representation. The important question is which vector representation is more suitable for compression. A straightforward approach is to use directly control points of *Bézier* cubics generated by contour tracing algorithm as was described in Section 2.2.9. Although they provide an optimal approximation in the rate-distortion sense [192] and a resolution independent GPU-based rendering is possible [108], the problem is that 2D coordinates are not feasible for further compression since they represent shape energy uniformly. From this point of view 1D DCT-based approach is much more convenient since it redistributes the energy so that it accumulates to a small number of coefficients. This property can be utilized for efficient compression [162]. However, in the frequency domain as opposed to *Bézier* cubic representation, an additional information is needed to produce optimal bit allocation for DCT coefficients in the rate-distortion sense (*quantization table*) [141]. The problem is that such a table is in general different for each shape therefore in practice some globally optimal table should be estimated as for example in JPEG compression [175].

In the framework proposed here a novel DCT-based shape compression scheme is introduced. It exploits two important observations based on experiments with cartoon shapes (see Figure 5.3 and 5.2):

1. When the original shape is sampled adaptively, then the resulting DCT coefficients after quantization better approximate the shape as compared to uniform sampling with the same quantization table.

2. When the original shape is sampled adaptively, it is possible to estimate the optimal number of DCT coefficients using the number of control points needed to produce optimal *Bézier* cubic approximation.

Resulting shape compression scheme works as follows: first a standard contour tracing algorithm [181] is used to fit a set of *Bézier* cubics to approximate the raster representation of the original shape. Then the overall number of resulting control points N_p is remembered and

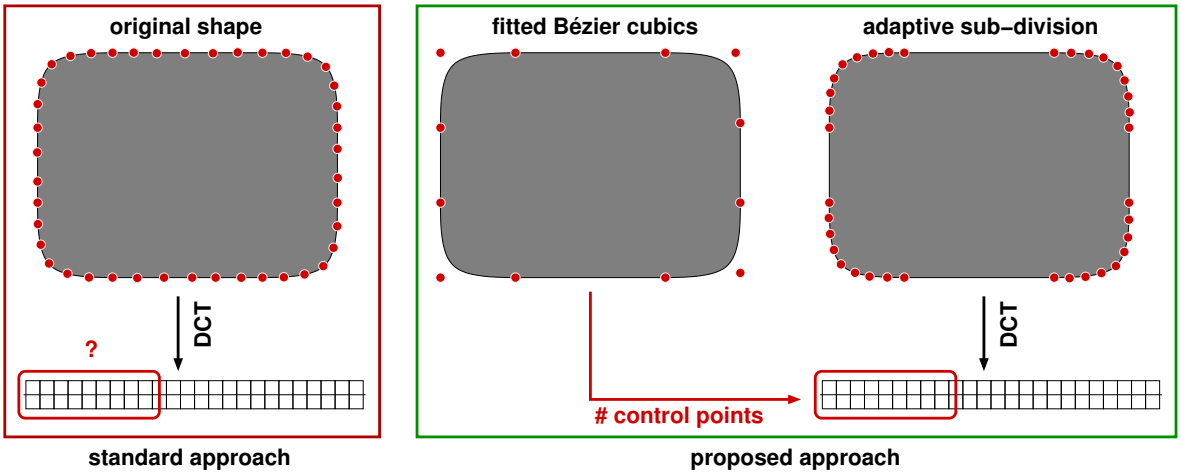

Figure 5.2: **A novel shape compression scheme**: using standard approach the shape is sampled uniformly and then transformed to the frequency domain. An issue is how to quantize DCT coefficients to reach the optimal result in the rate-distortion sense. In the proposed approach the shape is sampled adaptively and a simple binary quantization table is used. The number of important DCT coefficients is estimated from the number of control points obtained after the optimal *Bézier* cubic fitting.

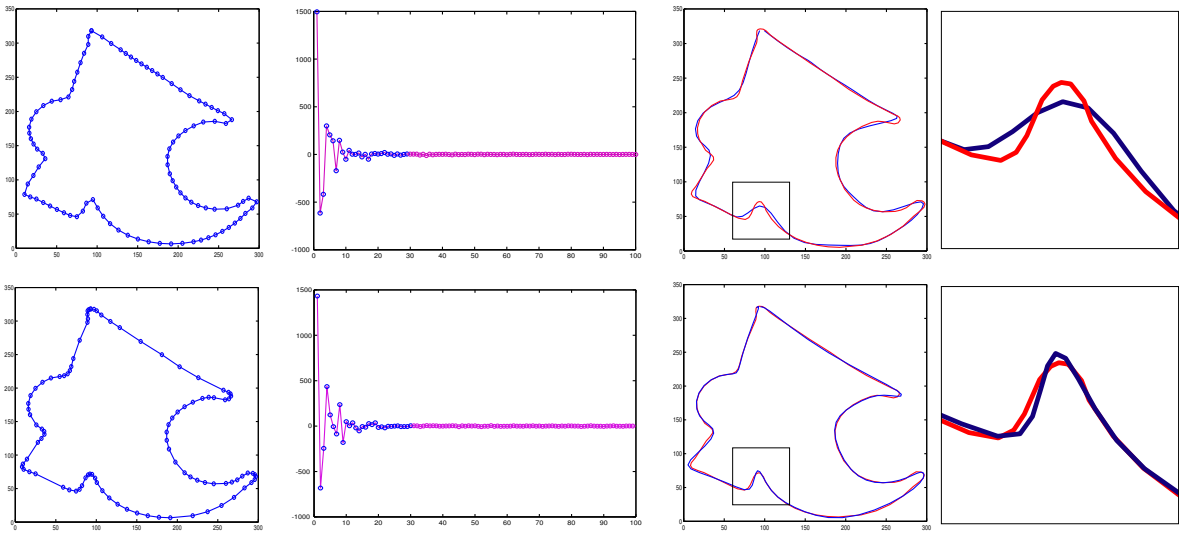

Figure 5.3: **Compare the efficiency of the 1D DCT-based vector compression for uniformly** (top) **and adaptively** (bottom) **sampled shape**: for the same number of DCT coefficients (blue dots) adaptive sampling produces much better approximation.

curvature sensitive subdivision is performed [37] to obtain adaptive sampling of the original shape. Resulting 2D coordinates of points are converted to the sequence of complex numbers (x as real and y as imaginary part). These are transformed to the frequency domain. Here first N_p DCT coefficients are normalized and converted to 16-bit integers. Finally, *Burrows-Wheeler* block-sorting transform [19] and *Huffman* coding [148] are used to further reduce information redundancy.

5.2.3 Temporal coherency

As was discussed in Section 2.3.1 in classical cartoon animations the motion is usually coarse and repetitive. To save tedious manual work artists reuse animation phases as much as possible. This typical redundancy allows much better compression ratios as compared to real-world videos. To exploit it a pool of already stored frames (*frame vocabulary*) is maintained. Whenever a new frame is going to be encoded, frame vocabulary is searched for possible duplicities and when the search is successful only a frame reference is stored.

Another possibility is to reuse similar region shapes. However, in classical cartoon animations each animation phase contains unique region shapes even if only a small part of body moves. This is because artists draws all poses manually without the possibility to copy-and-paste unchanged parts. In the final animation this limitation produces unique spatiotemporal noise that brings the feeling that the cartoon was created manually. It is clear that this artistic noise cannot be exactly modelled using rigid transformations and so it is not possible to simply correct visual inconsistency arising when a single region shape is reused in different frames. Although free-form deformations can solve this problem well, the amount of additional information usually does not produce meaningful savings.

5.2.4 Color assignment

The another issue connected with compression is the estimation of a visually dominant mean color for each foreground region. A naive approach is to simply compute mean over all pixels and then store it to accompany shape information. The problem is that mean color tends to flicker due to global brightness fluctuation in aged movies. Also small regions are problematic. Here the mean color is usually strongly biased due to outline anti-aliasing and results in disturbing spot flashing.

To avoid these artifacts a static palette is generated using mean-shift color clustering [40] where clusters are constrained to maximize the brightness difference. For each region the mean color is computed and the index of the nearest color from the clustered palette is assigned. Since the palette is usually much smaller then the number of all possible colors (typically only 16 colors out of 16M) temporal flickering disappear.

5.2.5 Playback

As was mentioned in Section 5.2.1 real-time performance and compelling visual quality can be reached by utilizing GPU-based approach to render background and foreground layers in each animation frame. The problem is that playback phase is time critical since typically tens

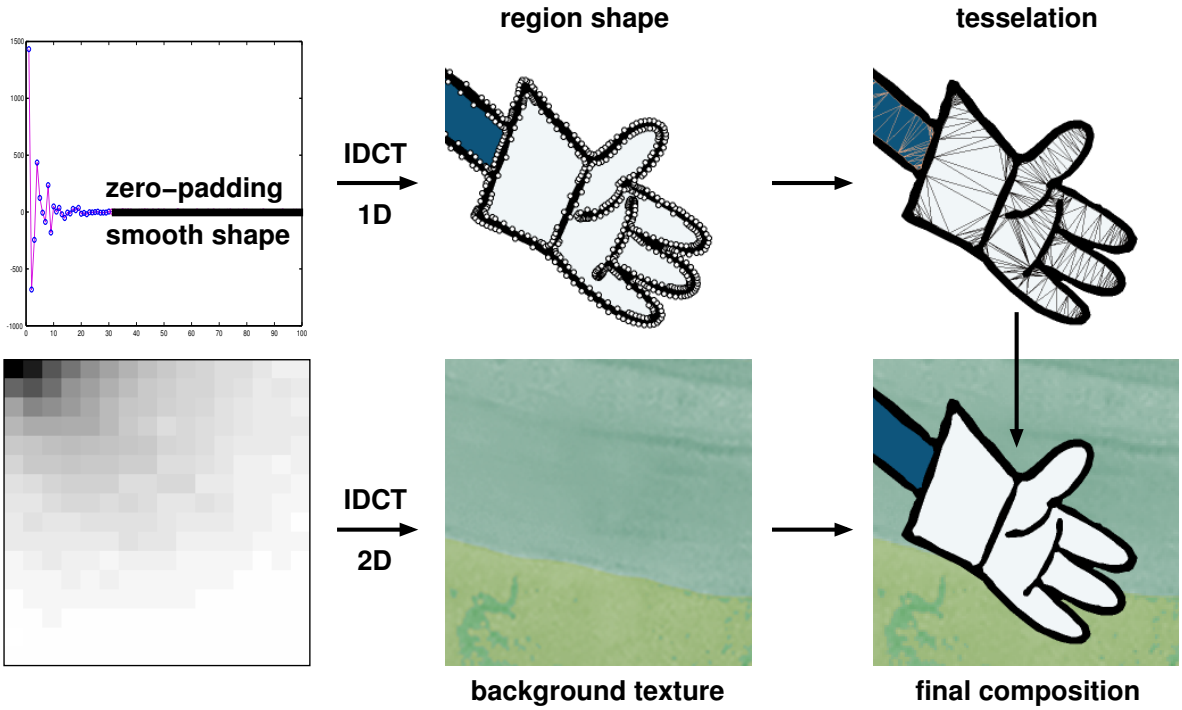

Figure 5.4: **An overview of the playback phase**: zero-padding together with complex `1D IDCT` is used to recover an adaptive piecewise liner representation of region shapes (top left), `2D IDCT` is used to decode the background image (bottom left) that is afterwards stored in the texture memory (bottom middle). Foreground regions are tesselated to triangles (top right) and rendered over the background texture to produce the final composition (bottom right).

of polygons with hundreds of vertices should be decoded, tesselated, and rendered 25 times per second. In this section relevant implementation issues are discussed in more detail.

Before the playback starts it is necessary to activate hardware accelerated full screen anti-aliasing to alleviate polygon boundary aliasing (see Figure 5.5). Optionally also all background layers can be uploaded to the texture memory in the beginning. However, since this step is nonrecurring and relatively fast, it is usually possible to upload textures on demand when the animation sequences change.

During the playback textured rectangle with proper background image is first rendered to the frame-buffer according to stored camera pan and zoom. Then the `DCT`-based representation of shapes is decoded and transformed back to the spatial domain. To preserve smoothness, zero-padding is used in the frequency domain before the `FFT`-based `IDCT` [58] recalculates `2D` coordinates of points (see Figure 5.4). Thanks to prior adaptive sampling, this process yields good piecewise linear approximation of the region shape that can be directly used for polygon rendering. Extra processing time is required only for non-convex polygons that have to be tesselated into the triangle strips. For this task standard `gluTess*` framework [187] is sufficient. However, for polygons with larger number of vertices *Held*'s `FIST` library [66] performs slightly better. Additionally the topology-based depth ordering (Section 2.2.9) and

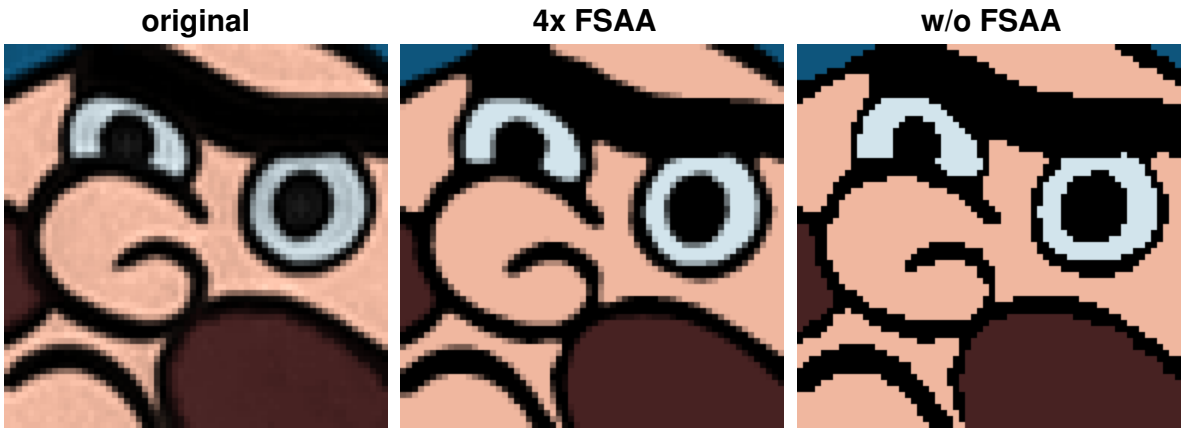

Figure 5.5: **Hardware accelerated full screen anti-aliasing for compelling visual quality**: the original image (left) in contrast to rendering with (middle) and without (right) full screen anti-aliasing.

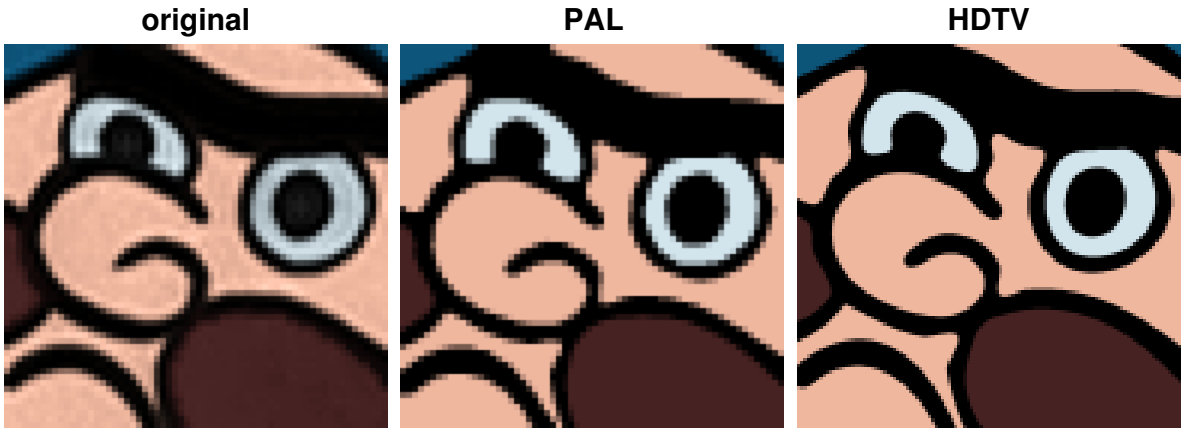

Figure 5.6: **Resolution independent video playback**: the proposed video codec allows to broadcast the original image sequence (left) in `PAL` (middle) or `HDTV` (right) resolution without resampling artifacts.

per pixel depth tests are used to draw regions in a correct order and to discard regions that actually represent holes to background layer.

5.3 Results

In this section several video compression results are presented. Similarly to previous applications experimental data are classical cartoon animations scanned in `PAL` resolution (`720x576`) from the original celluloid negative of Czech cartoon *O loupežníku Rumcajsovi* additionally colorized using framework proposed in this thesis.

Five different animation sequences have been encoded using the proposed video codec. Encoding parameters has been set to achieve an average bit-rate of `256 kbps` (suitable for broadcasting through `ADSL` internet connections). On `AMD Athlon A64 2800+` with `ATI Radeon 9700`

and 6x FSAA it takes on average 10 seconds to compress and 30 milliseconds to decompress and render one frame.

To see results in motion, a simple OpenGL-based player is available on a web page[1] together with short testing sequence and AVI encoded using DivX for comparison.

In Figure 5.7 and 5.8 details of several decompressed frames from different animation sequences are depicted. To render them the original PAL resolution has been used. However, the resolution itself can be arbitrary. This can be useful especially when the original cartoon animation stored in PAL have to be broadcasted in HDTV (see Figure 5.6). The only limiting factor here is a bilinear interpolation of the background texture which fortunately does not produce disturbing artifacts since the background layer usually does not contain high-frequency details.

In Figure 5.7 and 5.8 no blocking or ringing artifacts are visible in the output of the proposed video codec, only several high-frequency shape details are omitted. The another artifact that is not visible in static images is negligible temporal shape noise due to different loss of precision after binary quantization of DCT coefficients. In the animation sequence there are also several examples of small misclassified regions that are fortunately not so disturbing since the background layer is usually almost homogeneous at the same position.

5.4 Summary

A novel video compression pipeline for traditional outline-based cartoon animations has been presented. Practical experiments performed on real cartoon animations confirm that for comparable encoding bit-rates the proposed approach achieves superior visual quality as compared to standard video compression techniques. Moreover, the proposed video codec allows to playback compressed sequences in an arbitrary resolution thanks to partial spatial scalability.

The important result verified in this application is the fact that large savings can be reached due to significant redundancy of animation phases not available in real-world videos. Also two new shape coding heuristics have been established: (1) the use of adaptive sampling prior the application of DCT and (2) the use of the number of optimal control points of fitted *Bézier* cubic for the estimation of the number of dominant DCT coefficients.

The proposed approach has two major limitations: (1) slow compression phase and (2) inability to process cartoon animations that does not fit to the basic assumptions of the proposed cartoon analysis framework, i.e. static background layer and outlined homogenous regions in the foreground. Another issue that has not been explicitly solved is an automatic sequence annotation that is desirable for consistent background reconstruction.

[1] http://www.cgg.cvut.cz/~sykorad/codec

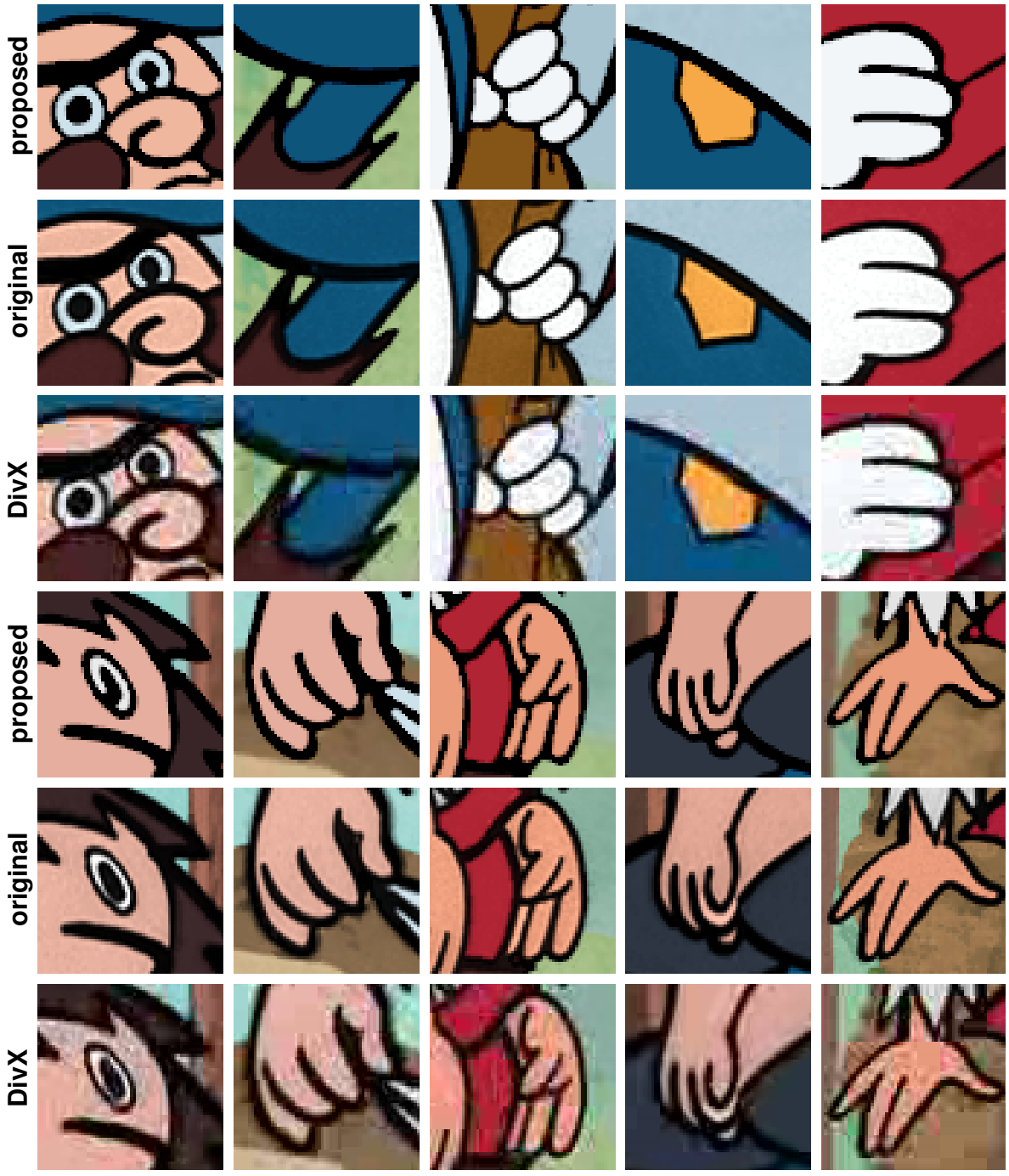

Figure 5.7: **Results – video compression (1)**: compare detail views in three rows – results of the proposed approach (top), the original images (middle) and results of `DivX` at comparable bit-rate (bottom). No blocking or ringing artifacts are visible in the output of the proposed approach only several high-frequency shape details are omitted.

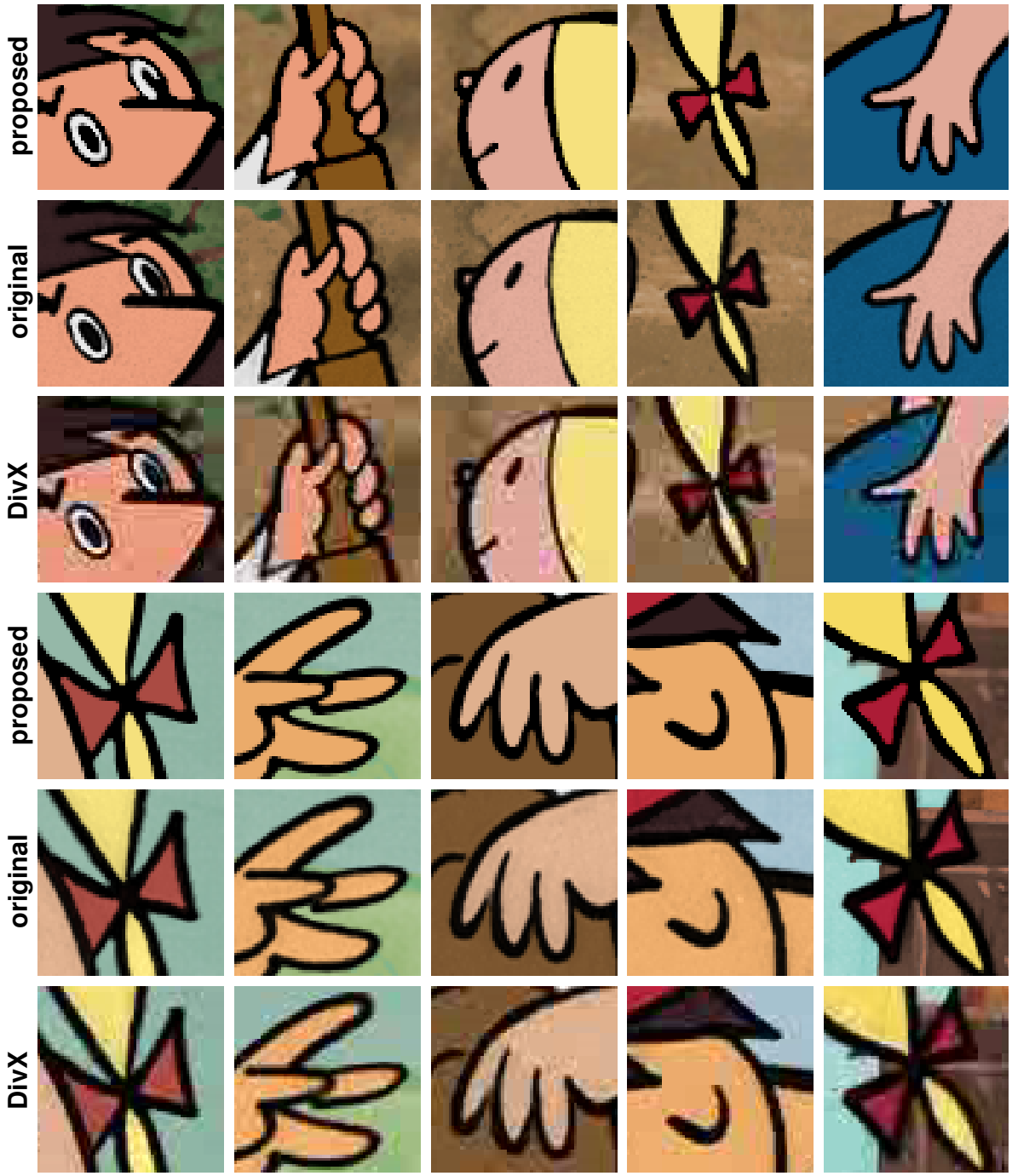

Figure 5.8: **Results – video compression (2)**: compare detail views in three rows – results of the proposed approach (top), the original images (middle) and results of `DivX` at comparable bit-rate (bottom). No blocking or ringing artifacts are visible in the output of the proposed approach only several high-frequency shape details are omitted.

6 Conclusions and future work

This thesis introduced a novel cartoon analysis framework suitable for cel- or paper-based animation techniques where each frame is created as a planar composition of static textural background and dynamic homogenous foreground.

The framework allows to pop-up `2.5D` structure of the scene by extracting outlines, locating foreground regions, vectorizing foreground layers, and reconstructing the visible portion of the background layer. On the basis of this decomposition structural similarity analysis is performed to estimate frame-to-frame and region-to-region correspondences in the foreground layer.

Additionally three practical applications have been implemented to verify the general usability the proposed framework:

1. **Colorization and restoration.**

   Bringing new color information into gray-scale images proven to be tedious and time consuming. Despite of recent advances in colorization research the process is still labour intensive and for long animations practically intractable. The framework proposed in this thesis allows to significantly reduce the amount manual intervention and make the colorization and restoration of classical cartoon animations practical and cost effective.

   Thanks to layer decomposition and region-to-region correspondence retrieval two significant speed-ups have been reached:

   a) The reconstructed background layer can be colorized at one snap and then reused in each animation frame.

   b) Once the first frame is colorized region correspondences and patch pasting allows to propagate color information to the following frames.

   Exploiting them colorization workflow reduces from tedious scribbling to few one-click corrections per frame. The same workflow is also applicable to semi-automatic cel painting where the number of corrections is higher but in the presence of extensive motion still lower as compared to position-based techniques such as simple onion fill.

   Thanks to automatic color modulation and layer composition color images can be reproduced in broadcast quality without additional user intervention. The key advantage is the ability to manipulate not only hue and saturation but also the color brightness. This is useful especially for an artist who has a complete control over the process of color transfer. Moreover, this feature together with the proposed automatic dust spot and image vignetting removal technique allows to extend the usability of colorization framework to the restoration and enhancement of classical color cartoons. By seamlessly manipulating hue, saturation, and brightness of outlines/regions local color contrast can be reasonably adjusted yielding much pleasant look of the whole animation.

2. **Cartoon-by-example.**

   A serious issue in computer assisted cartooning is to mimic well-known artistic styles in order to create new poses and characters undistinguishable from the original. One possible approach to this problem is to combine fragments of the original artwork.

Using standard image manipulation tools this task is laborious and non-intuitive. The proposed cartoon analysis framework allows to pre-process the input image so that the fragment extraction requires only few sloppy selection scribbles. Moreover, thanks to adaptive outline reconstruction and high-quality vectorization scheme compelling visual quality of the final composition is preserved. For the fragment positioning an intuitive scribble-based composition tool has been proposed allowing to define relative scale and free-form deformation at one snap. Altogether, new cartoon drawings can be produced within a few seconds. Moreover, the framework can be easily extended for animation where the key-frame composition scribbles are interpolated to reach smooth inbetweening.

3. **Video compression.**

   In last decades only little effort has been spent on the development of native cartoon oriented video compression techniques despite of constant public interest on this topic. The key issue that hampers the progress in this area was persisting inability to automatically produce precise `2.5D` decomposition of the input video sequence. The framework proposed in this thesis is the first one that allows to break this fundamental barrier and proceeds toward first native cartoon oriented video codec that outperforms standard video compression approaches.

   It has been shown that large savings can be reached by storing whole background layer as a single image and by exploiting significant redundancy in the animation together with adaptive shape coding heuristics. Moreover, thanks to the static background and vector-based representation of the foreground layer high-quality anti-aliased playback can be performed in real-time on a commodity graphics hardware in the resolution independent manner. The only limiting factor is a bilinear interpolation in the background layer.

6.1 Future work

Drawing styles similar to *Radek Pilař*'s represent an ideal case for the proposed framework. Other styles that do not satisfy the basic assumption on region homogeneity and outline continuity require additional processing that has not been discussed in this thesis. One possible extensions is to exploit local mean-shift segmentation for piecewise homogenous regions or to incorporate techniques similar to [136] that handle regions with patterns and larger gaps between outlines.

The proposed framework completely fails when outlines are drawn close together or when the image contains lots of small unstructured and/or non-homogenous regions. In Figure 6.1 several examples of such challenging drawing styles are depicted. Resulting vectorization for such images appears visually corrupted as compared to the original drawing. For this type of images completely different approaches need to be developed. Systems like *ARDECO* [97] represent a good starting point for further research.

Figure 6.1: **Several examples of drawing styles that are not suitable for the framework proposed in this thesis**: problems arise when the input image contains lots of unstructured outlines, small and inhomogeneous regions, soft shadows and color gradients, dark coalescent outlines in the background, etc.

Bibliography

[1] A. Agarwala. SnakeToonz: A semi-automatic approach to creating cel animation from video. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering*, pages 139–146, 2002.

[2] A. Ahmadyfard and J. Kittler. Region-based object recognition: Pruning multiple representations and hypotheses. In *Proceedings of British Machine Vision Conference*, pages 745–754, 2000.

[3] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C(23):90–93, 1974.

[4] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. In *ACM SIGGRAPH Conference Proceedings*, pages 157–164, 2000.

[5] D. J. Bancroft. Advanced and economical telecine technology for global DTV production. In *Proceedings of Broadcast Engineering Conference*, 2000.

[6] W. A. Barrett and A. S. Cheney. Object-based image editing. *ACM Transactions on Graphics*, 21(3):777–784, 2002.

[7] P. J. L. van Beek and A. M. Tekalp. Object-based video coding using forward tracking 2-D mesh layers. In *Proceedings of SPIE Visual Communications and Image Processing*, pages 699–710, 1997.

[8] T. Beier and S. Neely. Feature-based image metamorphosis. In *ACM SIGGRAPH Conference Proceedings*, pages 35–42, 1992.

[9] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24):509–522, 2002.

[10] G. Bendazzi. *Cartoons: One Hundred Years of Cinema Animation*. Indiana University Press, Bloomington, IN, 1995.

[11] B. Besserer and C. Thiré. Detection and tracking scheme for line scratch removal in an image sequence. In *Proceedings of European Conference on Computer Vision*, pages 264–275, 2004.

[12] G. di Blasi and D. R. Recupero. Fast colorization of gray images. In *Processings of Eurographics Italian Chapter*, 2003.

[13] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, 1986.

[14] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *Proceedings of Internation Conference on Computer Vision*, volume 1, pages 105–112, 2001.

[15] C. Bregler, L. Loeb, E. Chuang, and H. Deshpande. Turning to the masters: Motion capturing cartoons. *ACM Transactions on Graphics*, 21(3):399–407, 2002.

[16] M. Brown and D. G. Lowe. Recognising panoramas. In *Proceedings of International Conference on Computer Vision*, volume 2, pages 1218–1225, 2003.

[17] V. Bruni and D. Vitulano. Old movies noise reduction via wavelets and wiener filters. In *Proceedings of WSCG – International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, pages 65–72, 2004.

[18] I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D. Salesin, J. Seims, R. Szeliski, and K. Toyama. Performance-driven hand-drawn animation. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering*, pages 101–108, 2000.

[19] M. Burrows and D. J. Wheeler. Block-sorting lossless data compression algorithm. Technical Report 124, SRC, Palo Alto, USA, 1994.

[20] N. Burtnyk and M. Wein. Computer-generated key frame animation. *Journal of the Society Motion Picture and Television Engineers*, 80(3):149–153, 1971.

[21] N. Burtnyk and M. Wein. Computer animation of free form images. In *ACM SIGGRAPH Conference Proceedings*, pages 78–80, 1975.

[22] N. Burtnyk and M. Wein. Interactive skeleton techniques for enhancing motion dynamics in key frame animation. *Communications of the ACM*, 19(10):564–569, 1976.

[23] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[24] V. Caselles, B. Coll, and J. M. Morel. Geometry and color in natural images. *Journal of Mathematical Imaging and Vision*, 16(2):89–105, 2002.

[25] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22(1):61–79, 1997.

[26] E. Catmull. The problems of computer-assisted animation. In *ACM SIGGRAPH Conference Proceedings*, pages 348–353, 1978.

[27] M. Chambah, B. Besserer, and P. Courtellemont. Latest results in digital color film restoration. *Machine Graphics & Vision International Journal*, 11(2–3):363–395, 2002.

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

[29] H. Chen, Z. Liu, C. Rose, Y. Xu, H.-Y. Shum, and D. Salesin. Example-based composite sketching of human portraits. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering*, pages 95–102, 2004.

[30] J. S. Chen, A. Huertas, and G. Medioni. Fast convolution with Laplacian-of-Gaussian masks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):584–590, 1987.

[31] Q. Chen, Z. Wu, F. Tian, H. S. Seah, J. Qiu, and K. Melikhov. Vectorization of raster line-drawings in cartoons. In *Proceedings of International Conference on Computer Animation and Social Agents*, 2005.

[32] T. Chen, Y. Wang, V. Schillings, and C. Meinel. Grayscale image matting and colorization. In *Proceedings of Asian Conference on Computer Vision*, pages 1164–1169, 2004.

[33] Y.-S. Chen, Y.-P. Hung, and C.-S. Fuh. Fast block matching algorithm based on the winner-update strategy. *IEEE Transactions on Image Processing*, 10(8):1212–1222, 2001.

[34] H.-D. Cheng and Y. Sun. A hierarchical approach to color image segmentation using homogenity. *IEEE Transactions on Image Processing*, 9(12):2071–2082, 2000.

[35] Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski. A Bayesian approach to digital matting. In *Processings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 264–271, 2001.

[36] H. Chui and A. Rangarajan. A new algorithm for non-rigid point matching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 44–51, 2000.

[37] J. H. Clark. A fast scan-line algorithm for rendering parametric surfaces. *IEEE Transactions on Image Processing*, 13(2):289–299, 1979.

[38] J. J. Clark. Authenticating edges produced by zero-crossing algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(1):43–57, 1989.

[39] R. J. Clarke. Image and video compression: A survey. *International Journal of Imaging Systems and Technology*, 10(1):20–32, 1999.

[40] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[41] C. Concolato, J. C. Dufourd, and J. C. Moissinac. Creating and encoding of cartoons using MPEG-4 BIFS: Methods and results. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(11):1129–1135, 2003.

[42] R. Cooper. Colorization and moral rights: Should the united states adopt unified protection for artists? *Journalism Quarterly (Urbana, Illinois)*, 68:465–473, 1990.

[43] W. T. Corrêa, R. J. Jensen, C. E. Thayer, and A. Finkelstein. Texture mapping for cel animation. In *ACM SIGGRAPH Conference Proceedings*, pages 435–446, 1998.

[44] I. Drori, D. Cohen-Or, and H. Yeshurun. Example-based style synthesis. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, pages 143–150, 2003.

[45] C. X. Durand. The TOON project: Requirements for a computerized 2D animation system. *Computers & Graphics*, 15(2):285–293, 1991.

[46] T. Ebrahimi and C. Horne. MPEG-4 natural video coding – An overview. *Signal Processing: Image Communication*, 15:365–385, 2000.

[47] G. Fan and W.-K. Cham. Model-based edge reconstruction for low bit-rate wavelet-compressed images. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(1):120–132, 2000.

[48] J.-D. Fekete, E. Bizouarn, E. Cournarie, T. Galas, and F. Taillefer. TicTacToon: A paperless system for professional 2D animation. In *ACM SIGGRAPH Conference Proceedings*, pages 79–90, 1995.

[49] P. F. Felzenszwalb. Representation and detection of deformable shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2):208–220, 2005.

[50] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. In *Processings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 261–268, 2004.

[51] F. di Fiore, P. Schaeken, K. Elens, and F. van Reeth. Automatic in-betweening in computer assisted animation by exploiting 2.5D modelling techniques. In *Proceedings of Computer Animation*, pages 192–200, 2001.

[52] F. di Fiore and F. van Reeth. Multi-level performance-driven stylised facial animation. In *Proceedings of International Conference on Computer Animation and Social Agents*, pages 73–78, 2005.

[53] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381–391, 1954.

[54] B. Flach and M. I. Schlesinger. A class of solvable consistent labelling problems. pages 462–471, 2000.

[55] R. Frankot and R. Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):439–4515, 1988.

[56] J. Fransens, F. di Fiore, and F. van Reeth. The reconstruction of missing frames in historical films, a layered approach. In *Proceedings of GraphiCon – International Conference on Computer Graphics & Vision*, pages 63–69, 2005.

[57] W. T. Freeman, J. B. Tenenbaum, and E. Pasztor. An example-based approach to style translation for line drawings. Technical Report TR99-11, MERL, 1999.

[58] M. Frigo and S. G. Johnson. FFTW: Library for computing the discrete Fourier transform, 2005. `http://www.fftw.org`.

[59] D. Geiger, T.-L. Liu, and R. V. Kohn. Representation and detection of deformable shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):86–99, 2003.

[60] M. Gleicher. Image snapping. In *ACM SIGGRAPH Conference Proceedings*, pages 183–190, 1995.

[61] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley Publishing, Reading, Massachusetts, 2nd edition, 1987.

[62] A. Goshtasby. On edge focusing. *Image and Vision Computing*, 12(4):247–256, 1994.

[63] W. van Haevre, F. di Fiore, and F. van Reeth. Uniting cartoon textures with computer assisted animation. In *Proceedings of International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pages 245–253, 2005.

[64] K. Haris, S. N. Estradiadis, N. Maglaveras, and A. K. Katsaggelos. Hybrid image segmentation using watersheds and fast region merging. *IEEE Transactions on Image Processing*, 7(12):1684–1699, 1998.

[65] M. D. Heath, S. Sarkar, T. Sanocki, and K. W. Bowyer. Comparison of edge detectors: A methodology and initial study. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 143–148, 1996.

[66] M. Held. FIST: Fast industrial-strength triangulation of polygons. *Algorithmica*, 30(4):563–596, 2001.

[67] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. Salesin. Image analogies. In *ACM SIGGRAPH Conference Proceedings*, pages 327–340, 2001.

[68] A. Hertzmann, N. Oliver, B. Curless, and S. Seitz. Curve analogies. In *Proceedings of Eurographics Workshop on Rendering*, pages 233–245, 2002.

[69] T. Horiuchi. Estimation of color for gray-level image by probabilistic relaxation. In *Proceedings of IEEE International Conference on Pattern Recognition*, pages 867–870, 2002.

[70] T. Horiuchi. Colorization algorithm using probabilistic relaxation. *Image and Vision Computing*, 22(3):197–202, 2004.

[71] T. Horiuchi and S. Hirano. Colorization algorithm for grayscale image by propagating seed pixels. In *Proceedings of IEEE International Conference on Pattern Recognition*, pages 457–460, 2003.

[72] T. Horiuchi and H. Kotera. Colorization algorithm for monochrome image sequences using optical flow. In *Proceedings of Color Imaging Conference*, pages 140–143, 2004.

[73] T. Horiuchi and H. Kotera. Colorization algorithm for monochrome video by sowing color seeds. *Journal of Imaging Science and Technology*, 50(3):243–250, 2006.

[74] Y.-C. Huang, Y.-S. Tung, J.-C. Chen, S.-W. Wang, and J.-L. Wu. An adaptive edge detection based colorization algorithm and its applications. In *Proceedings of ACM International Conference on Multimedia*, pages 351–354, 2005.

[75] A. Huertas and G. Medioni. Detection of intensity changes with subpixel accuracy using Laplacian-Gaussian masks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):651–664, 1986.

[76] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3D freeform design. In *ACM SIGGRAPH Conference Proceedings*, pages 409–416, 1999.

[77] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics*, 24(3):1134–1141, 2005.

[78] Independent JPEG Group. JPEG 2000 image coding system: Core coding system, 2004. International Standard: ISO/IEC 15444-1.

[79] R. Irony, D. Cohen-Or, and D. Lischinski. Colorization by example. In *Proceedings of Eurographics Symposium on Rendering*, pages 201–210, 2005.

[80] J. Jang and S. Rajala. Texture segmentation-based image coder incorporating properties of the human visual system. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 2753–2756, 1991.

[81] Y. Ji, H.-B. Liu, X.-K. Wang, and Y.-Y. Tang. Color transfer to greyscale images using texture spectrum. In *Proceedings of International Conference on Machine Learning and Cybernetics*, volume 7, pages 4057–4061, 2004.

[82] P.-M. Jodoin, E. Epstein, M. Granger-Pichi, and V. Ostromoukhov. Hatching by example: A statistical approach. In *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering*, pages 29–36, 2002.

[83] S. F. Johnston. Lumo: Illumination for cel animation. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering*, pages 45–52, 2002.

[84] L. Joyeux, S. Boukir, B. Besserer, and O. Buisson. Reconstruction of degraded image sequences. Application to film restoration. *Image and Vision Computing*, 19(8):503–516, 2001.

[85] C. de Juan and B. Bodenheimer. Cartoon textures. In *Proceedings of Eurographics Symposium on Computer Animation*, pages 267–276, 2004.

[86] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[87] Y. Kho and M. Garland. Sketching mesh deformations. In *Proceedings of Symposium on Interactive 3D Graphics and Games*, pages 147–154, 2005.

[88] S. L. Kilthau, M. S. Drew, and T. Möller. Full search content independent block matching based on the fast fourier transform. In *Proceedings of IEEE International Conference on Image Processing*, pages 669–672, 2002.

[89] D. King. Implementation of the Marr-Hildreth theory of edge detection. Technical Report ISG-102, The University of Southern California, 1982.

[90] J. Kittler and J. Illingworth. Relaxation labelling algorithms — a review. *Image and Vision Computing*, 3(11):206–216, 1985.

[91] A. C. Kokaram. *Motion Picture Restoration: Digital Algorithms for Artefact Suppression in Degraded Motion Picture Film and Video.* Springer-Verlag, Berlin, Germany, 1998.

[92] A. C. Kokaram, R. D. Morris, W. J. Fitzgerald, and P. J. W. Rayner. Interpolation of missing data in image sequences. *IEEE Transactions on Image Processing*, 4(11):1509–1519, 1995.

[93] A. Kort. Computer aided inbetweening. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering*, pages 125–132, 2002.

[94] C. D. Kuglin and D. C. Hines. The phase correlation image alignment method. In *Proceedings of IEEE International Conference on Cybernetics and Society*, pages 163–165, 1975.

[95] V. Kwatra and J. Rossignac. Space-time surface simplification and edgebreaker compression for 2D cel animations. *International Journal on Shape Modeling*, 8(2):119–137, 2002.

[96] O. Kwon and R. Chellappa. Segmentation-based image compression. *Optical Engeneering*, 32(7):1581–1587, 1993.

[97] G. Lecot and B. Levy. ARDECO: Automatic Region DEtection and Conversion. In *Proceedings of Eurographics Symposium on Rendering*, pages 349–360, 2006.

[98] W. S. Lee and A. A. Kassim. Image and animation cel coding using wedgelets and beamlets. In *Proceedings of SPIE Visual Communications and Image Processing*, volume 5150, pages 1460–1469, 2003.

[99] T. M. Lehmann, A. Kaser, and R. Repges. A simple parametric equation for pseudocoloring grey scale images keeping their original brightness progression. *Image and Vision Computing*, 15(3):251–257, 1997.

[100] F. Leibowitz. Movie colorization and the expression of mood. *Journal of Aesthetics and Art Criticism (Cleveland, Ohio)*, 49(4):363–364, 1991.

[101] J. Lenburg. *The Encyclopedia of Animated Cartoons (2nd edition).* Facts on File, New York, NY, 1999.

[102] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23(3):689–694, 2004.

[103] M. Levoy. A color animation system based on the multiplane technique. In *ACM SIGGRAPH Conference Proceedings*, pages 65–71, 1977.

[104] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Transactions on Graphics*, 23(3):303–308, 2004.

[105] T. Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 21(2):224–270, 1994.

[106] U. Lipowezky. Grayscale aerial and space image colorization using texture classification. *Pattern Recognition Letters*, 27(4):275–286, 2006.

[107] P. C. Litwinowicz. Inkwell: A 2-D animation system. In *ACM SIGGRAPH Conference Proceedings*, pages 113–122, 1991.

[108] C. Loop and J. Blinn. Resolution independent curve rendering using programmable graphics hardware. *ACM Transactions on Graphics*, 24(3):1000–1009, 2005.

[109] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[110] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[111] J. S. Madeira, A. Stork, and M. H. Grob. An approach to computer-supported cartooning. *The Visual Computer*, 12(1):1–17, 1996.

[112] W. Markle. The development and application of colorization. *SMPTE Journal*, pages 632–635, 1984.

[113] W. Markle and B. Hunt. Coloring a black and white signal using motion detection, 1991. Canadian Patent, No. CA 01291260.

[114] D. Marr and E. C. Hildreth. Theory of edge detection. In *Proceedings of Royal Society*, volume B207, pages 187–217, 1980.

[115] K. Melikhov, F. Tian, H. S. Seah, Q. Chen, and J. Qiu. Frame skeleton based auto-inbetweening in computer assisted cel animation. In *Proceedings of International Conference on Cyberworlds*, pages 216–223, 2004.

[116] T. Milliron, R. J. Jensen, R. Barzel, and A. Finkelstein. A framework for geometric warps and deformations. *ACM Transactions on Graphics*, 21(1):20–51, 2002.

[117] E. N. Mortensen and W. A. Barrett. Toboggan-based intelligent scissors with a four parameter edge model. In *Processings of IEEE Computer Vision and Pattern Recognition*, volume 2, pages 452–458, 1999.

[118] MPEG-2 Video Group. Information technology – generic coding of moving pictures and associated audio: Part 2 – Video, 1995. International Standard: ISO/IEC 13818-2.

[119] K. M. Nam, J.-S. Kim, R.-H. Park, and Y. S. Shim. A fast hierarchical motion vector estimation algorithm using mean pyramid. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(4):344–351, 1995.

[120] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or. A sketch-based interface for detail-preserving mesh editing. *ACM Transactions on Graphics*, 24(3):1142–1147, 2005.

[121] D. Nie, L. Ma, S. Xiao, and X. Xiao. Grey-scale image colorization by local correlation based optimization algorithm. In *Proceedings of International Conference Visual Information and Information Systems*, pages 13–23, 2005.

[122] M. Nitzberg, D. Mumford, and T. Shiota. *Filtering, segmentation, and depth*, volume 662 of *Lecture Notes in Computer Science*. Springer-Verlag Inc., New York, NY, USA, 1993.

[123] H. Noda, H. Korekuni, N. Takao, and M. Niimi. Bayesian colorization using MRF color image modeling. In *Proceedings of Pacific-Rim Conference on Multimedia*, pages 889–899, 2005.

[124] J.-M. Odobez and P. Bouthemy. Robust multiresolution estimation of parametric motion models. *Journal of Visual Communication and Image Representation*, 6(4):348–365, 1995.

[125] A. Oliva, A. Torralba, and P. G. Schyns. Hybrid images. *ACM Transactions on Graphics*, 25(3):527–532, 2006.

[126] Z. Pan, Z. Dong, and 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*, pages 515–519, 2004.

[127] H.-K. Pao, D. Geiger, and N. Rubin. Measuring convexity for figure/ground separation. In *Proceedings of IEEE International Conference on Computer Vision*, pages 948–955, 1999.

[128] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on Graphics*, 22(3):313–318, 2003.

[129] L. Petrović, B. Fujito, L. Williams, and A. Finkelstein. Shadows for cel animation. In *ACM SIGGRAPH Conference Proceedings*, pages 511–516, 2000.

[130] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C: The art of scientific computing*. Cambridge University Press, 1992.

[131] G. Qiu and J. Guan. Color by linear neighborhood embedding. In *Proceedings of IEEE International Conference on Image Processing*, volume 3, pages 988–991, 2005.

[132] J. Qiu, H. S. Seah, F. Tian, Q. Chen, and K. Melikhov. Computer-assisted auto coloring by region matching. In *Proceedings of Pacific Conference on Computer Graphics and Applications*, pages 175–185, 2003.

[133] J. Qiu, H. S. Seah, F. Tian, Q. Chen, and Z. Wu. Enhanced auto coloring with hierarchical region matching. *Computer Animation and Virtual Worlds*, 16(3–4):463–473, 2005.

[134] J. Qiu, H. S. Seah, F. Tian, Z. Wu, and Q. Chen. Topology enhanced feature-based auto coloring for computer-assisted cel animation. In *Proceedings of IASTED International Conference on Computer Graphics and Imaging*, pages 26–32, 2004.

[135] J. Qiu, H. S. Seah, F. Tian, Z. Wu, and Q. Chen. Feature- and region-based auto painting for 2D animation. *The Visual Computer*, 21(11):928–944, 2005.

[136] Y. Qu, T. T. Wong, and P. A. Heng. Manga colorization. *ACM Transactions on Graphics*, 25(3):1214–1220, 2006.

[137] B. S. Reddy and B. N. Chatterji. An FFT-based technique for translation, rotation and scale-invariant image registration. *IEEE Transactions on Computers*, 5(8):1266–1271, 1996.

[138] F. van Reeth. Integrating 2.5D computer animation techniques for supporting traditional animation. In *Proceedings of Computer Animation*, pages 118–125, 1996.

[139] M. M. Reid, R. J. Millar, and N. D. Black. Second-generation image coding: An overview. *ACM Computing Surveys*, 29(1):3–29, 1997.

[140] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Transactions on Computer Graphics and Applications*, 21(5):34–41, 2001.

[141] E. Riskin. Optimal bit allocation via the generalized BFOS algorithm. *IEEE Transactions on Information Theory*, 37(2):400–402, 1991.

[142] A. Roberts. Fast and accurate multigrid solution of poissons equation using diagonally oriented grids. *ANZIAM*, E(43):1–36, 2001.

[143] A. Rosenfeld and A. C. Kak. *Digital Picture Processing*, volume 1. Academic Press, Orlando, USA, 1982.

[144] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):47–61, 1999.

[145] J. Rossignac, A. Megahed, and B. O. Schneider. Interactive inspection of solids: Cross-Sections and interferences. In *ACM SIGGRAPH Conference Proceedings*, pages 353–360, 1992.

[146] C. Rother, V. Kolmogorov, and A. Blake. Grabcut - Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004.

[147] D. L. Ruderman, T. W. Cronin, and C. C. Chiao. Statistics of cone responses to natural images: Implications for visual coding. *Journal of Optical Society of America*, 15(8):2036–2045, 1998.

[148] D. Salomon. *Data compression: The complete reference*. Springer Verlag, 1998.

[149] G. Sapiro. Inpainting the colors. In *Proceedings of IEEE International Conference on Image Processing*, volume 2, pages 698–701, 2005.

[150] E. Saund, D. Fleet, D. Larner, and J. Mahoney. Perceptually-supported image editing of text and graphics. In *Proceedings of ACM Symposium on User Interface Software and Technology*, pages 183–192, 2003.

[151] S. Schaefer, T. McPhail, and J. Warren. Image deformation using moving least squares. *ACM Transactions on Graphics*, 25(3):533–540, 2006.

[152] P. Schallauer, A. Pinz, and W. Haas. Automatic restoration algorithms for 35mm film. *Videre: Journal of Computer Vision Research*, 1(3):60–85, 1999.

[153] H. S. Seah and B. C. Chua. A skeletal line joining algorithm. In *Proceedings of the Computer Graphics International*, pages 62–73, 1994.

[154] H. S. Seah and T. Feng. Computer-assisted coloring by matching line drawings. *The Visual Computer*, 16(5):289–304, 2000.

[155] A. Selle, A. Mohr, and S. Chenney. Cartoon rendering of smoke animations. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering*, pages 57–60, 2004.

[156] J. Serra. *Image Analysis and Mathematical Morphology*, volume 1. Academic Press, Oval Road, London, UK, 4th edition, 1993.

[157] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3463, 1993.

[158] J. Shi and J. Malik. Normalized cuts and image segmentation,. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[159] S. M. Smith and J. M. Brady. SUSAN - A new approach to low-level image processing. *International Journal of Computer Vision*, 32(1):45–78, 1997.

[160] M. Soryani and R. J. Clarke. Segmented coding of digital image sequences. In *Proceedings of IEEE Communications, Speech and Vision*, volume 139(2), pages 212–218, 1991.

[161] G. E. Sotak and 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):147–189, 1989.

[162] F. Spaan, R. L. Lagendijk, and J. Biermond. Shape coding using polar coordinates and the discrete cosine transform. In *Proceedings of International Conference on Image Processing*, pages 516–519, 1997.

[163] G. Stern. SoftCel - An application of raster scan graphics to conventional cel animation. In *ACM SIGGRAPH Conference Proceedings*, pages 284–288, 1979.

[164] Y.-W. Tai, J. Jia, and C.-K. Tang. Local color transfer via probabilistic segmentation by expectation-maximization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 747–754, 2005.

[165] T. Takahama, T. Horiuchi, and H. Kotera. Improvement on colorization accuracy by partitioning algorithm in CIELAB color space. In *Processings of Pacific Rim Conference on Multimedia*, pages 794–801, 2004.

[166] B. Tang, G. Sapiro, and V. Caselles. Color image enhancement via chromaticity diffusion. *IEEE Transactions Image Processing*, 10(5):701–707, 2001.

[167] L. Tenze, G. Ramponi, and S. Carrato. Blotches correction and contrast enhancement for old film pictures. In *Proceedings of International Conference on Image Processing*, volume 2, pages 660–663, 2000.

[168] M. Thorne, D. Burke, and M. van de Panne. Motion doodles: An interface for sketching character motion. *ACM Transactions on Graphics*, 23(3):424–431, 2004.

[169] C. Tomasi and T. Kanade. Shape and motion from image streams: A factorization method (part 2): Detection and trcking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, School of Computer Science, 1991.

[170] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

[171] C. Vantighem, N. Laurent, D. Deckeur, and V. Plantinet. Vector Eye: Raster to SVG converter, 2003. `http://www.siame.com`.

[172] L. F. M. Vieira, R. D. Vilela, E. R. do Nascimento, F. A. Fernandes, R. L. Carceroni, and A. de A. Araújo. Automatically choosing source color images for coloring grayscale images. In *Processings of Brazilian Symposium on Computer Graphics and Image Processing*, pages 151–158, 2003.

[173] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[174] B. A. Wallace. Merging and transformation of raster images for cartoon animation. In *ACM SIGGRAPH Conference Proceedings*, pages 253–262, 1981.

[175] G. K. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.

[176] H. Wang and H. Li. Cartoon motion capture by shape matching. In *Proceedings of Pacific Conference on Computer Graphics and Applications*, pages 454–456, 2002.

[177] J. Wang and M. F. Cohen. An iterative optimization approach for unified image segmentation and matting. In *Processings of IEEE International Conference on Computer Vision*, pages 936–943, 2005.

[178] J. Wang, Y. Xu, H.-Y. Shum, and M. F. Cohen. Video tooning. *ACM Transactions on Graphics*, 23(3):574–583, 2004.

[179] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 1994.

[180] Web-3D Consortioum. Virtual reality modeling language, 2004. International Standards: ISO/IEC 14772-1:1997 and ISO/IEC 14772-2:2004.

[181] M. Weber and B. Herzog. Autotrace, 2004. `http://autotrace.sourceforge.net`.

[182] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *Processings of IEEE International Conference on Computer Vision*, pages 975–982, 1999.

[183] Y. Weiss. Deriving intrinsic images from image sequences. In *Processings of International Conference on Computer Vision*, pages 68–75, 2001.

[184] T. Welsh, M. Ashikhmin, and K. Mueller. Transferring color to greyscale images. *ACM Transactions on Graphics*, 21(3):277–280, 2002.

[185] A. P. Witkin. Scale space filtering. In A. P. Pentland, editor, *From Pixels to Predicates: Recent Advances in Computational and Robot Vision*, pages 5–19, 1986.

[186] A. H. Wong and C. Chen. Comparison of ISO MPEG-1 and MPEG-2 video-coding standards. In *Proceedings of SPIE Visual Communications and Image Processing*, volume 2094, pages 1436–1448, 1993.

[187] M. Woo, T. Davis, and M. B. Sheridan. *OpenGL Programming Guide: The Official Guide to Learning OpenGL*. Addison-Wesley, 1999.

[188] D. N. Wood, A. Finkelstein, J. F. Hughes, C. E. Thayer, and D. H. Salesin. Multiperspective panoramas for cel animation. In *ACM SIGGRAPH Conference Proceedings*, pages 243–250, 1997.

[189] Y. Yang, N. P. Galatsanos, and A. K. Katsaggelos. Projection-based spatially adaptive reconstruction of block-transform compressed images. *IEEE Transactions on Image Processing*, 4(7):896–908, 1995.

[190] L. Yatziv and G. Sapiro. Fast image and video colorization using chrominance blending. *IEEE Transactions on Image Processing*, 15(5):1120–1129, 2006.

[191] J. Ying and L. Ji. Pattern recognition based color transfer. In *Proceedings of International Conference on Computer Graphics, Imaging and Visualization*, pages 55–60, 2005.

[192] J. Zaletelj, R. Pecci, F. Spaan, A. Hanjalic, and R. L. Lagendijk. Rate distortion optimal contour compression using cubic B-splines. In *Proceedings of European Signal Processing Conference*, pages 1497–1500, 1998.

[193] X. Zeng, W. Chen, and Q. Peng. A novel unified variational image editing model. *Journal of Computer Science and Technology*, 21(2):224–231, 2006.

[194] S. Zhang, L. Li, and H. S. Seah. Vectorization of digital images using algebraic curves. *Computers & Graphics*, 22(1):91–101, 1998.

[195] D. Ziou and S. Tabbone. Edge detection techniques – An overview. *International Journal of Pattern Recognition and Image Analysis*, 8(4):537–559, 1998.

[196] B. Zitová and J. Flusser. Image registration methods: A survey. *Image and Vision Computing*, 21(11):977–1000, 2003.

[197] J. J. Zou and H. Yan. Cartoon image vectorization based on shape subdivision. In *Proceedings of Computer Graphics International*, pages 225–231, 2001.

Publications of the author

Refereed journal papers

[A.1] Daniel Sýkora, Jan Buriánek, and Jiří Žára. Colorization of Black-and-White Cartoons. *Image and Vision Computing*, 23(9):767–852, Elsevier, September 2005.

Reviewed conference papers

[A.2] Daniel Sýkora, Jan Buriánek, and Jiří Žára. Video Codec for Classical Cartoon Animations with Hardware Accelerated Playback. In *Proceedings of the 1st International Symposium on Visual Computing*, pages 43–50, Springer, Lake Tahoe, Nevada, December 2005.

[A.3] Daniel Sýkora, Jan Buriánek, and Jiří Žára. Sketching Cartoons by Example. In *Proceedings of the 2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 27–34, Dublin, Ireland, August 2005.

[A.4] Daniel Sýkora, Jan Buriánek, and Jiří Žára. Unsupervised Colorization of Black-and-White Cartoons. In *Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering*, ACM SIGGRAPH, pages 121–127, Annecy, France, June 2004.

[A.5] Daniel Sýkora, Jan Buriánek, and Jiří Žára. Segmentation of Black-and-White Cartoons. In *Proceedings of the 19th Spring Conference on Computer Graphics*, ACM SIGGRAPH, pages 233–230, Budmerice, Slovakia, April 2003.

Other publications

[A.6] Jan Buriánek and Daniel Sýkora. Jak Rumcajs k barvám přišel. In *PIXEL*, 89(5):30–33, May 2004.

[A.7] Daniel Sýkora, Jan Buriánek, and Jiří Žára. Color for Black-and-White Cartoons. In *Proceedings of Workshop*, ČVUT, pages 184–185, Prague, Czech Republic, March 2004.