Matthias Reiterer, BSc

Learning to consistently recolor video scenes

MASTER'S THESIS

to achieve the university degree of
Diplom-Ingenieur

Master's degree programme
Information and Computer Engineering

submitted to

Graz University of Technology

Advisor
Dipl.-Ing. Erich Kobler
Institute of Computer Graphics and Vision

Supervisor
Univ.-Prof. Dipl.-Ing. Dr.techn. Thomas Pock
Institute of Computer Graphics and Vision

Graz, Austria, July 2020

Abstract

Historically, adding color to monochrome recorded photographs or movies was done manually in a tedious process. Nowadays, different algorithms have been developed to automate this problem, where most of the recent colorization approaches are based on deep learning.

In this work we investigate learning based methods to persistently colorize monochromatic video sequences. We begin our considerations on single image colorization, where we color grayscale images based on a color reference image. For this first approach we use the concept of optimal transport. Based on the initial results, we devote to grayscale video colorization using convolutional neural networks. Given an initial colored first frame and a colored last frame of a video sequence, we colorize the rest of the scene using a combination of global and local color transfer stages. All combination techniques are numerically evaluated and various colorization results highlight the quality of our approach.

Kurzfassung

Noch vor einigen Jahrzehnten wurde das Einfärben von monochrom aufgenommenen Bildern und Filmen in mühevoller Handarbeit erledigt. Heute gibt es unterschiedliche Algorithmen, die diesen Prozess unterstützen, wobei die meisten dieser Algorithmen auf Deep-Learning Ansätzen basieren.

In dieser Arbeit beschäftigen wir uns mit Methoden des Einfärbens von Schwarz-Weiß-Videosequenzen, die auf maschinellem Lernen beruhen. Wir befassen uns zunächst mit dem Einfärben einzelner Schwarz-Weiß-Bilder. Mit Hilfe eines farbigen Referenzbildes werden dessen Farben auf das monochrome Bild übertragen. Als Grundlage dieser Methode verwenden wir das Konzept des optimalen Transports. Ausgehend von den erzielten Ergebnissen verwenden wir für das Einfärben ganzer Schwarz-Weiß-Videosequenzen Ansätze, die mittels "faltenden neuronalen Netzwerken" funktionieren. Schwarz-Weiß-Videosequenzen mit farbigem erstem und letztem Bild werden mittels Kombination von globalen und lokalen Farbtransfer-Algorithmen eingefärbt. Um die Qualität unseres Ansatzes zu zeigen, werden die unterschiedlichen Kombinationstechniken numerisch evaluiert und diverse Einfärbungsresultate präsentiert.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.

The text document uploaded to TUGRAZonline is identical to the present master's thesis dissertation.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Ort	Datum	Unterschrift

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor Erich Kobler. He continuously supported me during this thesis and helped me when I was lost. In those times he encouraged me to keep going and rekindled my motivation. The meetings with him helped me to gain lots of knowledge in the field of computer vision and machine learning. A big thank you for all of this.

Besides that, I would like thank my fellow study colleges, especially Thomas Schrotter, for their support and helpful discussions.

A special thanks goes out to my family. Without their support, it wouldn't have been possible for me to come this far. I thank my sister and brother and particularly my mother for their patience and support, whenever I needed them in my life. Also the time that I spent at my brothers house construction site and partying with my friends were a welcome alternation when my head was too full.

Finally, I would like to thank Helga for her support and love.

Contents

List of Figures

List of Tables

1 Introduction

Contents

For a long time, the colorization of monochrome images and videos was a laborious problem. The development of algorithms automated and improved this process. Since most of the time the real color of the scene is not known, the colorization of a single color medium depends on either the artist who is recoloring manually or reference images where colors can be extracted.

In this work, we start by studying the problem of single image colorization. Later we focus our considerations on video colorization. As the title of this thesis suggests, "Learning to consistently recolor video scenes", we want to establish a process to persistently transfer color to grayscale video sequences with the help of modern learning based methods.

1.1 Image Colorization

Today, almost everyone is able to record images anytime. People take out their smartphones, open their camera app and in a blink of an eye, an image is taken. Retouch some parts, add fancy filters - Et voilà! Ready to share!

Historically, the recording of images was more complicated. The first cameras had exposure times of multiple hours and the first photographs were limited to monochrome. There are still many monochromatic images left from previous times. In order to reimagine this historic events, people started to colorize them. In the beginning this colorization was done by hand using conventional painting techniques, as can be seen in Figure 1.1. Here, a historical image (Fig. 1.1a) was colorized (Fig. 1.1b) by hand. This

colorization was done between 1875 and 1885. Later, with the development of digital computing this process has become much easier. Old single color images are scanned and color gets added with the help of an image processing software. In recent years, new methods have been developed that make the colorization even more convenient.

(a) **(b)**

Figure 1.1: Image colorization. A historical image colorization. (a) is the monochromatic original recording, (b) the colorized version. Images are taken from Wikimedia Commons[1].

1.2 Video Colorization

The extension of image colorization is video colorization. As one can imagine, historically, this process of colorizing multiple single image by hand must have been very laborious. Nowadays, algorithms have been proposed that propagate colors through frames, where the assigned color should be coherent throughout the sequence.

The colorization of movies is sometimes used for historical documentations, e.g. footage from the First and Second World War or for old movies. There is often no definite solution when recoloring grayscale source material. For some objects in the scene the main color may be obvious, but for color gradients and unknown objects, different colorizations can be possible. Especially for historical film material, people investigate and search in museums to get the real color of objects, *e.g.* the color of the tunics of the French soldiers in the First World War. This may be important to achieve a historical correct recoloring.

[1] `https://commons.wikimedia.org/wiki/File:Takaboko_Island_bw.jpg` *resp.* `https://commons.wikimedia.org/wiki/File:Takaboko_Island_hc.jpg`, Accessed: 4^{th} May 2020

2 Related Work

Contents

In this chapter, we describe modern, state-of-the-art colorization approaches. The colorization of grayscale images and videos has become a fundamental part in image restoration. Many different approaches have been developed from the computer vision community. As a starting point we give an overview of different methods to colorize single grayscale images (Section 2.1). We then present different methods to colorize grayscale videos (Section 2.2).

2.1 Image Colorization

Image colorization methods can be divided into methods that require human interaction and fully automated methods. Furthermore, we categorize into the following colorization methods:

1. Scribble-based colorization: where the user has to make colorful scribbles on the grayscale images to specify the target color for regions,

2. Example-based methods: where the user has to provide a reference image from which the colors are transferred to the target grayscale images, and

3. Fully-automatic colorization methods: here no human interaction is needed. They are mostly based on Convolutional Neural Networks (CNNs). A huge database of all kinds of objects are used to train these networks. Input grayscale images are then colorized with the help of these pre-trained *CNNs*.

In the category of using scribbles to colorize grayscale images, Levin et al. [22] proposed an optimization based approach to propagate the colors onto the entire image. Huang et al. [14] improved this idea by adding an adaptive edge detection to avoid bleeding at object boundaries. The work of Yatziv and Sapiro [42] proposed a computationally more efficient method, which is based on chrominance blending. An example for scribble-based image colorization can be seen in Figure 2.1. The grayscale image in Fig. 2.1a is provided with scribbles as in Fig. 2.1b. The final colorized image can be seen in Fig. 2.1c.

(a) **(b)** **(c)**

Figure 2.1: Scribble-based image colorization. Given a grayscale image (a) the user has to make scribbles to specify colors (b). Based on the scribbles the algorithm achieves a final colorization (c). Images are taken from [22].

In the field of example-based image colorization Welsh et al. [40] proposed a colorization method based on matching rectangular swatches between the grayscale and the reference image. Based on this work, many new, improved versions were introduced [9], [17], [20]. Another version, the work of Liu et al. [24], uses multiple reference images to colorize images. Different color transfer algorithms have been developed, that use Optimal Transport (OT), such as [8] and [31]. This approaches can be adapted to the colorization of grayscale images. Figure 2.2 illustrates a sample for example-based image colorization. To colorize the grayscale image in Fig. 2.2a a colorful reference image as in Fig. 2.2b is needed. The final colorization can be seen in Fig. 2.2c.

Fully automatic methods using *CNNs* pre-trained on a huge image database, are *e.g.* the work of Cheng et al. [6], Iizuka et al. [15] or Zhang et al. [44]. Zhang et al. [45] proposed a method, that combines sribble-based and fully automatic methods. To achieve a colorization of a grayscale image, the user has to make sparse color splashes. A pre-trained *CNN* then colors the whole image based on the user input. In Figure 2.3 an example for a fully-automatic colorization method is shown. The grayscale image in Fig. 2.3a is colorized without any additional input (Fig. 2.3b).

(a) (b) (c)

Figure 2.2: Example-based image colorization. Given a grayscale image (a) the user has to provide a colorize reference image (b) to specify colors. Based on the reference image the algorithm transfers color to the grayscale image, to find a final colorization (c). Images are taken from [9].

(a) (b)

Figure 2.3: Fully-automatic image colorization. Given a grayscale image (a), no additional input is required to achieve a final colorization (b). Images are taken from [44].

2.2 Video Colorization

The colorization of videos can be seen as an extension of image colorization. Many image colorization approaches can be adapted for videos, e.g. make colorful scribbles on many different frames and incorporate the temporal neighborhood to generate a colored video. Another technique is to colorize each image individually.

Aside from these approaches other methods have been developed. In their work Vondrick et al. [39] introduced a method to colorize videos by the use of a colorful reference image. They use this approach for self-supervised tracking. Xia et al. [41] use a set of color reference frames from a sequence, to transfer color to grayscale frames based on the optical flow calculated between them. Jampani et al. [18] proposed video propagation networks. These networks can be used to propagate the colors of a reference image to subsequent, grayscale frames. Video propagation networks consist of a temporal, bilateral network for dense and video adaptive filtering, as well as a spatial network that refines features. Schaub et al. [33] proposed a video colorization scheme that utilizes a local and global

method to consistently propagate color throughout a grayscale sequence. In their work Iizuka et al. [16] proposed a colorization and restoration approach for old video sequences, that removes impurities and colors the sequence based on a set of reference images.

3 Theoretical Background

Contents

In this section we discuss the relevant theoretical prerequisites. We do not consider the processes of image formation and acquisition, but rather focus on the colorization of already pre-processed digital grayscale images. The following descriptions are partially build on the work of Burger and Burge [3] and Szeliski [37].

3.1 Notation and Conventions

Before we start with the theoretical background, we introduce the mathematical notations used for images and image sequences throughout the thesis. We define images with binary pixel values in the space

$$\mathbb{B} = \{0, 1\}^{M \times N}, \tag{3.1}$$

e.g. $b \in \mathbb{B}$ is a binary image of size $M \times N$. A grayscale image, *e.g.* g, is defined in the space

$$\mathbb{G} = [0, 1]^{M \times N}, \tag{3.2}$$

therefore $g \in \mathbb{G}$. We index binary respectively grayscale image at the position (i, j) via b_{ij} *resp.* g_{ij}. For color images we need additional channels, so we expand the space,

$$\mathbb{C} = \mathbb{G}^3. \tag{3.3}$$

A color image $c \in \mathbb{C}$ is in the space $[0, 1]^{3\times M\times N}$. We have three color channels and we index color channels first, c_{kij} is the k-th channel at the location (i, j). c_k gives the k-th color channel of the image c such that $c_k \in \mathbb{G}$. A sequence of grayscale images with length T *e.g.* $G \in \mathbb{G}^T$ is defined by

$$G = \{g_t\}_{t=1}^{T}, \tag{3.4}$$

where a grayscale sequence is then indexed via G_{tij}, which gives the t-th frame at location (i, j). A sequence of color images, *e.g.* $C \in \mathbb{C}^T$ is defined by

$$C = \{c_t\}_{t=1}^{T}, \tag{3.5}$$

where C_{tkij} corresponds to the t-th element in the sequence, the k-th channel at location (i, j). C_t denotes the t-th element of C.

3.2 Digital Images

A digital image on our computer, smartphone, *etc.*, is a regular arranged, rectangular grid made up of uniform pixels. A pixel, picture element, is the smallest entity of a digital image. Depending on the information the pixels of an image hold, we distinguish between binary or black-and-white images $b \in \mathbb{B}$, monochromatic or grayscale images $g \in \mathbb{G}$ and color images $c \in \mathbb{C}$. Each of the different images has a spatial size of $M \times N$, meaning an image spans a two-dimensional, regular rectangle with M rows and N columns. The simplest form of images are black-and-white images b, or binary images. Here, each pixel of the image is just able to hold 1 bit of information, *resp.* two different numeric intensity values. This is used to distinguish between white (1) and black (0). Figure 3.1 visualizes a black-and-white image (Fig. 3.1a) and a zoom (Fig. 3.1b), where individual pixels can be seen.

3.3 Grayscale Images

When we use more bits of information for the pixels, we can represent *e.g.* different shades of gray. Grayscale images g use typically 8 bits of information, which leads to 256 different intensity values. In this thesis we define a grayscal images $g \in \mathbb{G}$. This implies that we normalize the grayscale images to get values in the interval $[0, 1]$, in order to have floating point images. A grayscale image consists of a single channel, which represents the brightness or intensity of the image. In Figure 3.2 we can see a grayscale variant of the images in Figure 3.1.

(a) (b)

Figure 3.1: Black-and-white image. (a) shows a blackswan in binary form. (b) gives a zoomed-in look of the black-and-white image, where single pixels can be seen.

(a) (b)

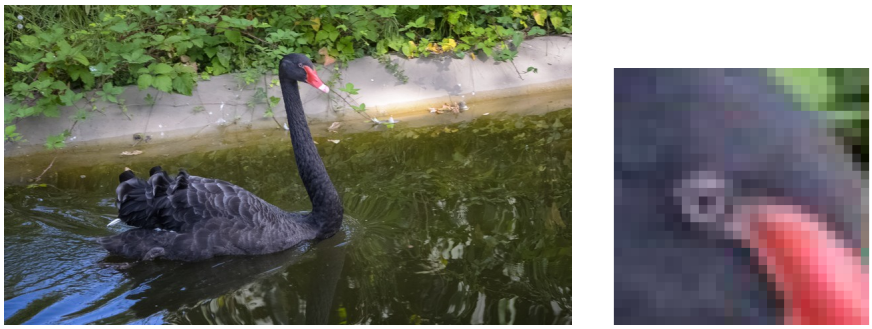
Figure 3.2: Grayscale image. (a) shows a blackswan in grayscale. (b) gives a zoomed-in look of the grayscale image, where single pixels can be seen.

3.4 Color Images

The next extension for images is to use multiple channels. Color images, for example, use three or even more channels. We define a color image c with three channels, each consisting of 8 bits of information, $c \in \mathbb{C}$. Again, all channels are normalized in the range of $[0, 1]$. Each of the channels now represents a color, a color difference, *etc.* The definition depends on the used color space (Section 3.5). Figure 3.3 shows the color variant of the images in Figure 3.1 and Figure 3.2.

3.5 Color Spaces

As already mentioned there are various color spaces. Each defines the individual channels of color images in a different way. Some color spaces define the separate channels based on the human perception, others use a definition based on display hardware. It depends on



(a) **(b)**

Figure 3.3: Color image. (a) shows a blackswan in color. (b) gives a zoomed-in look of the color image, where single pixels can be seen.

the scientific field to decide which color space to use. In Sections 3.5.1 - 3.5.4, we describe some of the the most commonly used color spaces and specify the conversion rules between them. We use the superscript for a color image $c \in \mathbb{C}$ to indicate the various color spaces. c^{RGB} means that the color image is in RGB color space.

3.5.1 RGB

Probably the most known color space is RGB. RGB uses the colors red, green and blue (R, G, B) to encode all other colors. This color space is used for computer displays. Displays utilize red, green and blue Light-Emitting Diodes (LEDs) to produce a huge variety of colors. Also many image processing programs and Application Programming Interfaces (APIs) use RGB as their standard definition of color images. RGB is an additive color scheme. We consider a single pixel, when all color channels have no intensity, *i.e.* $(0, 0, 0)$, the resulting color is black. If the red channel has full intensity but the others none $(1, 0, 0)$, the color is red, whereas $(1, 1, 1)$ defines white. Figure 3.4 illustrates the color image from Fig. 3.3a in its individual channels in the RGB color space. One can see, that the red beak of the blackswan has high intensity values (brighter) in the R-channel (3.4a), whereas the intensity in G- and B-channel is low (darker).

Now follows the conversion from the RGB color space to grayscale images and vice versa. Since we convert from a color space with three channels (RGB) to grayscale with one channel, we loose information. Therefore, the inverse conversion does not result in the original image (see Colorization Problem, Section 3.6). Given a color image in RGB color space $c^{\text{RGB}} \in \mathbb{C}$, and the R-channel with index $k = 1$, G-channel with $k = 2$ and B-channel with index $k = 3$, the conversion to a grayscale image $g \in \mathbb{G}$ for the location

(a) **(b)** **(c)**

Figure 3.4: RGB channels. Illustration of the color image (Fig. 3.3a) in the individual RGB-channels. (a) is the R-channel, (b) the G-channel and (c) is the B-channel.

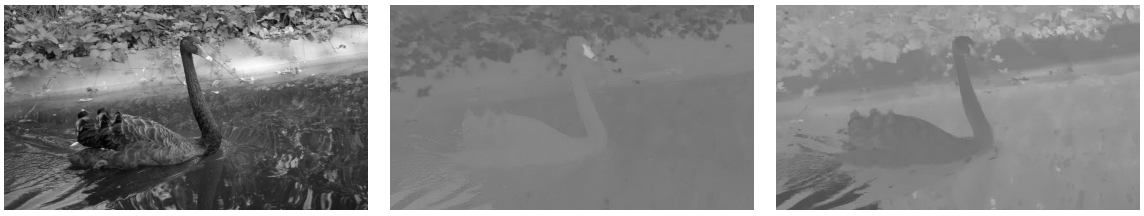
(i, j) is the following:

RGB to Grayscale:
$$g_{ij} = 0.299 c_{1ij}^{\mathrm{RGB}} + 0.587 c_{2ij}^{\mathrm{RGB}} + 0.114 c_{3ij}^{\mathrm{RGB}} \tag{3.6}$$

Grayscale to RGB:
$$c_{1ij}^{\mathrm{RGB}} = c_{2ij}^{\mathrm{RGB}} = c_{3ij}^{\mathrm{RGB}} = g_{ij}. \tag{3.7}$$

3.5.2 LAB

The LAB color space is more linear with respect to the human color perception. This means that a change in a color value should result in about the same change in the visual presentation. The LAB color space is widely used in professional photographic applications. The L-channel specifies the luminosity, the A-channel measures the color hue and saturation along the green-red axes and the B-channel specifies the color hue and saturation along the blue-yellow axes. The channels contain relative values and refer to a specified white point reference. Figure 3.5 visualizes the color image from Fig. 3.3a in the LAB color space.



(a) **(b)** **(c)**

Figure 3.5: LAB channels. Illustration of the color image (Fig. 3.3a) in the individual LAB-channels. (a) is the L-channel, (b) the A-channel and (c) is the B-channel.

The conversion from the RGB color space to the LAB color space requires a pre-conversion into the XYZ color space. The XYZ color space is based on imaginary primary colors, which are chosen such that all colors can be defined as a positive summation of

them. The conversion of RGB to LAB is stated below. The inverse process, the conversion from LAB to RGB, can be found in Burger et al. [3]. Given a color image in RGB color space $c^{\mathrm{RGB}} \in \mathbb{C}$, we define the intermediate conversion to $c^{\mathrm{XYZ}} \in \mathbb{C}$ and to $c^{\mathrm{LAB}} \in \mathbb{C}$. For all color spaces the first color channel stands for the first capitalized letter of the color space, same as in Section 3.5.1. The conversions at location (i, j) are the following:

RGB to XYZ:

$$\begin{aligned}
\begin{bmatrix} \hat{c}^{\mathrm{XYZ}}_{1ij} \\ c^{\mathrm{XYZ}}_{2ij} \\ \hat{c}^{\mathrm{XYZ}}_{3ij} \end{bmatrix} &= \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \cdot \begin{bmatrix} c^{\mathrm{RGB}}_{1ij} \\ c^{\mathrm{RGB}}_{2ij} \\ c^{\mathrm{RGB}}_{3ij} \end{bmatrix} \\
c^{\mathrm{XYZ}}_{1ij} &= \frac{\hat{c}^{\mathrm{XYZ}}_{1ij}}{X_n}, \text{where } X_n = 0.950456 \\
c^{\mathrm{XYZ}}_{3ij} &= \frac{\hat{c}^{\mathrm{XYZ}}_{3ij}}{Z_n}, \text{where } Z_n = 1.088754
\end{aligned} \tag{3.8}$$

XYZ to LAB:

$$\begin{aligned}
\hat{c}^{\mathrm{LAB}}_{1ij} &= \begin{cases} 116 \cdot {c^{\mathrm{XYZ}}_{2ij}}^{\frac{1}{3}} - 16, & \text{for } c^{\mathrm{XYZ}}_{2ij} > 0.008856 \\ 903.3 \cdot c^{\mathrm{XYZ}}_{2ij}, & \text{for } c^{\mathrm{XYZ}}_{2ij} \leq 0.008856 \end{cases} \\
c^{\mathrm{LAB}}_{1ij} &= \frac{\hat{c}^{\mathrm{LAB}}_{1ij}}{100} \\
c^{\mathrm{LAB}}_{2ij} &= \frac{500 \cdot (f(c^{\mathrm{XYZ}}_{1ij}) - f(c^{\mathrm{XYZ}}_{2ij})) + 128}{255} \\
c^{\mathrm{LAB}}_{3ij} &= \frac{200 \cdot (f(c^{\mathrm{XYZ}}_{2ij}) - f(c^{\mathrm{XYZ}}_{3ij})) + 128}{255} \\
f(t) &= \begin{cases} t^{\frac{1}{3}}, & \text{for } > 0.008856 \\ 7.787 \cdot t + \frac{16}{116}, & \text{for } t \leq 0.008856. \end{cases}
\end{aligned} \tag{3.9}$$

This outputs $0 \leq c^{\mathrm{LAB}}_{1ij} \leq 1$, $0 \leq c^{\mathrm{LAB}}_{2ij} \leq 1$ and $0 \leq 1$. Hence, all channels again fit within the interval $[0, 1]$.

3.5.3 HSV

The initials in HSV stand for hue, saturation and value. This color space has a more natural aspect regarding colors. Colors are not produced by addition or subtraction, but in terms of hue and saturation. Hue is measured in an angular dimension. In Figure 3.6 the color image (Fig. 3.3a) is visualized in the HSV color space.

Now follows the conversion from RGB to HSV. The conversion from HSV to RGB, can be found in Burger et al. [3]. Given a color image in RGB color space $C^{\mathrm{RGB}} \in \mathbb{C}$ we convert to a color image in HSV, $C^{\mathrm{HSV}} \in \mathbb{C}$ at the location (i, j) by

(a) (b) (c)

Figure 3.6: HSV channels. Illustration of the color image (Fig. 3.3a) in the individual HSV-channels. (a) is the H-channel, (b) the S-channel and (c) is the V-channel.

RGB to HSV:

$$
\begin{aligned}
c_{3ij}^{\mathrm{HSV}} &= \max(c_{1ij}^{\mathrm{RGB}}, c_{2ij}^{\mathrm{RGB}}, c_{3ij}^{\mathrm{RGB}}) \\
c_{2ij}^{\mathrm{HSV}} &= \begin{cases} c_{3ij}^{\mathrm{HSV}} - \min(c_{1ij}^{\mathrm{RGB}}, c_{2ij}^{\mathrm{RGB}}, c_{3ij}^{\mathrm{RGB}}), & \text{if } c_{3ij}^{\mathrm{HSV}} \neq 0 \\ c_{3ij}^{\mathrm{HSV}}, & \text{else} \end{cases} \\
\hat{c}_{1ij}^{\mathrm{HSV}} &= \begin{cases} \frac{60(c_{2ij}^{\mathrm{RGB}} - c_{3ij}^{\mathrm{RGB}})}{c_{3ij}^{\mathrm{HSV}} - \min(c_{1ij}^{\mathrm{RGB}}, c_{2ij}^{\mathrm{RGB}}, c_{3ij}^{\mathrm{RGB}})}, & \text{if } c_{3ij}^{\mathrm{HSV}} = c_{1ij}^{\mathrm{RGB}} \\ 120 + \frac{60(c_{3ij}^{\mathrm{RGB}} - c_{1ij}^{\mathrm{RGB}})}{c_{3ij}^{\mathrm{HSV}} - \min(c_{1ij}^{\mathrm{RGB}}, c_{2ij}^{\mathrm{RGB}}, c_{3ij}^{\mathrm{RGB}})}, & \text{if } c_{3ij}^{\mathrm{HSV}} = c_{2ij}^{\mathrm{RGB}}, \\ 240 + \frac{60(c_{1ij}^{\mathrm{RGB}} - c_{2ij}^{\mathrm{RGB}})}{c_{3ij}^{\mathrm{HSV}} - \min(c_{1ij}^{\mathrm{RGB}}, c_{2ij}^{\mathrm{RGB}}, c_{3ij}^{\mathrm{RGB}})}, & \text{if } c_{3ij}^{\mathrm{HSV}} = c_{3ij}^{\mathrm{RGB}} \end{cases} \\
c_{1ij}^{\mathrm{HSV}} &= \begin{cases} \frac{\hat{c}_{1ij}^{\mathrm{HSV}} + 360}{360}, & \text{if } \hat{c}_{1ij}^{\mathrm{HSV}} < 0 \\ \frac{\hat{c}_{1ij}^{\mathrm{HSV}}}{360}, & \text{else.} \end{cases}
\end{aligned}
\tag{3.10}
$$

We achieve final output ranges: $0 \leq c_{3ij}^{\mathrm{HSV}} \leq 1$, $0 \leq c_{2ij}^{\mathrm{HSV}} \leq 1$, $0 \leq c_{1ij}^{\mathrm{HSV}} \leq 1$. All three color channels fit in the interval $[0, 1]$.

3.5.4 YCbCr

YCbCr is a color space that is used for TV applications. YCbCr can be seen as a newer version of the YUV color space, which was originally developed for television services. The idea of the YCbCr color space is to have a separate brightness channel Y and two independent color channels Cb and Cr, which carry the blue and red color difference signals. All channels of the YCbCr JPEG standard, which we are using, fit in the interval $[0, 1]$. Since this color space is used for JPEG files, it carries a high importance in digital image processing. Its layout conveniently allows to apply image compression without being noticed by the human visual system. Figure 3.7 visualizes the color image from Fig. 3.3a in the YCbCr color space.

The conversion between RGB and YCbCr and vice versa is shown as follows. Given a color image in RGB color space $C^{\mathrm{RGB}} \in \mathbb{C}$ we convert to a color image in YCbCr

(a) (b) (c)

Figure 3.7: YCbCr channels. Illustration of the color image (Fig. 3.3a) in the individual YCbCr-channels. (a) is the Y-channel, (b) the Cb-channel and (c) is the Cr-channel.

$C^{\text{YCbCr}} \in \mathbb{C}$ at the location (i, j). It can be seen in Equation (3.6) that the formula of the Y-channel is exactly the same as when converting from RGB to grayscale, see Equation (3.11).

$$\text{RGB to YCbCr:} \quad \begin{aligned} c_{1ij}^{\text{YCbCr}} &= 0.299 c_{1ij}^{\text{RGB}} + 0.587 c_{2ij}^{\text{RGB}} + 0.114 c_{3ij}^{\text{RGB}} \\ c_{3ij}^{\text{YCbCr}} &= (c_{1ij}^{\text{RGB}} - c_{1ij}^{\text{YCbCr}}) \cdot 0.713 + 0.5 \\ c_{2ij}^{\text{YCbCr}} &= (c_{3ij}^{\text{RGB}} - c_{1ij}^{\text{YCbCr}}) \cdot 0.564 + 0.5 \end{aligned} \tag{3.11}$$

$$\text{YCbCr to RGB:} \quad \begin{aligned} c_{1ij}^{\text{RGB}} &= c_{1ij}^{\text{YCbCr}} + 1.402 \cdot (c_{3ij}^{\text{YCbCr}} - 0.5) \\ c_{2ij}^{\text{RGB}} &= c_{1ij}^{\text{RGB}} - 0.714 \cdot (c_{3ij}^{\text{YCbCr}} - 0.5) - 0.344 \cdot (c_{2ij}^{\text{YCbCr}} - 0.5) \\ c_{3ij}^{\text{RGB}} &= c_{1ij}^{\text{YCbCr}} + 1.772 \cdot (c_{2ij}^{\text{YCbCr}} - 0.5) \end{aligned} \tag{3.12}$$

3.6 Colorization Problem

As already briefly mentioned in Section 3.5.1, there is an information loss when converting from RGB (or any other multi channel color space) to grayscale. The reason for this is that we convert from a space with three or more channels to a space with a single channel. Therefore, the inverse conversion from grayscale to color does not result in the original image. This problem is now defined as the colorization problem. The colorization problem is an inverse and ill-posed problem. Most inverse problems in practice are ill-posed. An ill-posed problem is a problem that doesn't meet one of the three Hadamard criteria [10] for being well-posed. These criteria are:

1. existence of a solution,
2. having an unique solution, and
3. having a solution that depends continuously on the parameters or input data.

For the colorization problem, we can see that there is no unique solution for recoloring grayscale images. It can be seen in Figure 3.8, that two different color images, Fig. 3.8a

and Fig. 3.8b, converted to grayscale, yield the same grayscale image Fig. 3.8c. Without further knowledge (*e.g.* blackswan is black, leaves are green, *etc.*), both color images would be appropriate solutions to the inverse problem of colorizing the grayscale image.

(a) **(b)** **(c)**

Figure 3.8: Colorization problem. When converting two color images, (a) and (b), into grayscale, they achieve the same gray image (c). Therefore, the inverse problem of finding a colorization of (3.8c) is ill-posed, since there is no unique solution.

We can see from the example in Figure 3.8 that we need additional information of the image or scene to find a well-suited colorization. For image and video colorization, these information can come from reference images, colored scribbles or the usage of trained Convolutional Neural Networks (CNNs), that recognize objects and have a color memorized for these objects.

In the following chapters we propose methods to colorize grayscale images and sequences based on reference images.

4

Colorization using Optimal Transport (OT)

Contents

As a starting point, we decided to begin with the colorization of grayscale images. We use a similar looking color reference image to colorize a grayscale image, so we work with an example-based colorization method. The used reference image may need to contain all objects that are in the grayscale image to perform a visual appealing colorization. We decided to follow the approach of Gupta et al. [9] and incorporate the concept of *OT* into this baseline.

The concept that Gupta et al. [9] are using in their work, is to generate a set of features for each pixel of the grayscale image and a set of features for each pixel of the reference color image. They use superpixels to achieve a semantic correct colorization. For each superpixel they calculate a set of mean-features for all pixels located in the superpixel. Given those two sets, a matching between each superpixel of the first set with the superpixels of the second set is computed. Based on this matching, a color transfer between the images, and therefore a recoloring of the grayscale image, is performed.

Given a target grayscale image $g^t \in \mathbb{G}$ and a colored source frame $c^s \in \mathbb{C}$, we want to find a colored version c^t of g^t. Here we use the superscript to differentiate between target, g^t and c^t, and source c^s. We write the colorization as a mapping $\mathcal{T}_{OT}$

$$\begin{aligned} \mathcal{T}_{OT} : (\mathbb{G}, \mathbb{C}) &\to \mathbb{C}, \\ (g^t, c^s) &\mapsto c^t. \end{aligned} \tag{4.1}$$

Figure 4.1 visualizes the concept we are following. After a pre-processing step we extract and match features to achieve a colorization.

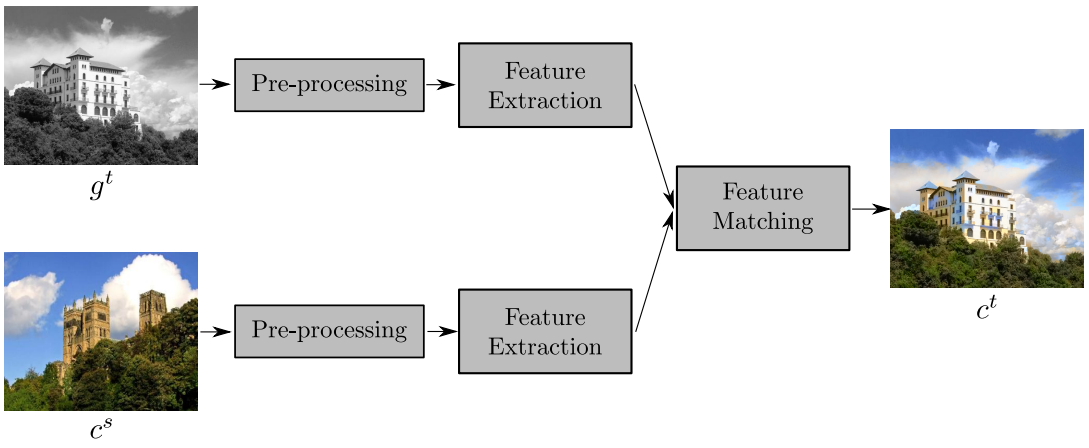


Figure 4.1: Concept of image colorization using a color reference image. Given a grayscale image g^t and a color reference image c^s, we extract for each image a feature set. We then match both feature sets and perform a colorization c^t of g^t.

4.1 Approach

The following sections give an explanation of each individual block visualized in Figure 4.1.

4.1.1 Pre-processing

As a first step we convert both images g^t and c^s into the LAB color space (see Section 3.5.2). First we convert the grayscale image g^t to the LAB color space $\hat{c}^t \in \mathbb{C}$. We do not write the superscript LAB, since we define the color images to be in this space. The L-channel of c^s is comparable to the intensity of the grayscale image g^t, *resp.* the L-channel of $\hat{c}^t$. From now on we work with the L-channel of both images. We perform a linear mapping of the color image c^s in order to minimize differences in the L-channel via

$$c^s_{1ij} = \frac{\sigma^t}{\sigma^s}(c^s_{1ij} - \mu^s) + \mu^t, \tag{4.2}$$

with c^s_{1ij} being the L-channel at position (i, j). μ^s is the mean and σ^s the standard deviation of the L-channel of c^s. μ^t is the mean and σ^t the standard deviation of the L-channel of $\hat{c}^t$. For the L-channel of an image $c \in \mathbb{C}$, μ and σ are given by

$$\mu = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} c_{1ij}, \tag{4.3}$$

and

$$\sigma = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}(c_{1ij}-\mu)^2}{MN}}. \tag{4.4}$$

This linear mapping was introduced by Hertzman et al. [12]. It is a luminance transformation that brings the histograms of $\hat{c}^t$ and c^s into correspondence, without having the side affects of a non-smooth mapping.

4.1.2 Statistical and Textural Features

We use different kinds of features for the matching. For statistical features we use the luminance (L-channel), mean deviation in a local neighborhood, as well as the standard deviation in a local neighborhood around a pixel. We additionally use simple textural features, as introduced by Law [21]. We calculate all features for the L-channel of c^s and $\hat{c}^t$.

The mean deviation for a pixel is calculated within an $n \times n$ window around a pixel using the L-channel. We introduce a kernel $\mathbf{K}_{n\times n} \in \mathbb{R}_{+}^{n\times n}$ to calculate the mean and standard deviation:

$$\mathbf{K}_{n\times n} = \frac{1}{n^2}\begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix}. \tag{4.5}$$

The mean deviation $I_{\mu_{n\times n}} \in \mathbb{R}^{M\times N}$ for all pixels in the L-channel of an image $c \in \mathbb{C}$ is calculated via:

$$I_{\mu_{n\times n}} = |c_1 - (c_1 * \mathbf{K}_{n\times n})|. \tag{4.6}$$

For the standard deviation $I_{\sigma_{n\times n}} \in \mathbb{R}^{M\times N}$ we again use a $n \times n$ window around a pixel. The calculation for all pixels of the L-channel in c gives:

$$I_{\sigma_{n\times n}} = \sqrt{|({c_1}^2 * \mathbf{K}_{n\times n}) - (c_1 * \mathbf{K}_{n\times n})^2|}. \tag{4.7}$$

For the final calculation we use a window of size 5×5 pixels.

As already mentioned, the textural features are based on the work of Law [21]. They give different responses based on the "structure" of the image. There are different filters to detect edges (E), spots (S), levels (L) and ripples (R) of size 5:

$$E_5 = \begin{bmatrix} -1 & -2 & 0 & 2 & 1 \end{bmatrix} \tag{4.8}$$

$$S_5 = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \end{bmatrix} \tag{4.9}$$

$$L_5 = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{4.10}$$

$$R_5 = \begin{bmatrix} 1 & 4 & 6 & -4 & 1 \end{bmatrix}. \tag{4.11}$$

We apply a combination of these filters in a horizontal and vertical orientation to construct a final filter of size 5×5. To get the corresponding filter responses, we convolve the L-channel of our images with the combination of filters from (4.8) - (4.11), *e.g.* for a color image $c \in \mathbb{C}$ we yield $c_1 * {E_5}^T S_5$, $c_1 * {L_5}^T S_5$, $c_1 * {E_5}^T L_5$ and $c_1 * {R_5}^T R_5$. We calculate these responses for the images $\hat{c}^t$ and c^s.

4.1.3 Feature Descriptors

For even more features we use feature descriptors. There are many different types, *e.g.* Scale-Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF), Oxford-Visual Geometry Group (VGG), Gabor filters, *etc.* We decided to use *SURF* and Gabor filters. These feature descriptors *resp.* filter banks are shortly described in the following sections. We calculate the descriptors and responses for each pixel of the image, and not just for salient points.

4.1.3.1 SURF

SURF was introduced by Bay et al. [2]. It represents a scale- and rotation-invariant feature descriptor and its computation is fast compared to other descriptors. *SURF* is based on the same principles as *SIFT* but simpler in terms of complexity. The definition of *SURF* for an interest point is the following. The first step is the calculation of an orientation in the circular neighborhood around the interest point. Then, a square region around the interest point and oriented the same as the circular neighborhood is constructed. This square region is described by extracting smaller sub-squares. For each sub-region a weighted Gaussian wavelet response is calculated. These sub-region-wavelet responses are used as the descriptor for the interest point. We calculate *SURF* for the L-channel of the images $\hat{c}^t$ and c^s.

4.1.3.2 Gabor Filters

We additionally calculate the response of the L-channel of the images $\hat{c}^t$ and c^s to Gabor filters ([25] and [27]). The Gabor filter has a real and an imaginary component and can be defined in the complex space. We calculate responses with eight orientations, varying in increments of $i\frac{\pi}{8}$, with $i \in \{0, 1, \ldots, 6, 7\}$. Additionally, we use five exponential scales $\exp(i \cdot \pi)$, with $i \in \{1, 2, 3, 4, 5\}$.

4.1.4 Superpixels

We use Simple Linear Iterative Clustering (SLIC)-superpixels as proposed by Achanta et al. [1] to incorporate semantic information into our colorization procedure. Based on the segmentation given by the superpixels, we combine the features of pixels that belong to the same superpixel and calculate mean-features for that corresponding superpixel. We calculate this mean-feature superpixel for $\hat{c}^t$ and c^s based on the individual L-channels.

To increase the number of features for a mean-feature superpixel, we calculate the mean luminance and the mean statistical features for its neighboring superpixels. This again helps to add semantic information.

4.1.5 Feature Matching via Optimal Transport

Now that we have a set of features for the color reference image c^s and the grayscale image g^t via $\hat{c}^t$, we are able to generate a matching between each individual superpixel of the grayscale image and the superpixels of the color image. We can say that we want to match the center of each superpixel from $\hat{c}_1^t$ with a center of a superpixel of c_1^s.

We decided to use a matching approach based on *OT*. The problem of *OT* has originally been proposed by Monge [26] in 1781 and deals with the optimal transportation and allocation of resources. Later, Kantorovich [19] in 1942 proceeded with the initial concept and refined Monge's ideas.

We work with the discrete version of *OT*. *OT* deals with the problem of finding an optimal transport plan $\mathbf{\Pi} \in \mathbb{R}_+^{N\times N}$ from a source set $\mathcal{X} = \{X_i\}_{i=1}^N$ to a target set $\mathcal{Y} = \{Y_i\}_{i=1}^N$. We limit both sets to have N elements. The i-th row of the transport plan, Π_i, contains for all j the probability that X_i maps to Y_j. To find the best assignment we want to minimize

$$\min_{\mathbf{\Pi}\in\mathbb{R}_+^{N\times N}} \langle \mathbf{C}, \mathbf{\Pi}\rangle_F \quad \text{s.t.} \quad \sum_{i=0}^{N} \Pi_{ij} = p_j, \quad \sum_{j=0}^{N} \Pi_{ij} = q_i, \tag{4.12}$$

which is known as the Kantorovich relaxation of the optimal transport problem. In Equation (4.12), $\mathbf{C} \in \mathbb{R}_+^{N\times N}$ is a cost matrix, where the cost between two samples X_i and Y_j is calculated via $C_{ij} = ||X_i - Y_j||^2$, and $\mathbf{p}, \mathbf{q} \in \mathbb{R}^N$ are the marginal probability distributions of $\mathbf{\Pi}$. It holds that $\sum_{i=0}^N p_i = 1$ and $\sum_{i=0}^N q_i = 1$. $\langle\cdot,\cdot\rangle_F$ is the Frobenius dot product and it is defined as

$$\langle \mathbf{C}, \mathbf{\Pi}\rangle_F = \sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij}\Pi_{ij}. \tag{4.13}$$

The solution of Equation (4.12) yields the bijective assignment

$$\mathbf{\Pi}: \quad \mathcal{X} \to \mathcal{Y}, \tag{4.14}$$

$$X_i \mapsto \sum_{j=1}^{N} \Pi_{ij} Y_j. \tag{4.15}$$

This means that a target point is a linear combination of source points. There are different algorithms that solve this problem. In the following sections we present such methods, like the Primal-Dual algorithm (Section 4.1.5.1) as well as the Sinkhorn-Knopp algorithm (Section 4.1.5.2).

4.1.5.1 Primal-Dual Algorithm

With the Primal-Dual algorithm we can solve problems of the form

$$\min_{\mathbf{x}} f(\mathbf{K}\mathbf{x}) + g(\mathbf{x}), \tag{4.16}$$

where f, g are convex lower-semicontinuous (l.s.c.) and 'simple' functions, and $\mathbf{K} \in \mathbb{R}^{M \times N}$ is a bounded linear operator. We can rewrite the primal problem Equation (4.16) to a saddle-point problem using the Fenchel conjugate function

$$\min_{\mathbf{x}} \max_{\mathbf{y}} \langle \mathbf{K}\mathbf{x}, \mathbf{y} \rangle - f^*(\mathbf{y}) + g(\mathbf{x}). \tag{4.17}$$

Problems of the form Equation (4.17) can be solved with the Primal-Dual Hybrid Gradient (PDHG) algorithm, as proposed by Chambolle and Pock [4],

$$\begin{cases} x^{k+1} = \text{prox}_{\tau g}(x^k - \tau K^T y^k) \\ y^{k+1} = \text{prox}_{\sigma f^*}(y^k + \sigma K(2x^{k+1} - x^k)) \end{cases}, \tag{4.18}$$

given an initial pair of primal and dual points (x^0, y^0) and step sizes $\tau, \sigma > 0$. The requirement $\sigma\tau||\mathbf{K}||^2 < 1$ guarantees convergence of the algorithm. Now we would like to re-write the optimization problem in Equation (4.12) to a saddle-point problem, as in Equation (4.17).

$$\begin{aligned} &\min_{\mathbf{\Pi} \in \mathbb{R}_+^{N \times N}} \langle \mathbf{C}, \mathbf{\Pi} \rangle_F \quad \text{s.t.} \quad \sum_{i=1}^{N} \Pi_{ij} = p_j, \quad \sum_{j=1}^{N} \Pi_{ij} = q_i \\ &\min_{\mathbf{\Pi} \in \mathbb{R}_+^{N \times N}} \langle \mathbf{C}, \mathbf{\Pi} \rangle_F \quad \text{s.t.} \quad \sum_{i=1}^{N} \Pi_{ij} = p_j, \quad \sum_{j=1}^{N} \Pi_{ij} = q_i, \quad \forall ij : \Pi_{ij} \geq 0 \\ &\min_{\mathbf{\Pi} \in \mathbb{R}_+^{N \times N}} \langle \mathbf{C}, \mathbf{\Pi} \rangle_F \quad \text{s.t.} \quad \mathbf{\Pi}\mathbf{1} = \mathbf{p}, \quad \mathbf{\Pi}^T\mathbf{1} = \mathbf{q}, \quad \forall ij : \Pi_{ij} \geq 0 \end{aligned} \tag{4.19}$$

with $\mathbf{1}$ as the one vector with size N. Here, we introduce the following variables $\mathbf{x} \in \mathbb{R}_+^{N^2 \times 1}$ which is $\mathbf{\Pi}$ flattened, and $\mathbf{c} \in \mathbb{R}_+^{N^2 \times 1}$ which is $\mathbf{C}$ flattened. This then leads to

$$\begin{aligned} &\min_{\mathbf{\Pi} \in \mathbb{R}_+^{N \times N}} \langle \mathbf{C}, \mathbf{\Pi} \rangle_F \quad \text{s.t.} \quad \mathbf{\Pi}\mathbf{1} = \mathbf{p}, \quad \mathbf{\Pi}^T\mathbf{1} = \mathbf{q}, \quad \forall ij : \Pi_{ij} \geq 0 \\ &\min_{\mathbf{x} \in \mathbb{R}_+^{N^2 \times 1}} \mathbf{c}^T\mathbf{x} \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{p} \quad \mathbf{B}\mathbf{x} = \mathbf{q}, \quad \forall i : x_i \geq 0 \end{aligned}, \tag{4.20}$$

with $\mathbf{A} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & & \ddots & & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{1} \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} \text{diag}(\mathbf{1}) & \text{diag}(\mathbf{1}) & \cdots & \text{diag}(\mathbf{1}) \end{bmatrix}$,

with $\mathbf{A}, \mathbf{B} \in \mathbb{R}_+^{N \times N^2}$. We now introduce Lagrange multipliers $\lambda, \mu \in \mathbb{R}_+^N$ to express the constraints.

$$\begin{aligned} &\min_{\mathbf{x} \in \mathbb{R}_+^{N^2 \times 1}} \mathbf{c}^T \mathbf{x} \quad \text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{p} \quad \mathbf{B}\mathbf{x} = \mathbf{q}, \quad \forall i : x_i \geq 0 \\ &\max_{\lambda, \mu} \min_{\mathbf{x} \in \mathbb{R}_+^{N^2 \times 1}} \mathbf{c}^T \mathbf{x} + \langle \mathbf{A}\mathbf{x} - \mathbf{p}, \lambda \rangle + \langle \mathbf{B}\mathbf{x} - \mathbf{q}, \mu \rangle + \delta_{\geq 0}(\mathbf{x}), \end{aligned} \tag{4.21}$$

with

$$\delta_{\geq 0}(\mathbf{x}) \text{ element-wise as } \delta_{\geq 0}(x_i) = \begin{cases} 0, & \text{if } x_i \geq 0 \\ \infty, & \text{else.} \end{cases} \tag{4.22}$$

Now, we define $\mathbf{y} = \begin{pmatrix} \lambda \\ \mu \end{pmatrix}$, $\mathbf{y} \in \mathbb{R}_+^{2N \times 1}$ as the Lagrange multipliers, $\mathbf{K} = \begin{pmatrix} \mathbf{A} \\ \mathbf{B} \end{pmatrix}$, $\mathbf{K} \in \mathbb{R}_+^{2N \times N^2}$ and $\mathbf{w} = \begin{pmatrix} \mathbf{p} \\ \mathbf{q} \end{pmatrix}$, $\mathbf{w} \in \mathbb{R}_+^{2N \times 1}$ as the constraints, that gives us

$$\begin{aligned} &\max_{\lambda, \mu} \min_{\mathbf{x} \in \mathbb{R}_+^{N^2 \times 1}} \mathbf{c}^T \mathbf{x} + \langle \mathbf{A}\mathbf{x} - \mathbf{p}, \lambda \rangle_F + \langle \mathbf{B}\mathbf{x} - \mathbf{q}, \mu \rangle + \delta_{\geq 0}(\mathbf{x}) \\ &\max_{\mathbf{y} \in \mathbb{R}^{2N \times 1}} \min_{\mathbf{x} \in \mathbb{R}_+^{N^2 \times 1}} \mathbf{c}^T \mathbf{x} + \langle \mathbf{K}\mathbf{x} - \mathbf{w}, \mathbf{y} \rangle + \delta_{\geq 0}(\mathbf{x}) \\ &\max_{\mathbf{y} \in \mathbb{R}^{2N \times 1}} \min_{\mathbf{x} \in \mathbb{R}_+^{N^2 \times 1}} \langle \mathbf{K}\mathbf{x}, \mathbf{y} \rangle - \langle \mathbf{w}, \mathbf{y} \rangle + \mathbf{c}^T \mathbf{x} + \delta_{\geq 0}(\mathbf{x}). \end{aligned} \tag{4.23}$$

The comparison of the final result with the definition of the saddle-point problem

$$\begin{aligned} &\max_{\mathbf{y} \in \mathbb{R}^{2N \times 1}} \min_{\mathbf{x} \in \mathbb{R}_+^{N^2 \times 1}} \langle \mathbf{K}\mathbf{x}, \mathbf{y} \rangle - \langle \mathbf{w}, \mathbf{y} \rangle + \mathbf{c}^T \mathbf{x} + \delta_{\geq 0}(\mathbf{x}) \\ &\textit{comp.} \min_{\mathbf{x} \in \mathbb{R}_+^{N^2 \times 1}} \max_{\mathbf{y} \in \mathbb{R}^{2N \times 1}} \langle \mathbf{K}\mathbf{x}, \mathbf{y} \rangle - f^*(\mathbf{y}) + g(\mathbf{x}) \end{aligned} \tag{4.24}$$

leads to:

$$\begin{aligned} f^*(\mathbf{y}) &= \langle \mathbf{w}, \mathbf{y} \rangle \\ g(\mathbf{x}) &= \mathbf{c}^T \mathbf{x} + \delta_{\geq 0}(\mathbf{x}). \end{aligned} \tag{4.25}$$

The steps σ and τ are both set to the value $\frac{1}{||\mathbf{K}||}$. This arises from the Lipschitz constant. The initial primal point x^0 is set to $\forall i : x_i^0 = \frac{1}{N^2}$ and the inital dual point y^0 is

set to $\forall i : y_i^0 = 0$. The proximal mappings of the Primal-Dual algorithm, Equation (4.18), are listed in Appendix B.

4.1.5.2 Sinkhorn-Knopp Algorithm

The Sinkhorn-Knopp algorithm [34] arises from an entropic, regularized optimal transport problem:

$$\min_{\mathbf{\Pi}\in\mathbb{R}_+^{N\times N}} \langle \mathbf{C}, \mathbf{\Pi}\rangle_F + \epsilon \sum_{i=1}^{N}\sum_{j=1}^{N} \Pi_{ij}\log\Pi_{ij} \quad \text{s.t.} \quad \sum_{i=1}^{N}\Pi_{ij} = p_j, \quad \sum_{j=1}^{N}\Pi_{ij} = q_i. \tag{4.26}$$

Iterations of the Sinkhorn-Knopp algorithm are given by

$$\begin{cases} \mathbf{u}^{k+1} = \frac{\mathbf{p}}{\mathbf{M}\mathbf{v}^k} \\ \mathbf{v}^{k+1} = \frac{\mathbf{q}}{\mathbf{M}^T\mathbf{u}^{k+1}}, \end{cases} \tag{4.27}$$

with $\mathbf{v}^0 = \mathbf{1}$ as the one vector with size N and $\mathbf{M} = \exp(\frac{-\mathbf{C}}{\epsilon})$, where the division is element-wise. The transport plan $\mathbf{\Pi}$ can be computed for each iteration with

$$\mathbf{\Pi}^k = \operatorname{diag}(\mathbf{u}^k)\ \mathbf{M}\ \operatorname{diag}(\mathbf{v}^k). \tag{4.28}$$

Having a solution of Equation (4.26) we can still use Equation (4.15) to compute the transport. Because of the entropic regularization we do not have a bijective mapping, but it is possible that a new point is a combination of several source points. We choose to use a winner-takes-it-all solution by using the mapping

$$\mathbf{\Pi}: \quad \mathcal{X} \to \mathcal{Y}, \tag{4.29}$$

$$X_i \mapsto Y_{j^*} \text{ with } \quad j^* \in \arg\max_j \Pi_{ij}, \tag{4.30}$$

which means that we use the match with the highest probability.

4.1.6 Colorization Using Optimization

We use the algorithm proposed by Levin et al. [22] to achieve a final coloration. Based on the results of the matching from Section 4.1.5, we just color the centers of the superpixels of g^t. The colorization of the whole superpixels would achieve visually unpleasing results. With the centers colored, we use Levin et al.'s algorithm [22] to color the whole image. This algorithm was designed to perform the colorization of grayscale images by predefined color scribbles, and we use the colored center of the superpixels as such. The principle of this algorithm is that neighboring, similar bright or dark pixels should also have similar color.

4.1.7 Spatial Consistency

To improve the results of the image colorization from Section 4.1.6, especially around object borders, we utilize spatial consistency. We perform image segmentation of $\hat{c}_1^t$ based on Felzenszwalb and Huttenlocher [7]. With the segmentation we sort out all segments which contain less than three superpixels (from Section 4.1.4). For each of the segments with more or equal than three superpixels in it, we perform a k-means ($k = 2$) clustering of the color channels. If the count of the cluster is less than 25% of the overall count, then we reassign the center of this clusters with the average color of the other cluster and again perform the algorithm from Section 4.1.6. This spatial consistency optimization is done only once to improve the final colorization c^t.

4.1.8 Video Colorization with Bijective Assignment

The previously described method can be used to colorize a grayscale image sequence $G \in \mathbb{G}^T$ as well. With a given grayscale sequence G and a colored image c^r, we want to find a sequence of color images $C \in \mathbb{C}^T$ where C_t corresponds to G_t. The extension of the single image colorization to video sequences is done via Algorithm 1.

Algorithm 1: Video Colorization with Bijective Assignment

Input: $G \in \mathbb{G}^T$, $c^r \in \mathbb{C}$
Output: $C \in \mathbb{C}^T$, where C_t corresponds to G_t

1 Perform the steps from Section 4.1.1 - 4.1.4 for the whole sequence G as well as the color image c^r.
2 Generate the bijective assignment $\mathbf{\Pi}$ from Section 4.1.5 between c^r and the first grayscale frame G_1.
3 Colorize frame G_1 using Section 4.1.6 and Section 4.1.7 to achieve C_1.
4 **for** $t = 2 \rightarrow T$ **do**
5 Use the previously calculated bijective assignment $\mathbf{\Pi}$ as a basis, look-up table and then apply Section 4.1.6 and Section 4.1.7 to G_t.

4.2 Evaluation

We evaluate our approach visually. The used images are extracted from [9]. The implementation details are listed in Section 4.2.1 and the results are presented in Section 4.2.2.

4.2.1 Implementation Details

This approach was realized in Python and as additional packages we used `OpenCV` and `skimage`. We use $\mathsf{n\_segments} = 2000$ for the skimage implementation of the *SLIC* superpixels. For the Felzenswalz segmentation, also already implemented in skimage, we use the following parameters: $\mathsf{scale} = 200$, $\mathsf{sigma} = 0.4$, $\mathsf{min\_size} = 10$. The parameter ϵ for the

Sinkhorn-Knopp algorithms is set to 0.17, $\mathbf{p}$ and $\mathbf{q}$ are chosen to be an uniform distribution, *i.e.* $\forall i : p_i = q_i = \frac{1}{N}$. All following results were calculated with the Sinkhorn-Knopp algorithm and the whole procedure was calculated on the Central Processing Unit (CPU).

4.2.2 Results

In the following sections we present the results of single image colorization (Section 4.2.2.1) and video colorization (Section 4.2.2.2). The quality of the colorization is evaluated visually.

4.2.2.1 Image Colorization

Figure 4.2 visualizes the result of our described method for single image colorization. The results for a variety of images are shown. It can be seen that there is a problem with the colorization around object borders. This may be due to visually wrong matches between the grayscale and the reference image. Also, the colorization of objects is not always continuous, this can be seen in the first colorization example. Here, the color of the castle is not white everywhere. There are blue areas, which lead to visually unpleasing results. A reason for this may be the calculated segmentation with *SLIC*-superpixels. Another factor, especially for this colorization example, may be the used reference image. The color of the reference castle is darker than the grayscale input castle. The proposed method may not be able to transfer this color to the grayscale image because of this difference. This example showcases the importance of a proper color reference image to result in a visual pleasing colorization.

4.2.2.2 Video Colorization

Figures 4.3 - 4.5 show results for the video colorization using our proposed method. For the first two sequences we chose reference images that are not part of the sequence. The reference image for the last image is an already colored frame of the sequence. Again, we observe that the colorization for objects in a single frame is not always continuous and the border problem is also present. By analyzing successive frames we notice flickering. This may be due to the fact that we do not use any information from one frame to another. The results for the sequence toddler, see Figure 4.5, suggests that when the reference image is an already colored frame of the sequence, the colorization is visually more pleasing. The other two sequences may achieve better results with other color reference images.

4.3 Conclusion

We proposed a novel image and video colorization approach motivated by the work of Gupta et al. [9]. We adapted the original idea by introducing the method of *OT* into the matching routine. Following the first initial results and comparing the run-time of

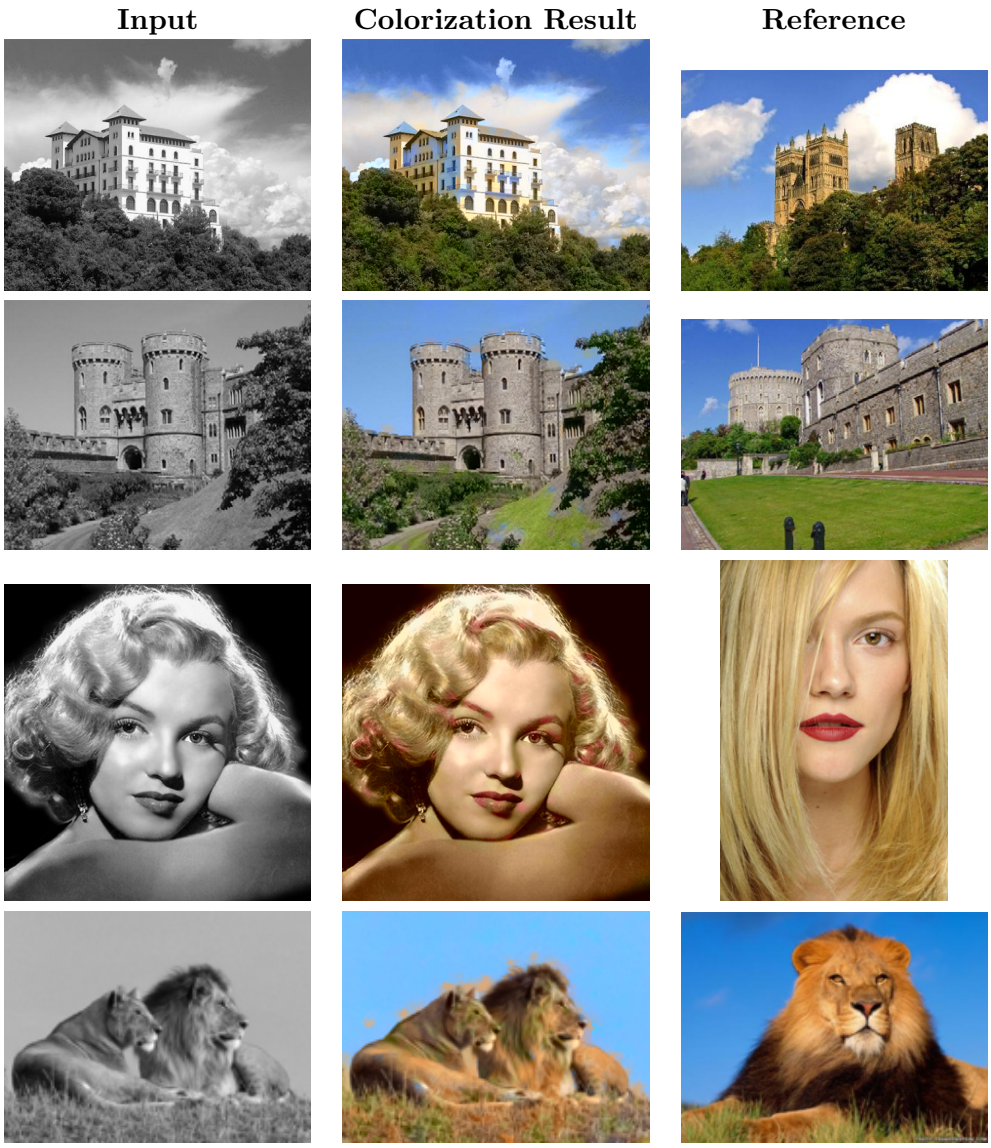


Figure 4.2: Colorization examples for various images. In every row a single colorization example is shown. The images in the first column are target grayscale images g^t. The right column shows colored source images c^s that are used as the color reference. In the middle column the colorization result for each individual example is shown. The results were computed with the proposed method and the Sinkhorn-Knopp algorithm. Implementation details can be found in Section 4.2.1. Images are taken from [9].

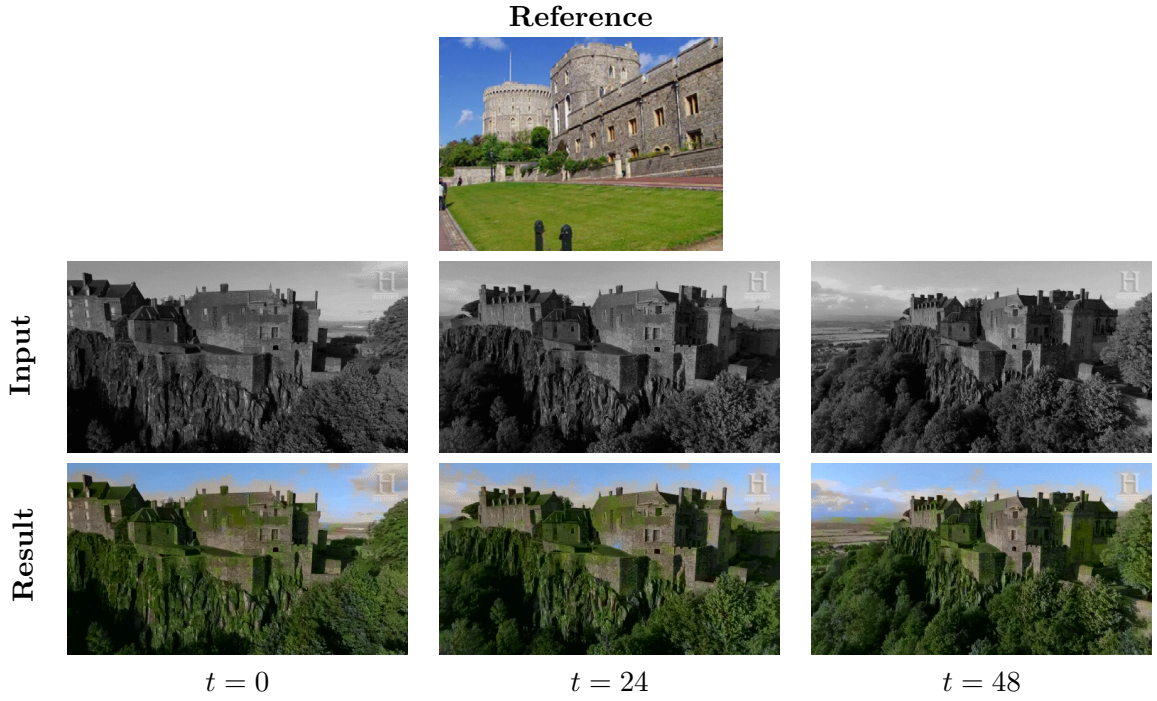


Figure 4.3: Colorization example for sequence castle. This figure visualizes the colorization of a grayscale video sequence G (Input) with the help of a color reference image c^r (Reference). With the proposed method from Section 4.1.8, a color sequence C (Result) is calculated. The results were achieved with the Sinkhorn-Knopp algorithm. Implementation details can be found in Section 4.2.1. Sequence is taken from Giphy[2].

[2] `https://giphy.com/gifs/historyuk-history-riverhunters-river-hunters-1yMfsnXPwZptM2buUU`, Accessed: 4^{th} May 2020

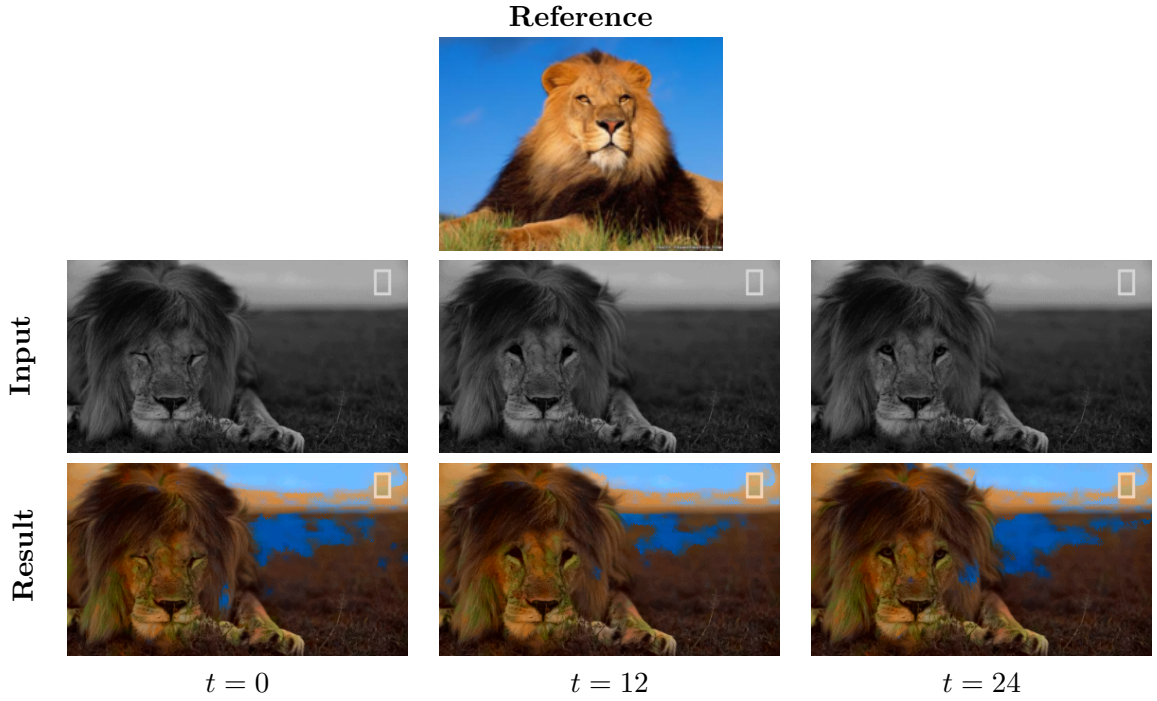


Figure 4.4: Colorization example for sequence lion. This figure visualizes the colorization of selected frames of a grayscale video sequence G (Input) with the help of a color reference image c^r (Reference). With the proposed method from Section 4.1.8, a color sequence C (Result) is calculated. The results were achieved with the Sinkhorn-Knopp algorithm. Implementation details can be found in Section 4.2.1. Sequence is taken from Giphy[3].

[3] `https://giphy.com/gifs/funny-lion-qPYcMv3bduwEg`, Accessed: 4^{th} May 2020

Figure 4.5: Colorization example for sequence toddler. This figure visualizes the colorization of selected frames of a grayscale video sequence G (Input) with the help of a color reference image c^r (Reference) from the same sequence. With the proposed method from Section 4.1.8, a color sequence C (Result) is calculated. The results were achieved with the Sinkhorn-Knopp algorithm. Implementation details can be found in Section 4.2.1. Sequence is taken from [22].

the different algorithms, we decided to just evaluate the Sinkhorn-Knopp algorithm. The Primal-dual algorithm needs many iterations to converge. Also, we decided to calculate the method on the *CPU*. Some of the procedures are computational expensive and a parallel implementation on a Graphics Processing Unit (GPU) would increase the performance.

Aside from the high run-time, the results are quite promising and visually pleasing. The problem of a non-continuous colorization and the object borders may be because of a wrong segmentation from the *SLIC*-superpixels. Additionally, the used reference image has to include all colors and correctly colored objects to obtain a sufficient colorization. To achieve a flicker-free colorization of video sequences, additional information for subsequent frames is needed.

5

Colorization using Convolutional Neural Networks (CNNs)

Contents

The approach with Optimal Transport (OT) was an appropriate entry in the field of image colorization. In the following chapter we exploit the modern approach of *CNNs* to overcome the problem of a long run-time. Additionally, we need to make sure that the colorization of video sequences is coherent in subsequent frames and contentious for objects.

The usage of *CNNs* is currently state-of-the-art for colorizing grayscale images and videos. Based on our results from Chapter 4, we decided to switch our considerations to the more modern approach of using *CNNs*. As Nielsen [28] mentions, *CNNs* are Artificial Neural Networks (ANNs) with one or more convolutional layers. A convolutional layer takes an image and a parametric kernel as input and convolves the whole image with the kernel. Additionally, many different methods and techniques have been developed to improve these networks. The Central Processing Unit (CPU) based implementation of the described method from Chapter 4 has the downside of being slow. With the help of different frameworks like TensorFlow or PyTorch we can efficiently generate *CNNs* on the Graphics Processing Unit (GPU) and use its massive parallelism to speed up computations.

Our considerations are based on Schaub et al.'s [33] work. They split the task of video colorization into a global and a local method. The local color transfer, coloring subsequent frames, has the drawback of worsening through time, whereas the global color transfer, coloring frames based on a common source image, lacks coloring details. The fusion of both strategies overcomes the downside of each individual strategy.

We extend the approach of [33] and we introduce a colorization technique based on

a forward and a backward path, see Figure 5.1. For this we use two colored frames per scene, one at the beginning of the sequence and one at the end. Instead of one global and one local color transfer (starting from the beginning) we utilize two global and two local (starting from the beginning *resp.* starting from the end) color transfers. We choose this concept to prevent the color fading over time as described in [33]. If there appear new objects during the scene that are not part of the colored first frame, we would not be able to achieve appropriate colorizations. But if these objects are contained in the newly introduced last colored frame, we are now able to color these objects adequately. Since we have two colored image frames from the sequence, we can categorize our task into an example-based method, where we want to distribute the colors of the reference frames onto the whole sequence. We also establish a consistency map for each individual color transfer. This consistency map is a pixel-wise measure and it defines the certainty of the image colorization. The combination of the different color transfer stages is done via the help of a *CNN*. We test different *CNN* architectures and evaluate the models based on ground-truth images.

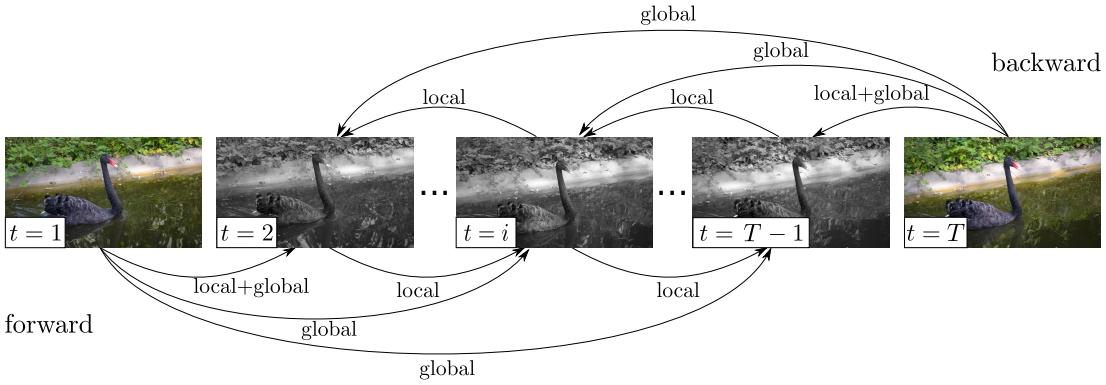


Figure 5.1: Concept of video colorization. We use two color reference images at frame $t = 1$ and $t = T$ to colorize the whole grayscale video sequence of length T. Our method combines a forward and a backward path, each consisting of a global and local color transfer. For the local stages, we continuously propagate color onward.

5.1 Method

Our goal is to colorize a grayscale image sequence by transferring the colors of the first and last frame throughout the sequence. With the introduced backward path we have five *CNN* based parts to achieve the colorization: a global color transfer forward, a local color transfer forward, a global color transfer backward, a local color transfer backward, and the fusion of the previous segments, see Figure 5.2.

With a given grayscale image sequence with length T and $G \in \mathbb{G}^T$, we want to find a

sequence of color images $C \in \mathbb{C}^T$ using the already colored first frame C_1 (corresponds to G_1) and last frame C_T (corresponds to G_T), where C_t corresponds to G_t. We introduce a mapping $\mathcal{T}$, which maps from a grayscale sequence G and colored frames C_1 and C_T to a colored sequence C:

$$\begin{aligned} \mathcal{T} : (\mathbb{G}^T, \mathbb{C}, \mathbb{C}) &\to \mathbb{C}^T, \\ (G, C_1, C_T) &\mapsto C. \end{aligned} \tag{5.1}$$

The colorization of *e.g.* grayscale frame G_t, using the colored reference frames C_1 and C_T yields: $\mathcal{T}_t(G_t, C_1, C_T) = C_t$. $\mathcal{T}$ is the combination *resp.* fusion of different sub-mappings via a *CNN*. A sub-mapping is also a mapping that produces intermediate results. We also introduce a confidence sequence $D \in \mathbb{G}^T$, that measures the certainty of each sub-mapping. A confidence sequence D^m of the sub-mapping m consists of $D^m = \{d_t^m\}_{t=1}^T$, where d_t^m is a pixel-wise measure of how certain we are, that the colorization C_t^m of the grayscale frame G_t from the sub-mapping m is correct. This confidence map ranges from 0 to 1, where 1 indicates that the colorization is accurate and 0 means the colorization is inaccurate. We use these confidence maps as an additional input for our fusion routine to improve the final colorization.

$\mathcal{T}$ can be written as the combination of its sub-mappings:

$$\mathcal{T}_t(G_t, C_1, C_T) = \mathcal{F}(\mathcal{T}_{\text{global}}(G_t, C_1), \mathcal{T}_{\text{local}}(G_t, C_{t-1}), \mathcal{T}_{\text{global}}(G_t, C_T), \mathcal{T}_{\text{local}}(G_t, C_{t+1}), G_t), \tag{5.2}$$

with $\mathcal{T}_{\text{global}}$ and $\mathcal{T}_{\text{local}}$ being sub-mappings. All sub-mappings and mappings $\mathcal{T}$ can be seen as warping operators.

As seen in Equation (5.2), we can split this task into five subtasks, see also Figure 5.2:

1. Global color transfer forward: Starting from the initial colored frame C_1 we transfer its colors to the grayscale frame: $\mathcal{T}_{\text{global}}(G_t, C_1) = (C_t, D_t)^{G,f}$,
2. Local color transfer forward: Starting from initial colored frame C_1 we transfer the colors frame-by-frame throughout the entire video ($C_1 \to C_2 \to \cdots \to C_{t-1}$): $\mathcal{T}_{\text{local}}(G_t, C_{t-1}) = (C_t, D_t)^{L,f}$,
3. Global color transfer backward: Starting from the initial colored frame C_T we transfer its colors to the grayscale frame: $\mathcal{T}_{\text{global}}(G_t, C_T) = (C_t, D_t)^{G,b}$,
4. Local color transfer backward: Starting from initial colored frame C_T we transfer the colors frame-by-frame throughout the entire video ($C_T \to C_{T-1} \to \cdots \to C_{t+1}$): $\mathcal{T}_{\text{local}}(G_t, C_{t+1}) = (C_t, D_t)^{L,b}$
5. Fusion: combination of all previous subtasks: $\mathcal{F}(\cdot)$.

Each individual subtask is explained in the following sections. We use superscripts to define the sub-mappings that provides the intermediate results. *E.g.* $C^{G,b} \in \mathbb{C}^T$ defines a

sequence of color images which was calculated with G, b, indicating the "global backward" color transfer.

By analyzing the different color spaces (Section 3.5) and the problem we are facing, we decided to work in the YCbCr color space. In this color space, grayscale images can simply be encoded as Y-channel data. The formula for calculating the Y-channel in YCbCr color space is the same as for converting images in RGB color space to grayscale images. Furthermore, the conversion between RGB and YCbCr is much simpler than *e.g.* LAB, and the task of colorizing grayscale images is more canonical when we use the given grayscale images as the L-channel. With this knowledge we just need to estimate the Cb and Cr channels.

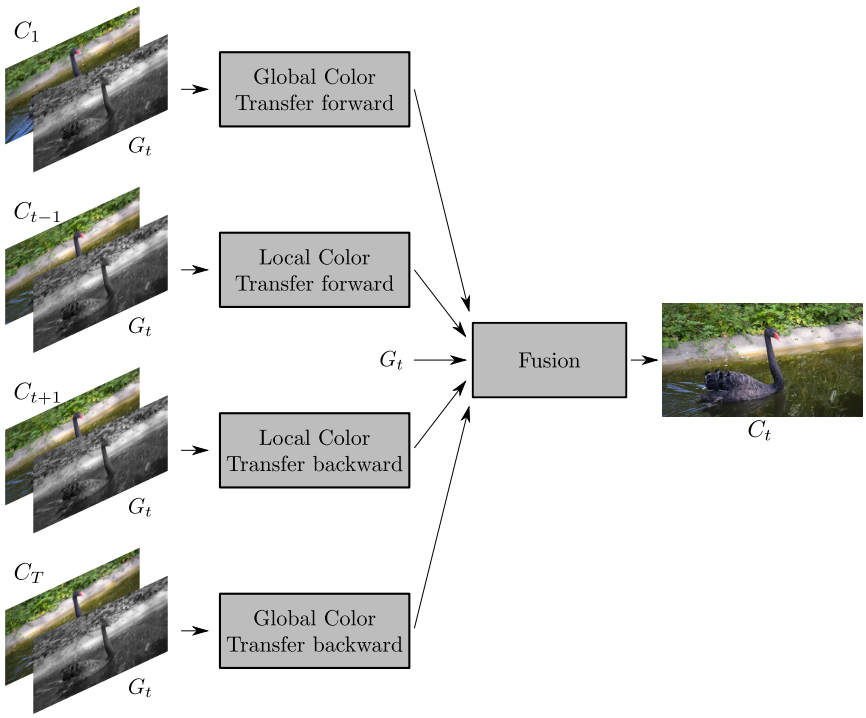


Figure 5.2: Approach of video colorization. We split the colorization of the grayscale frame G_t into four parts: the global color transfer in forward direction between the color image C_1 and G_t, the local color transfer in forward direction between C_{t-1} and G_1, as well as the global and local color transfer in backward direction with C_T and G_t, *resp.* C_{t+1} and G_t. We then combine these four parts and the grayscale image G_t in a separate fusion stage to achieve a final colorization C_t of the grayscale frame G_t.

5.1.1 Global Color Transfer

We want to colorize objects in the grayscale video sequence coherently throughout the video. In order to achieve this aim we use a global method. This global method extracts

semantic information of the scene to transfer color in space (where are objects in a frame) and time (where move objects during frames). This global, semantic color transfer is essential for coloring dis- and reappearing objects. Subsequent-frame based methods would loose the localization.

In Figure 5.3 the concept of the global color transfer can be seen. The basis for the global color transfer is an already colorized reference image C_1, from which we transfer the color onto the grayscale images. We utilize a two-stage feature matching between the grayscale variant G_1 of our reference color image C_1 and the grayscale target image we want to color, *e.g.* G_t. We use a pre-trained *CNN*, a ResNet-101, that was trained for image segmentation to extract coarse and fine image features. Based on the coarse features we perform a coarse matching. We refine this initial matching using finer features in a local neighborhood.

Following the previous notation, the global color transfer $\mathcal{T}_{\text{global}}$, between a reference color image C_1 and a grayscale target image G_t, can be written as:

$$\begin{aligned} \mathcal{T}_{\text{global}} : (\mathbb{G}, \mathbb{C}) &\rightarrow (\mathbb{C}, \mathbb{G})^G, \\ (G_t, C_1) &\mapsto (C_t, D_t)^{G,f}. \end{aligned} \tag{5.3}$$

The resulting global color transfer from C_1 to G_t, with $t > 1$ indicating forward direction, gives: *e.g.* $(C_t, D_t)^{G,f} = \mathcal{T}_{\text{global}}(G_t, C_1)$. Here we introduce the superscript G, f to indicate forward global colorization.

We define a global optical flow in backward direction $\hat{\mathbf{w}}^G = (\hat{u}, \hat{v})$. The optical flow is the apparent motion in a sequence of images. It can be described by a vector field that transforms one frame in a sequence into the next frame. For each pixel the optical flow gives an estimate of motion [37]. $\hat{u}$ and $\hat{v}$ are the components of the optical flow in horizontal and vertical direction. $\hat{\mathbf{w}}^G$ is the optical flow from the target grayscale frame G_t to the reference grayscale image G_1:

$$\tilde{G}_t^{G,f} = \mathcal{W}(G_1, \hat{\mathbf{w}}^G), \tag{5.4}$$

where $\tilde{G}_t^{G,f}$ should ideally be equal to G_t. Due to occlusions or strong motion the optical flow can be wrong in some regions and therefore $\tilde{G}_t^{G,f}$ may not be equal to G_t. $\mathcal{W}(p, \mathbf{w})$ is defined as an operator that warps input p based on the optical flow $\mathbf{w}$. The outcome of this operator has the same dimensionality as the input p. The optical flow $\hat{\mathbf{w}}^G$ gets calculated in the two matching stages. With this flow we are able to define our global color transfer also as:

$$C_t^{G,f} = \mathcal{W}(C_1, \hat{\mathbf{w}}^G). \tag{5.5}$$

5.1.1.1 Feature Extraction

As already mentioned, the basis for the global color transfer is a *CNN* trained for image segmentation. In our case we use a network trained for image segmentation and fine-tuned

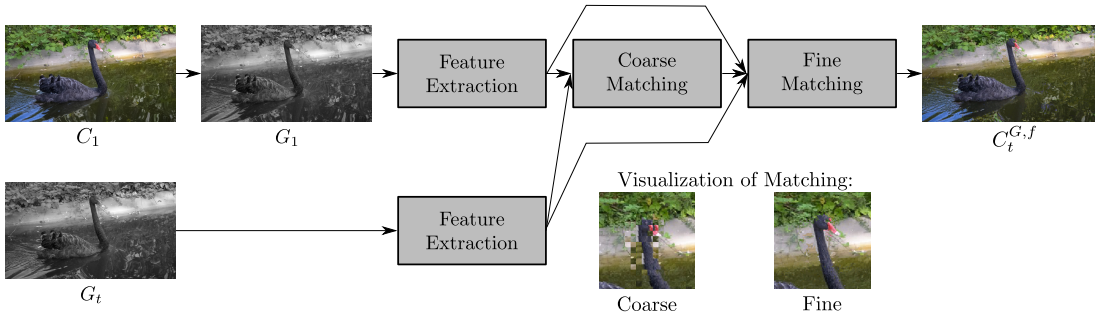


Figure 5.3: Concept of global color transfer. Given a grayscale image G_t and a reference color image C_1, we estimate the forward colorization $C_t^{G,f}$ of G_t via a two-stage feature matching. We extract features of each grayscale variant of the input images. Then we perform a global, coarse matching and a refinement in a local neighborhood to achieve the final colorization.

for semantic image segmentation, a ResNet-101 ([5] and [11]). ResNets rely on residual functions (Figure 5.5) that increase accuracy without increasing the networks complexity. We use this network to extract image features. We define a feature extraction function $\mathcal{E}(g, l) \mapsto \Phi_l^g$, that extracts features from a grayscale image $g \in \mathbb{G}$ at the layer l. With that we extract coarse $\Phi_{l_c}^{G_\tau}$ and fine features $\Phi_{l_f}^{G_\tau}$ for both grayscale images, with $\tau = \{1, t\}$, see Figure 5.4. The coarse features are extracted from a deep layer (l_c) of the network, whereas the finer features are extracted from a shallower layer (l_f). The *CNN* used for feature extraction can be replaced with any other network trained for image segmentation.

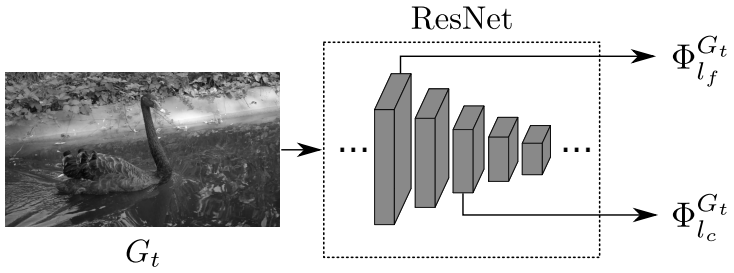


Figure 5.4: Feature extraction using a pre-trained *CNN*. To extract the features $\Phi_{l_c}^{G_t}$ and $\Phi_{l_f}^{G_t}$ for the grayscale image G_t, which are needed for our two-stage feature matching, we use a pre-trainied *CNN* trained for semantic image classification (here ResNet). The coarse features $\Phi_{l_c}^{G_t}$ are extracted from a deep layer l_c from the network, the fine features $\Phi_{l_f}^{G_t}$ from a shallower layer l_f.

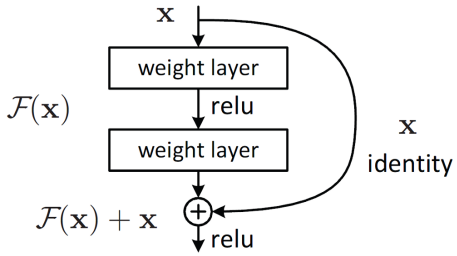


Figure 5.5: Architecture of ResNet. ResNets rely on residual functions. Skip-connections are used to jump over layers. Image is taken from [11].

5.1.1.2 Coarse Matching

Given the coarse features $\Phi_{l_c}^{G_1}$ of the reference grayscale image G_1 and the coarse features $\Phi_{l_c}^{G_t}$ of the target grayscale image G_t, we calculate a similarity $S_{G_t,G_1}((i,j),(i',j'))$ (Equation (5.6) with $l = l_c$) for each location over the whole space. We calculate the matching via Equation (5.7) to obtain the initial coarse matching $\tilde{C}_t^G$ and the initial coarse flow $\hat{\mathbf{w}}_c^G$, see Figure 5.6.

$$\mathcal{S}_{G_t,G_1}((i,j),(i',j')) = \|\Phi_l^{G_t}(i,j) - \Phi_l^{G_1}(i',j')\|_2^2 \tag{5.6}$$

$$\tilde{C}_{tkij}^{G,f} = C_{1k\hat{i}\hat{j}} \forall k \in \{1,2,3\} \text{ with } (\hat{i},\hat{j}) = \underset{(i' \in [1..M],\, j' \in [1..N])}{\arg\min} \mathcal{S}_{G_t,G_1}((i,j),(i',j')) \tag{5.7}$$

We also obtain $\tilde{C}_t^{G,f}$ via:

$$\tilde{C}_t^{G,f} = \mathcal{W}(C_1, \hat{\mathbf{w}}_c^G). \tag{5.8}$$

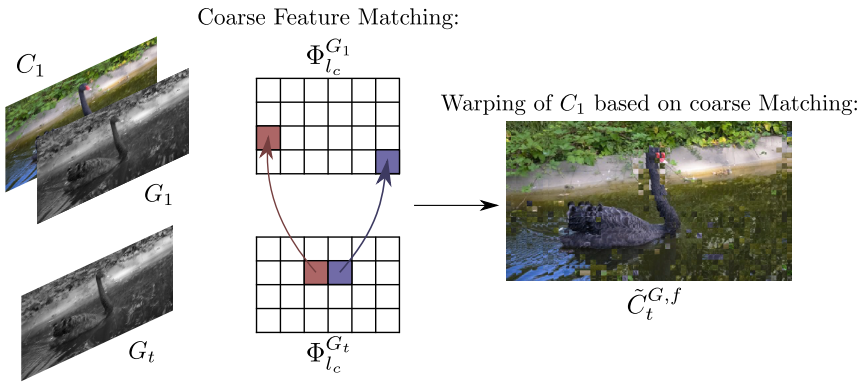


Figure 5.6: Coarse matching. Given the coarse features $\Phi_{l_c}^{G_1}$ and $\Phi_{l_c}^{G_t}$ of G_1 and G_t, we calculate a matching based on finding the most similar feature. The initial, coarse matching $\tilde{C}_t^{G,f}$ is patchy.

After the coarse matching we notice outliers in the coarse flow $\hat{\mathbf{w}}_c^G$, see Figure 5.7. These outliers have the properties of salt-and-pepper noise. Therefore, we choose a median

filter $\mathcal{M}_{3x3}$ to smoothen these outliers and to get a better initial matching. This median filter $\mathcal{M}_{3x3}$ extracts for a pixel location (i, j) a patch of size 3×3, flattens the patch, orders its values ascending and writes the central element to the location (i, j):

$$\hat{\mathbf{w}}_{c,f}^{G} = \mathcal{M}_{3x3}(\hat{\mathbf{w}}_{c}^{G}). \tag{5.9}$$

This operation is applied for each channel individually. Figure 5.8 shows the coarse flow, filtered with the median filter. It can be seen that outliers are removed and that the flow becomes smoother.

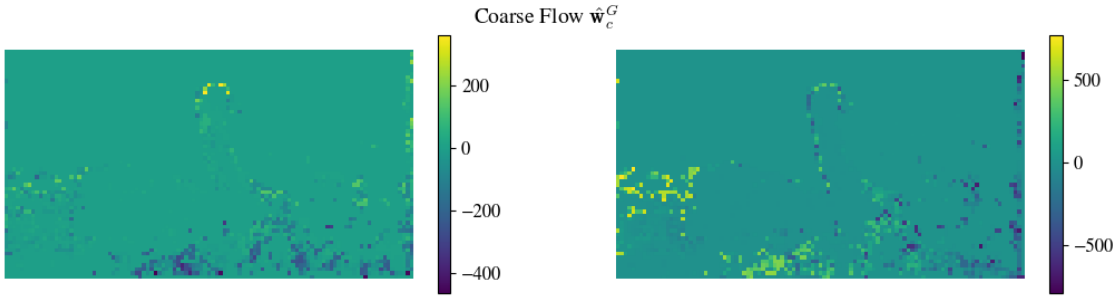


Figure 5.7: Unfiltered coarse flow. The calculated coarse flow $\hat{\mathbf{w}}_{c}^{G}$ has outliers due to the initial matching. The optical flow is resized for illustrative purposes, left: horizontal component of $\hat{\mathbf{w}}_{c}^{G}$, right: vertical component of $\hat{\mathbf{w}}_{c}^{G}$.

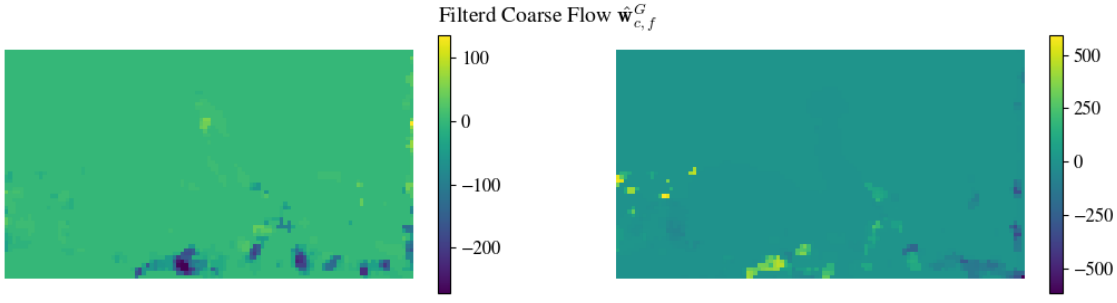


Figure 5.8: Filtered coarse flow. We apply a median filter to the coarse flow (Figure 5.7) to smooth out any outliers, we obtain $\hat{\mathbf{w}}_{c,f}^{G}$. The optical flow is resized for illustrative purposes, left: horizontal component of $\hat{\mathbf{w}}_{c,f}^{G}$, right: vertical component of $\hat{\mathbf{w}}_{c,f}^{G}$.

5.1.1.3 Fine Matching

Based on the initial matching and the extracted, fine features $\Phi_{l_f}^{G_1}$ for G_1 and $\Phi_{l_f}^{G_t}$ for G_t, we warp $\Phi_{l_f}^{G_1}$ via $\hat{\mathbf{w}}_{c,f}^{G}$ to align both feature maps:

$$\tilde{\Phi}_{l_f}^{G_1} = \mathcal{W}(\Phi_{l_f}^{G_1}, \hat{\mathbf{w}}_{c,f}^{G}). \tag{5.10}$$

We then calculate another similarity $S_{G_t,G_1}((i,j),(i',j'))$ as in Equation (5.6), but with $l = l_f$, $\Phi_{l_f}^{G_t}$ and the warped feature map $\tilde{\Phi}_{l_f}^{G_1}$. This time we compute the similarity just for the local neighborhood, where we use a 25×25 neighborhood, *i.e.* $i' \in [i-12\,..\,i+12]$ with $i \geq 1$ and $i \leq N$ *resp.* $j' \in [j-12\,..\,j+12]$ with $j \geq 1$ and $j \leq M$. The final matching is again calculated via Equation (5.7), which yields the final flow $\hat{\mathbf{w}}^{G}$ and the final image $C_t^{G,f}$. This procedure can be seen in Figure 5.9.

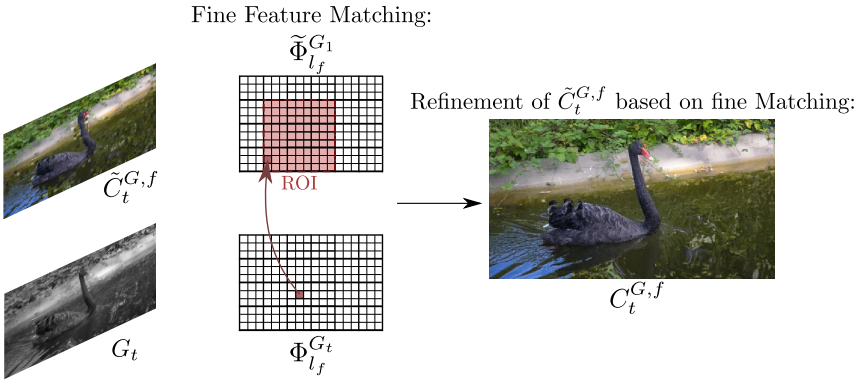


Figure 5.9: Fine matching. Given the fine features $\tilde{\Phi}_{l_f}^{G_1}$ and $\Phi_{l_f}^{G_t}$ corresponding to G_1 *resp.* G_t, we calculate a matching based on finding the most similar feature in a local neighborhood, Region Of Interest (ROI). After the refinement we achieve a smoother image $C_t^{G,f}$.

5.1.1.4 Confidence

As already mentioned, we also introduce a confidence map. The confidence map is calculated by combining a confidence calculated via the optical flow and a confidence based on the difference of single color channels.

For the optical flow confidence $D_{t,\text{flow}}^{G,f}$ we calculate a confidence based on the forward and backward optical flow between two frames (via forward-backward consistency check). Aside from the final global optical flow in backward direction $\hat{\mathbf{w}}^{G}$ from the target grayscale frame G_t to reference grayscale image G_1, we also calculated the global optical flow in forward direction $\mathbf{w}^{G}$, from G_1 to G_t. By warping the optical flow in forward direction

$\mathbf{w}^G$ using the optical flow in backward direction $\hat{\mathbf{w}}^G$ we obtain the flow $\tilde{\mathbf{w}}^G$,

$$\tilde{\mathbf{w}}^G = \mathcal{W}(\mathbf{w}^G, \hat{\mathbf{w}}^G), \tag{5.11}$$

which should ideally be equal to the opposite optical flow in backward direction $\hat{\mathbf{w}}^G$. Now we estimate the confidence $D^{G,f}_{t,\text{flow}}$ via:

$$D^{G,f}_{t,\text{flow}} = \min(1, \max(0, 1 - |\tilde{\mathbf{w}}^G + \hat{\mathbf{w}}^G|)). \tag{5.12}$$

We clip $D^{G,f}_{t,\text{flow}}$ to fit within the range $[0, 1]$ and for dis-occluded regions we set the confidence to 0. For dis-occluded regions, it holds that:

$$|\tilde{\mathbf{w}}^G + \hat{\mathbf{w}}^G| > 0.01(|\tilde{\mathbf{w}}^G|^2 + |\hat{\mathbf{w}}^G|^2) + 0.5. \tag{5.13}$$

The coefficients for Equation (5.13) were taken from [36]. An example optical flow confidence can be seen in Figure 5.10. The confidence measure based on the difference of color

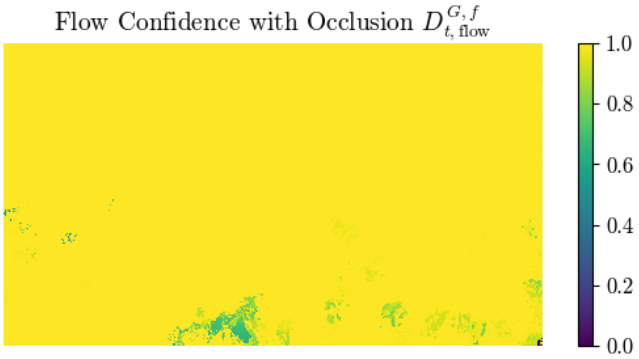


Figure 5.10: Flow confidence. Based on a forward-backward consistency check between the optical flows $\hat{\mathbf{w}}^G$ and $\mathbf{w}^G$ we calculate a flow consistency $D^{G,f}_{t,\text{flow}}$. Higher values indicate a high certainty that the color transfer is correct.

channels $D^{G,f}_{t,\text{color}}$ is defined by:

$$D^{G,f}_{t,\text{color}} = \exp(-100|\tilde{C}^{G,f}_{t1} - G_t|), \tag{5.14}$$

and an example can be seen in Figure 5.11. We calculated the final confidence map $D^{G,f}_t$ for the colorization $C^{G,f}_t$ with

$$D^{G,f}_t = D^{G,f}_{t,\text{flow}} \cdot D^{G,f}_{t,\text{color}}. \tag{5.15}$$

The combined confidence from the optical flow (Figure 5.10) and the difference of color channels (Figure 5.11) can be seen in Figure 5.12.

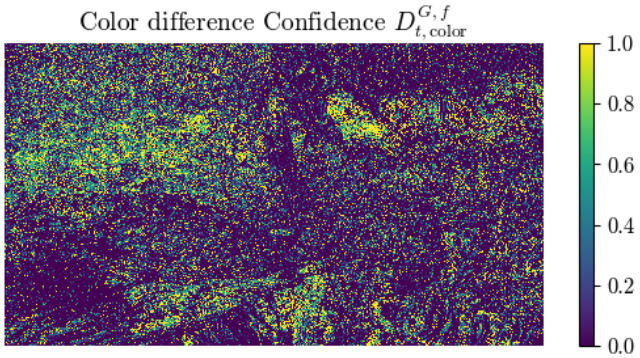


Figure 5.11: Color difference confidence. As an additional confidence $D^{G,f}_{t,\text{color}}$ we calculate the difference between the Y-channel of our estimation $C^{G,f}_{t1}$ and G_t. We also apply an exponential function to scale this confidence.

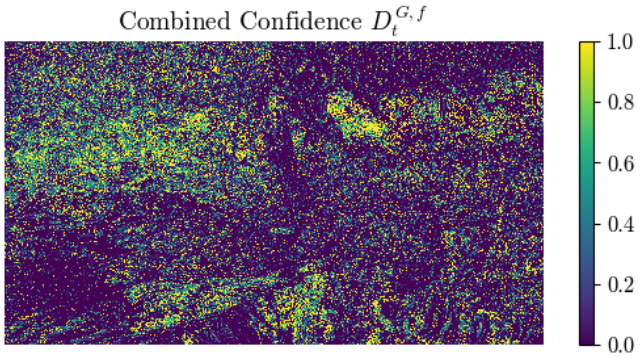


Figure 5.12: Combined confidence. We multiply the optical flow confidence (Figure 5.10) with the color channel confidence (Figure 5.11) to achieve a combined confidence.

5.1.2 Local Color Transfer

We utilize the assumption that for subsequent frames, the intensity values for given pixels do not change, but their location, to propagate colors from an already colored image to following grayscale images. This baseline is true as long the illumination of the scene does not change. We follow the concept shown in Figure 5.13. The colorization of the grayscale frame G_t relies on the already established colorization C_{t-1} of the previous grayscale frame G_{t-1}. We obtain the optical flow between this two consecutive frames via a pre-trained network, the PWC-Net, and estimate an initial warping of C_{t-1}. We then use a feature matching routine to refine our warping in a local neighborhood to achieve a final colorization $C^{L,f}_t$.

We can write the local color transfer $\mathcal{T}_{\text{local}}$, between an already colored image C_{t-1} and

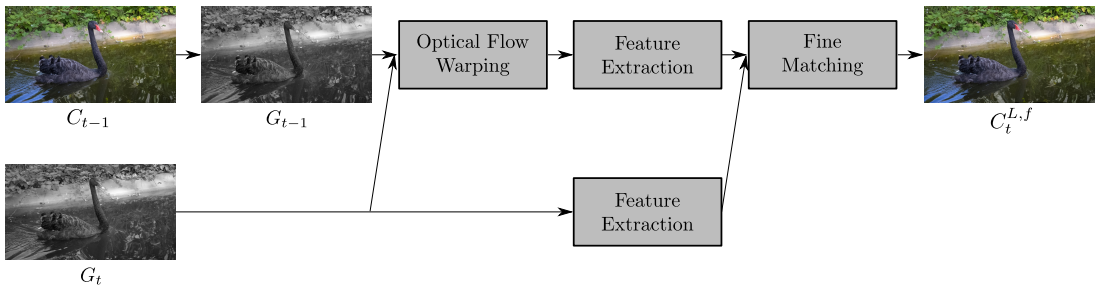


Figure 5.13: Concept of local color transfer. Given a grayscale image G_t and a previous reference color image C_{t-1} we estimate the colorization $C_t^{L,f}$ of G_t via a local optical flow estimation and a local refinement.

its following grayscale target image G_t as:

$$\begin{aligned} \mathcal{T}_{\text{local}} : (\mathbb{G}, \mathbb{C}) &\rightarrow (\mathbb{C}, \mathbb{D})^L, \\ (G_t, C_{t-1}) &\mapsto (C_t, D_t)^{L,f}. \end{aligned} \tag{5.16}$$

We define the colorization as a forward direction when we colorize from index $t-1$ to t and indicate this with the superscript L, f, *e.g.* $\mathcal{T}_{\text{local}}(G_t, C_{t-1}) = (C_t, D_t)^{L,f}$.

As in Section 5.1.1, we define a local optical flow in backward direction $\hat{\mathbf{w}}^L = (\hat{u}, \hat{v})$ from the target grayscale frame G_t to reference grayscale image G_{t-1}:

$$\tilde{G}_t^{L,f} = \mathcal{W}(G_{t-1}, \hat{\mathbf{w}}^L). \tag{5.17}$$

Ideally $\tilde{G}_t^{L,f}$ should be equal to G_t. This optical flow gets calculated in two stages, via the estimation of the already trained network and with the feature matching refinement. With the final flow we are able to define our global color transfer also as:

$$C_t^{L,f} = \mathcal{W}(C_{t-1}, \hat{\mathbf{w}}^L). \tag{5.18}$$

5.1.2.1 Flow Estimation

We use the PWC-Net, proposed by Sun et al. [35], to estimate the backward optical flow $\hat{\mathbf{w}}_c^L$ from G_t to G_{t-1} as well as the forward optical flow $\mathbf{w}_c^L$ from G_{t-1} to G_t. The networks architecture can be seen in Figure 5.14. PWC-Net uses many different optical flow estimation techniques, such as image pyramid, warping and cost volume in a deep neural network to calculate the optical flow. The used PWC-Net can be replaced with any other optical flow estimation method or network. Using $\hat{\mathbf{w}}_c$ we obtain an initial colorization $\tilde{C}_t^{L,f}$

$$\tilde{C}_t^{L,f} = \mathcal{W}(C_{t-1}, \hat{\mathbf{w}}_c^L), \tag{5.19}$$

and the warped grayscale image $\tilde{G}_t^L$

$$\tilde{G}_{t,c}^{L,f} = \mathcal{W}(G_{t-1}, \hat{\mathbf{w}}_c^L). \tag{5.20}$$

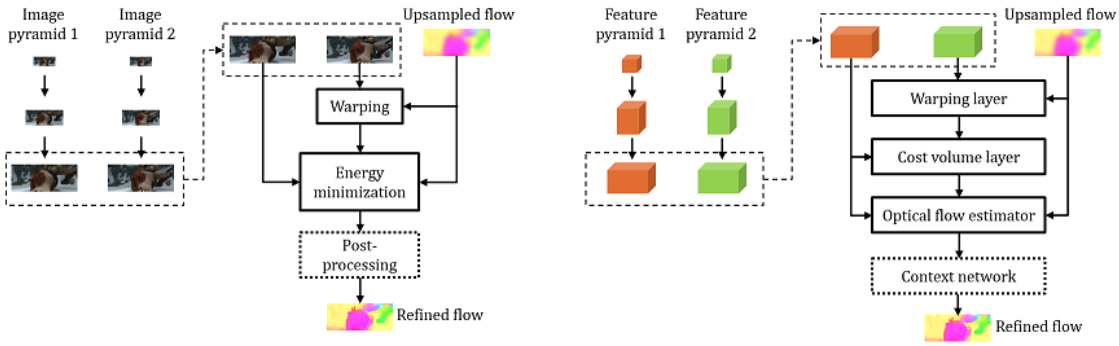


Figure 5.14: Architecture of PWC-Net. Many different optical flow estimation techniques are combined to calculate the optical flow. Image is taken from [35].

5.1.2.2 Fine Matching and Confidence

To refine our initial results we extract and match fine features for the images G_t and as in Section 5.1.1.1 and Section 5.1.1.3. We end up with the final backward flow $\hat{\mathbf{w}}^L$, the final image $C_t^{L,f}$ and the final forward flow $\mathbf{w}^L$. The calculation of the local confidence $D_t^{L,f}$ follows Section 5.1.1.4.

5.1.3 Combination of forward and backward Paths

As previously mentioned, we do not just use a forward path but also a backward path. Therefore, we calculate the global color transfer $\mathcal{T}_{\text{global}}(\cdot,\cdot)$ (Section 5.1.1) in backward direction: for the reference color image C_T and grayscale images G_t for $t < T$, as well as the local color transfer $\mathcal{T}_{\text{local}}(\cdot,\cdot)$ (Section 5.1.2) for pairs of images (G_t, C_{t+1}). Each of the color transfers $\mathcal{T}_i$ with $i \in \{\text{global}, \text{local}\}$, in forward and backward direction, estimates now a color image C_t, as well as a confidence map D_t.

We also introduce an initial fusion $\mathcal{F}_{\text{init}}$ (Section 5.1.4.1), such that we have a defined previous $(t-1)$ *resp.* next $(t+1)$ colorized image for every local color transfer. We also use this initial fusion as an additional input for the fusion procedure. The final fusion for a grayscale image G_t, given two initial color images C_1 and C_T, can now be written as:

$$\begin{aligned}\mathcal{T}_t(G_t, C_1, C_T) = \mathcal{F}(&\mathcal{T}_{\text{global}}(G_t, C_1), \mathcal{T}_{\text{local}}(G_t, C_{t-1}),\\ &\mathcal{T}_{\text{global}}(G_t, C_T), \mathcal{T}_{\text{local}}(G_t, C_{t+1}), G_t, \mathcal{F}_{\text{init},t}).\end{aligned} \tag{5.21}$$

Since we just work with the color channels (Cb, Cr) of our sub-mappings and not with the Y-channel we adapt the fusion

$$\begin{aligned} \mathcal{T}_t(G_t, C_1, C_T) = \mathcal{F}(&\mathcal{S}(C_t^{G,f}), D_t^{G,f}, \mathcal{S}(C_t^{L,f}), D_t^{L,f}, \\ &\mathcal{S}(C_t^{G,b}), D_t^{G,b}, \mathcal{S}(C_t^{L,b}), D_t^{L,b}, G_t, \mathcal{S}(C_t^M)), \\ C_t = \mathcal{T}_t(&G_t, C_1, C_T), \end{aligned} \tag{5.22}$$

with $\mathcal{S}(c)$ extracting the two color channels $k = \{2, 3\}$ of $c \in \mathbb{C}$ and $C_t^M = \mathcal{F}_{\text{init},t}(C_t^{G,f}, D_t^{G,f}, C_t^{G,b}, D_t^{G,b}, C_t^{L,f}, D_t^{L,f})$. If the color frame C_{t-1} *resp.* C_{t+1} is not known, we use the result from the initial fusion for it:

$$\begin{aligned} \mathcal{F}_{\text{init},t-1}(C_{t-1}^{G,f}, D_{t-1}^{G,f}, C_{t-1}^{G,b}, D_{t-1}^{G,b}, C_{t-1}^{L,f}, D_{t-1}^{L,f}) = C_{t-1}^M, \\ \mathcal{F}_{\text{init},t+1}(C_{t+1}^{G,f}, D_{t+1}^{G,f}, C_{t+1}^{G,b}, D_{t+1}^{G,b}, C_{t+1}^{L,f}, D_{t+1}^{L,f}) = C_{t+1}^M. \end{aligned} \tag{5.23}$$

The concept of the previously described fusion can be seen in Figure 5.15. We start by calculating step 1 and recurrently propagate these calculations onward. When we reach the last frame of our sequence, we start from behind and propagate backwards. A propagation from frame 1 to frame T is defined as one run. Two runs mean that we propagate from the frames in the sequence: $1 \rightarrow T \rightarrow 1$.

5.1.4 Fusion

In this section we describe the methods to combine each individual local and global, forward and backward intermediate result. We also define the initial fusion and the different *CNNs* that we use for the final fusion. We also analyse different estimation methods.

5.1.4.1 Initial Fusion

We calculate an initial fusion in order to acquire a first result, which is then used for later local color transfers. We calculate the fusion in a recurrent way and just for one run. The calculation of the initial fusion $\mathcal{F}_{\text{init},t}$ depends on the results of $\mathcal{T}_{\text{global}}(G_t, C_1)$, $\mathcal{T}_{\text{global}}(G_t, C_T)$ and $\mathcal{T}_{\text{local}}(G_t, C_{t-1})$. These three transformations give three color estimates $C_t^{G,f}$, $C_t^{G,b}$ and $C_t^{L,f}$, as well as three corresponding confidence maps $D_t^{G,f}$, $D_t^{G,b}$ and $D_t^{L,f}$. The initial fusion image C_t^M is calculated via the maximizing element of the confidence maps at the location (i, j):

$$\begin{aligned} \mathcal{F}_{\text{init},t}(C_t^{G,f}, D_t^{G,f}, C_t^{G,b}, D_t^{G,b}, C_t^{L,f}, D_t^{L,f}) &= C_t^M \\ C_{tkij}^M &= C_{tkij}^{\hat{m}} \forall k \in \{1, 2, 3\}, \text{ with} \\ \hat{m} &= \arg\max_m D_{ij}^m \end{aligned} \tag{5.24}$$

for $m \in \{(G, f), (G, b), (L, f)\}$. This means that we use the sub-mapping with the highest confidence.

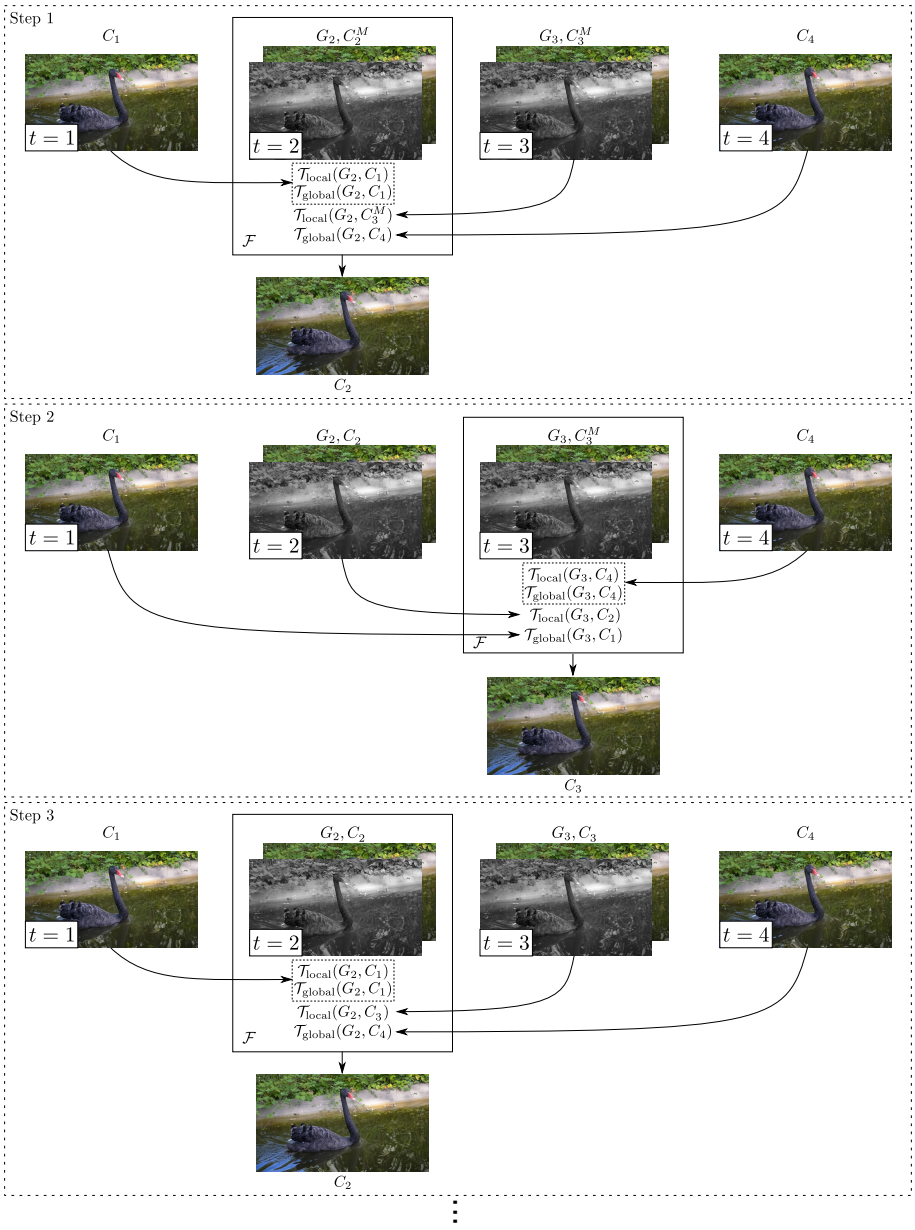


Figure 5.15: Combination of forward and backward path. This example shows the combination of the forward and backward path. We achieve the colorization C of the grayscale sequence G ($T = 4$) by consecutively running the fusion procedure, starting with colorizing frame $t = 2$, G_2, (Step 1). Here, frame $t = 1$ is already colorized (C_1) and it is used as the reference for the global and local forward color transfer. The initial fusion of frame $t = 3$, C_3^M, is the reference for the local color transfer in backward direction and the last color frame $t = T$, C_T, is the reference for the global color transfer backward. Based on these sub-results and the initial fusion of frame $t = 2$, C_2^M, we get a new colorization for frame $t = 2$, C_2, which is now the reference for the local forward color transfer for frame $t = 3$ (Step 2), and so on.

5.1.4.2 Learned Fusion

In this section we evaluate the different *CNNs* that we use to combine our results as described in Section 5.1.3. The goal of the fusion can be seen in Figure 5.16. The input consists of the previous results of the local and global color transfers, their color channels plus confidences, and the initial fusion. The used *CNNs* just estimate the color channels. For the Y-channel we simply copy the grayscale frame.

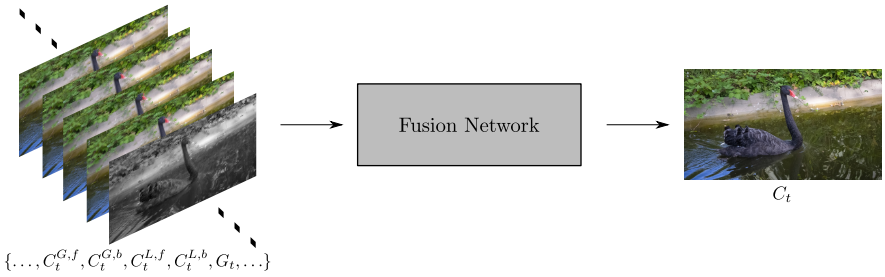


Figure 5.16: Concept of learned fusion. Given the sub-colorizations, their associating confidences, the initial fusion and the grayscale variant for the frame G_t, we use a fusion network to estimate the final colorization C_t.

As an initial network we decided to use a simple feed forward *CNN* (Figure 5.17). This network consists of convolutions and Rectified Linear Unit (ReLU) activation functions. To enlarge its receptive field, we progressively increase the dilation of the convolution, as proposed by Yu and Koltun [43].

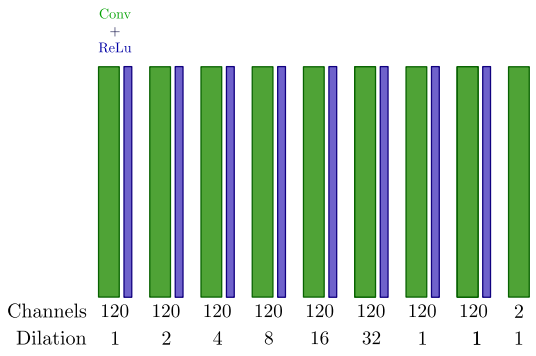


Figure 5.17: Simple network. This network uses convolutions and *ReLU* activation functions. To increase the receptive field we use a statically increasing dilation.

The next tested network is a DenseNet proposed by Huang et al. [13]. In their work they introduce a dense convolutional network, one block can be seen in Figure 5.18, in which each layer is connected to every other layer in a feed-forward fashion. DenseNets are

a combination of various blocks. For a single block, feature-maps of preceding layers are used as input for the current layer. An advantage of this architecture is that the number of parameter gets reduced compared to simpler *CNNs* that achieve similar results. We use a DenseNet consisting of four blocks, each block having five convolution and *ReLU* combinations and 20 output channels. As a transition layer between two blocks we also use a convolution and *ReLU* combination and 60 output channels. The final prediction layer is chosen to achieve an output of the same spatial dimension as the input, but with 2 channels for the color.

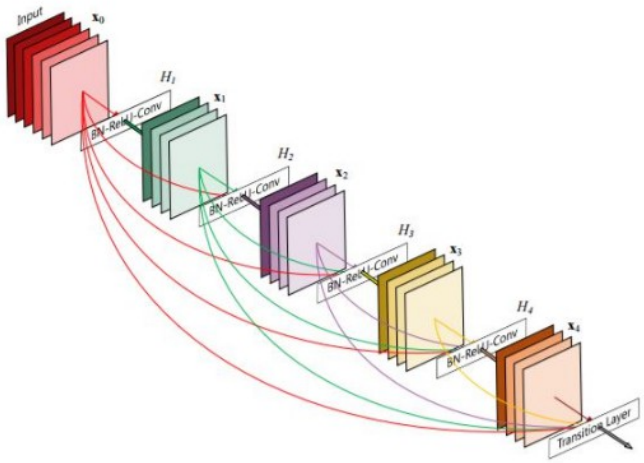


Figure 5.18: DenseNet. This figure visualizes a single DenseNet block. This block uses connections from each layer to every other in a feed-forward fashion. Feature-maps of preceding layers are used as input for the current layer. Image is taken from [13].

In Figure 5.19 the proposed network of Ronneberger et al. [32] is illustrated. This *CNN* architecture for image segmentation consists of a contracting path to capture context, and a symmetric expanding path that enables precise localization. This U-net architecture achieves good performance on different biomedical segmentation tasks and is also used in various areas. We use a contracting and expanding path of depth 5 and 64 channels for the output of the first contracting block. For the expanding block, we use bilinear upsampling. As for the DenseNet model, the final prediction layer parameters are set to achieve 2 channels. To get an output of the same spatial dimension as the input, we utilize padding.

For all of the networks we use a joint instance normalization Equation (5.25) for the inputs, introduced by Ulyanov et al. [38], to improve training and the prediction. We use this method to normalize all *e.g.* Cb-channels together, and then use the computed statistics to de-normalize the output later on.

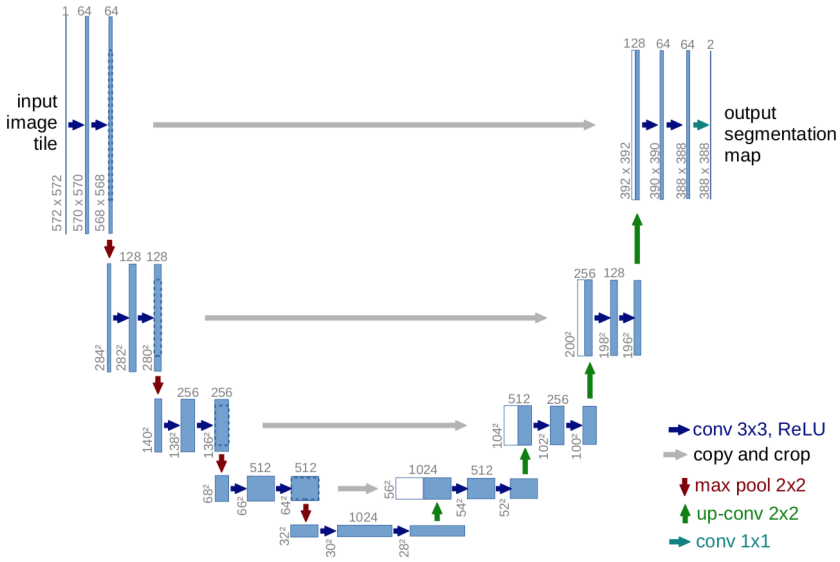


Figure 5.19: U-net. This network consists of a contracting and expanding path. Image is taken from [32].

Let $T \in \mathbb{R}^{m \times n \times o \times p}$ be an input tensor containing m batches, each with n channels of size $o \times p$. T_{tijk} denotes the $tijk$-th element of T. To calculate the joint instance normalization $\tilde{T}$ we extract for each batch the common channels n_Y, n_{Cb}, n_{Cr} and n_C, where n_Y, n_{Cb}, n_{Cr} correspond to the separate color channels of the YCbCr color space and n_C to the used confidences ($\#n_Y + \#n_{Cb} + \#n_{Cr} + \#n_C = n$). The joint instance normalization for a common channel n_i is then calculated via:

$$\begin{aligned} \mu_{tn_i} &= \frac{1}{n_i op} \sum_{i \in n_i} \sum_{j=1}^{o} \sum_{k=1}^{p} T_{tijk} \\ \sigma^2_{tn_i} &= \frac{1}{n_i op} \sum_{i \in n_i} \sum_{j=1}^{o} \sum_{k=1}^{p} (T_{tijk} - \mu_{tn_i})^2 \\ \tilde{T}_{tijk} &= \frac{T_{tijk} - \mu_{tn_i}}{\sqrt{\sigma^2_{tn_i} + \epsilon}}, \text{ with } i \in n_i, \end{aligned} \tag{5.25}$$

were ϵ denotes a constant to prevent division by zero.

5.1.4.3 Estimation Methods

We try out two different estimation methods for the final fusion. The first method is that we use the output of the *CNN* just as the new colors. Let $\hat{T} \in \mathbb{R}^{m \times 2 \times o \times p}$ be the output of

the *CNN*. We de-normalize the output using the computed statistics for the color channels Cb and Cr from (5.25) and obtain $\hat{T}'$:

$$\begin{aligned}\hat{T}'_{t1jk} &= (\hat{T}_{t1jk} \cdot \sigma_{tn_{Cb}}) + \mu_{tn_{Cb}} \\ \hat{T}'_{t2jk} &= (\hat{T}_{t2jk} \cdot \sigma_{tn_{Cr}}) + \mu_{tn_{Cr}}.\end{aligned} \tag{5.26}$$

The final colorization (for one batch $m = 1$ and $o = M$, $p = N$) becomes $C_t = \mathcal{C}(G_t, \hat{T}')$, with $\mathcal{C}$ being an operator that combines the grayscale image G_t and the two color channels of $\hat{T}'$ into an image $C_t \in \mathbb{C}$.

The second method is that we use the output of the *CNN* as the change of color. This residual estimation needs another de-normalization:

$$\begin{aligned}\hat{T}'_{t1jk} &= \hat{T}_{t1jk} \cdot \sigma_{tn_{Cb}} \\ \hat{T}'_{t2jk} &= \hat{T}_{t2jk} \cdot \sigma_{tn_{Cr}}.\end{aligned} \tag{5.27}$$

In this case the final colorization, also for one batch ($m = 1$, $o = M$ and $p = N$), is $C_t = \mathcal{C}(G_t, \mathcal{S}(C_t^M) + \hat{T}')$.

5.1.5 Training

Hence we use already pre-trained networks in Section 5.1.1 and Section 5.1.2, we just need to train our fusion *CNN*. We use the ℓ_1-norm evaluated on the difference of our estimation C_t and the ground-truth color image $\hat{C}_t$ as a loss:

$$\ell_1 = \|\mathcal{S}(C_t) - \mathcal{S}(\hat{C}_t)\|_1. \tag{5.28}$$

We compute the loss just on the color channels Cb and Cr. The Y-channel just gets copied.

5.2 Evaluation

We evaluate our approach with ground-truth colorized videos. A description of this dataset follows in Section 5.2.1. The implementation details for the evaluation are stated in Section 5.2.2. In Section 5.2.3 our results are presented.

5.2.1 Dataset

Our dataset consists of a total of 286 image sequences. The sequences were obtained from the DAVIS2017 video dataset ([29] and [30]) and the rotoscoping data set Roto++ [23]. All sequences are already colored and we perform the conversion to grayscale by ourselves. The sequences have a different number of frames, reaching from about 20 frames to 150 frames. Each image has a resolution of $480p$ (854×480 pixels). For training and evaluation of the different *CNNs* we split the sequences into 250 training sequences and 36 test sequences.

The training is done using patches of size 128×128 pixels, and we decrease the sequence length for training to 20 frames.

5.2.2 Implementation Details

The different parameters used for the *CNNs* are listed in the corresponding model definitions in Section 5.1.4.2. Additionally, we use the Adaptive Moment Estimation (ADAM)-optimizer with a learning rate of 10^{-4} to train our models. For the coarse layer of the feature extraction network, ResNet-101 ([5] and [11]), we use the output of the `conv3`-block as l_c. The fine layer l_f is chosen as the output of the first `conv1`-block. For the fine layer we set the stride of the first `conv1`-block to 1. This is done such the fine features have the same spatial dimension as the input images. The project was realized in Python and we used the `PyTorch` Application Programming Interface (API) to implement our *CNN* based approach. The training and testing was done on a Nvidia® GeForce Titan X.

5.2.3 Results

In the following sections we list the results of the different *CNNs* and estimation methods for one run (Section 5.2.3.1) and the results of running several runs for the U-net model with the residual estimation (Section 5.2.3.2).

As a quantitative measure we calculate the Peak Signal-To-Noise Ratio (PSNR)-value (5.29) in dB, between ground-truth color image $\hat{c} \in \mathbb{C}$ and the estimation $c \in \mathbb{C}$ in the YCbCr color space. To calculate the *PSNR*-value, we need to define the Mean Squared Error (MSE) for color images,

$$\begin{aligned} \text{MSE} &= \frac{1}{3MN} \sum_{k=1}^{3} \sum_{i=1}^{M} \sum_{j=1}^{N} (c_{kij} - \hat{c}_{kij})^2 \\ \text{PSNR} &= 10 \cdot \log_{10}\left(\frac{{v_{\max}}^2}{\text{MSE}}\right), \end{aligned} \tag{5.29}$$

with $v_{\max}$ being the maximum possible pixel value for $\hat{c}$. It holds that higher *PSNR*-values indicate a quantitatively better colorization.

5.2.3.1 Comparison *CNNs* and Estimation Methods

Table 5.1 gives a quantitative evaluation of the different fusion networks and estimation methods, each trained for one run, as well as the initial fusion (Section 5.1.4). We calculate the average *PSNR* value over all frames of each test sequence. The results show that the U-net residual model achieves the highest *PSNR* number, closely followed by the normal U-net estimation method. Compared to the initial fusion all *CNN* models boost this initial results. As for the learning based fusion procedures, the simple feed forward *CNN*, with both estimation methods, performs worst.

Model	initial fusion	U-net	DenseNet	Simple	U-net residual	DenseNet residual	Simple residual
Average *PSNR*	37.55	39.28	39.20	38.93	**39.30**	39.10	38.57

Table 5.1: Comparison of different *CNNs* regarding the average *PSNR* over all frames. The *PSNR*-values (dB) of the different *CNNs* and estimation methods calculated on the test set are listed.

The overall best architecture, U-net residual, also performs the best by evaluating average *PSNR* value over the first N frames, see Figure 5.20. The U-net and DenseNet networks, both without residual estimation, perform slightly worse. Again, it can be seen that the initial fusion performs worst.

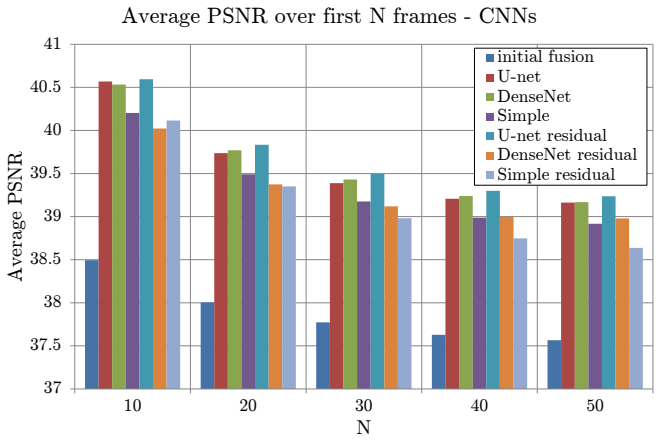


Figure 5.20: Comparison of different *CNNs* and estimation methods regarding the average *PSNR* over first N frames. The average *PSNR* (dB) over the first N frames is calculated for the different networks and estimation methods.

The average *PSNR* value per frame (Figure 5.21) is also in line with the previous results. It can also be seen, that there is a slight increase at the end of the curve. This may be due the fact, that we reach the end of the sequences, where we can transfer colors from the reference image. The results in Fig. 5.20 and 5.21 are just estimated over the first 50 frames. With many test sequences longer than 50 frames, these results can not indicate the impact of our proposed method regarding the use of second reference image at the end of the sequence. Figure 5.22 gives an overview of the calculated *PSNR* per frames for various example sequences. For these sequences we can see an improvement of

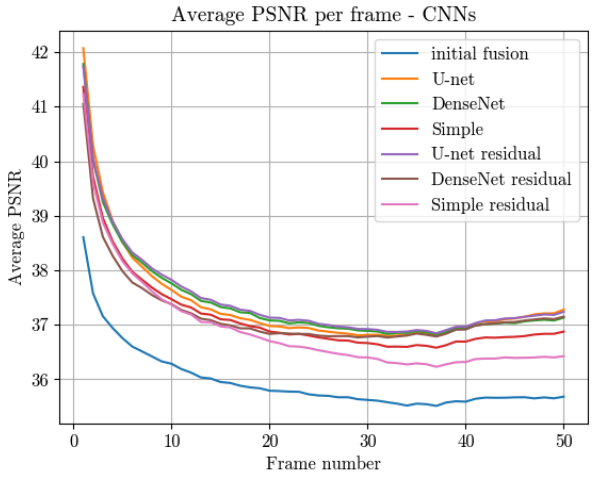


Figure 5.21: Comparison of the average *PSNR* per frame for different *CNNs* and estimation methods. The average *PSNR* (dB) is calculated per frame for different *CNNs* and estimation methods on the test set.

the *PSNR* when reaching the end of the sequence. The corresponding colorization results, achieved with the U-net residual estimation method, can be seen in Figure 5.23 - 5.26. We can see an improvement of the learned fusion compared to the initial fusion. The learned fusion is able to remove artifacts that are part of the initial fusion results.

5.2.3.2 U-Net-Residual: Multiple Runs

Following prior results, we decided to evaluate the U-net residual estimation model by training this method for two successive runs. We train the model from frame $1 \rightarrow T$ and then train from frame $T \rightarrow 1$, this means we train for one forward run and one backward run. Due to memory and time restrictions, we estimated temporary results after the first training run and based the training of the second run on them. We then recurrently estimate the colorization for a total of four runs and compare the results after each run. As can be seen in Table 5.2, we achieve the best results after 2 runs. From run 1 to run 2 there is an improvement, from run 2 onward, the average *PSNR* value over all frames and all sequences decreases.

By inspecting the average *PSNR* over the first N frames in Figure 5.27, we can see that for the first 10 and 20 frames the best results are achieved with 3 runs. For all other intervals of frames the best results are achieved with 2 runs.

Figure 5.28 visualizes the average *PSNR* value per frame for the U-net residual approach trained for multiple runs. Again, we can see that with 2 runs we achieve the overall highest *PSNR* values. With a higher run number there is a static decrease in the performance. The evaluation after 1 run shows the worst results for the first 20 frames. For

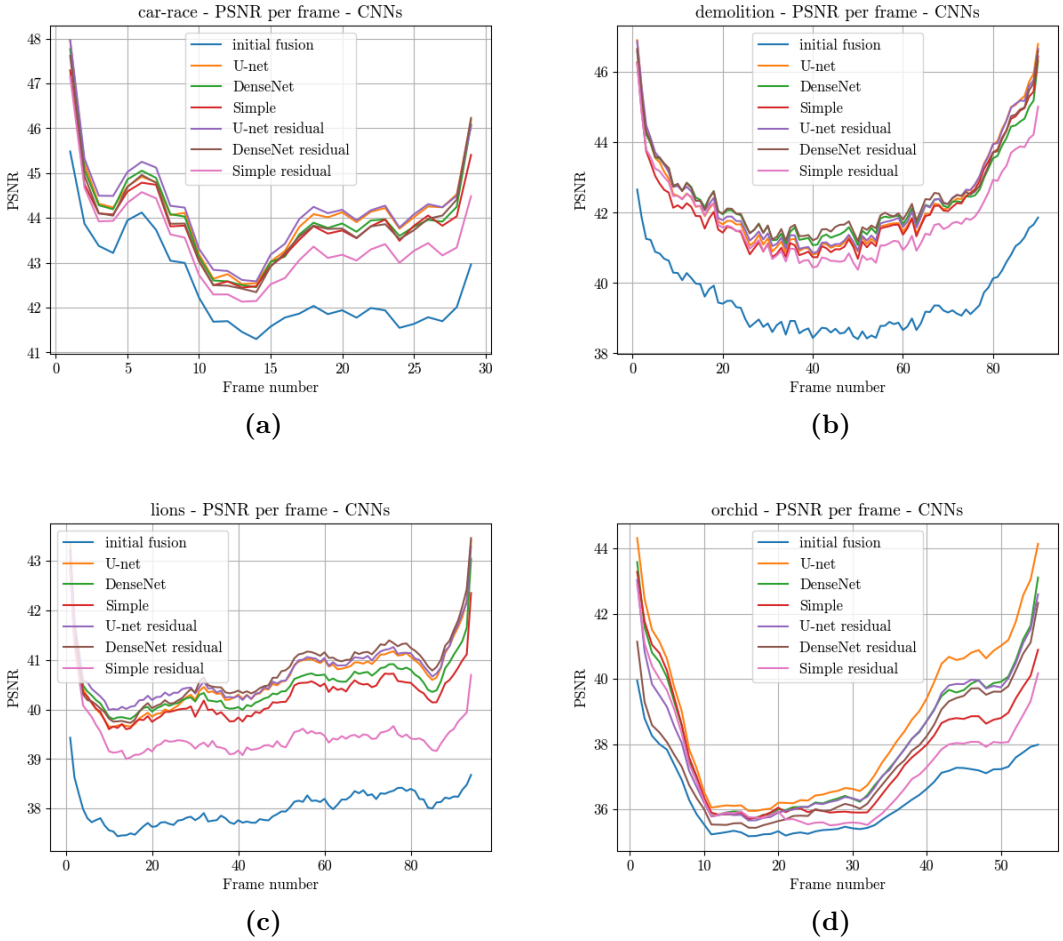


Figure 5.22: *PSNR* value per frame for individual sequences for different *CNNs* and estimation methods. (a): car-race, (b): demolition, (c): lions, (d): orchid. *PSNR* in dB.

later frames, the evaluation after 1 run starts to achieve better results than the evaluation after 3 and 4 runs. As in Section 5.2.3.1, the results in Figure 5.27 and 5.28 are just estimated over the first 50 frames. Therefore, these results can not indicate the impact of our proposed method regarding the use of second reference image at the end of the sequence, for sequences longer than 50 frames.

Figure 5.29 gives an overview of the calculated *PSNR* per frames for various example sequences. Again, we can seen an increasing *PSNR* value when reaching the end of the sequence. The corresponding colorization results, achieved with the U-net residual estimation method evaluated with 2 runs, can be seen in Figure 5.30 - 5.33. Figure 5.34 illustrates the colorization results for one frame of various sequences after different number of runs. The sequence chamaleon experiences an improvement for the colorization of

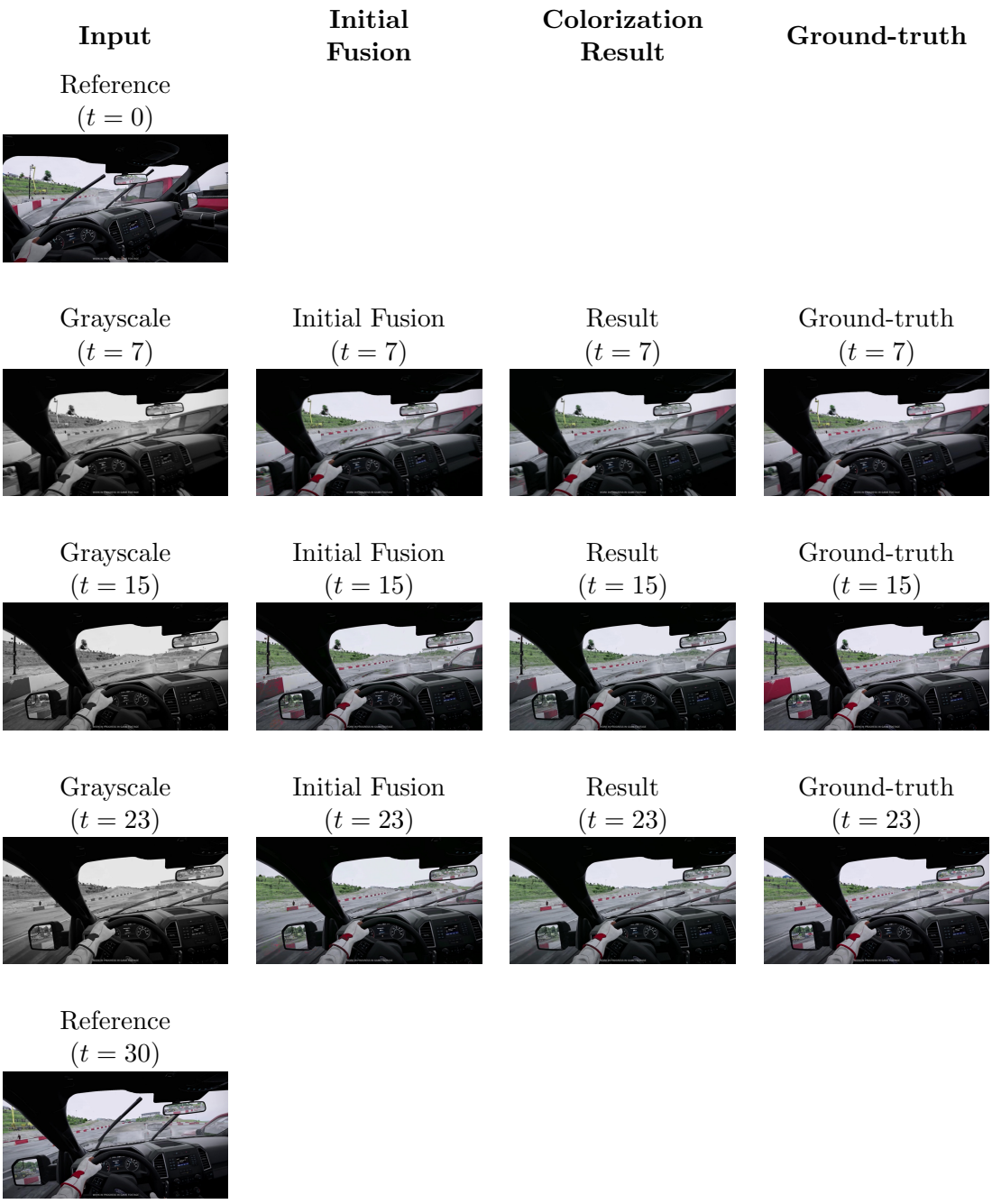


Figure 5.23: Colorization example for sequence car-race. The first column shows the input data. Here, the first ($t = 0$) and last ($t = 30$) frames are already colorized. The second column illustrates the results from the initial fusion. Our proposed method, with the U-net residual model, achieves the results in the third column. The last column visualizes the ground-truth color images. Sequence is taken from the test dataset.

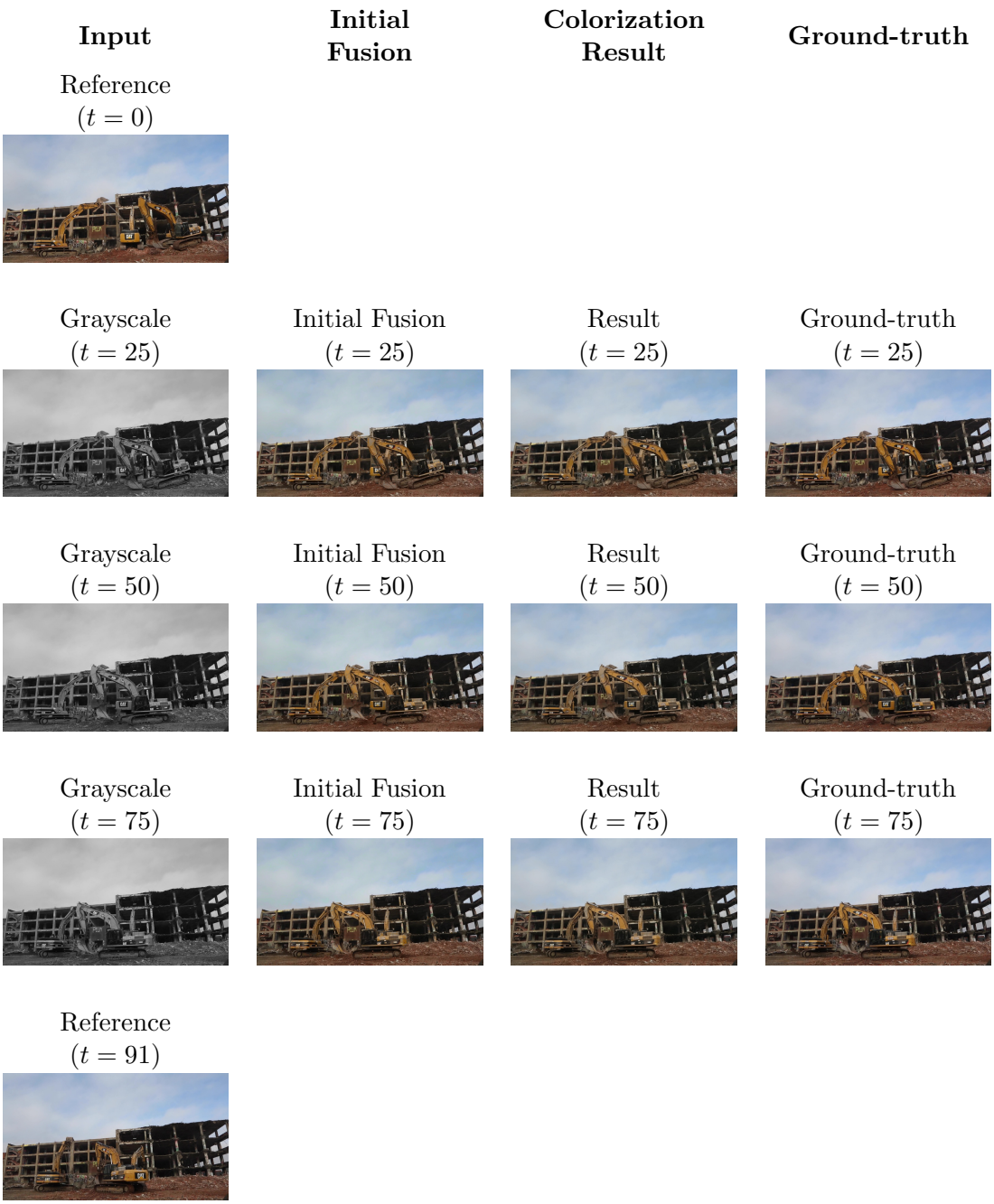


Figure 5.24: Colorization example for sequence demolition. The first column shows the input data. Here, the first ($t = 0$) and last ($t = 91$) frames are already colorized. The second column illustrates the results from the initial fusion. Our proposed method, with the U-net residual model, achieves the results in the third column. The last column visualizes the ground-truth color images. Sequence is taken from the test dataset.

Figure 5.25: Colorization example for sequence lions. The first column shows the input data. Here, the first ($t = 0$) and last ($t = 95$) frames are already colorized. The second column illustrates the results from the initial fusion. Our proposed method, with the U-net residual model, achieves the results in the third column. The last column visualizes the ground-truth color images. Sequence is taken from the test dataset.

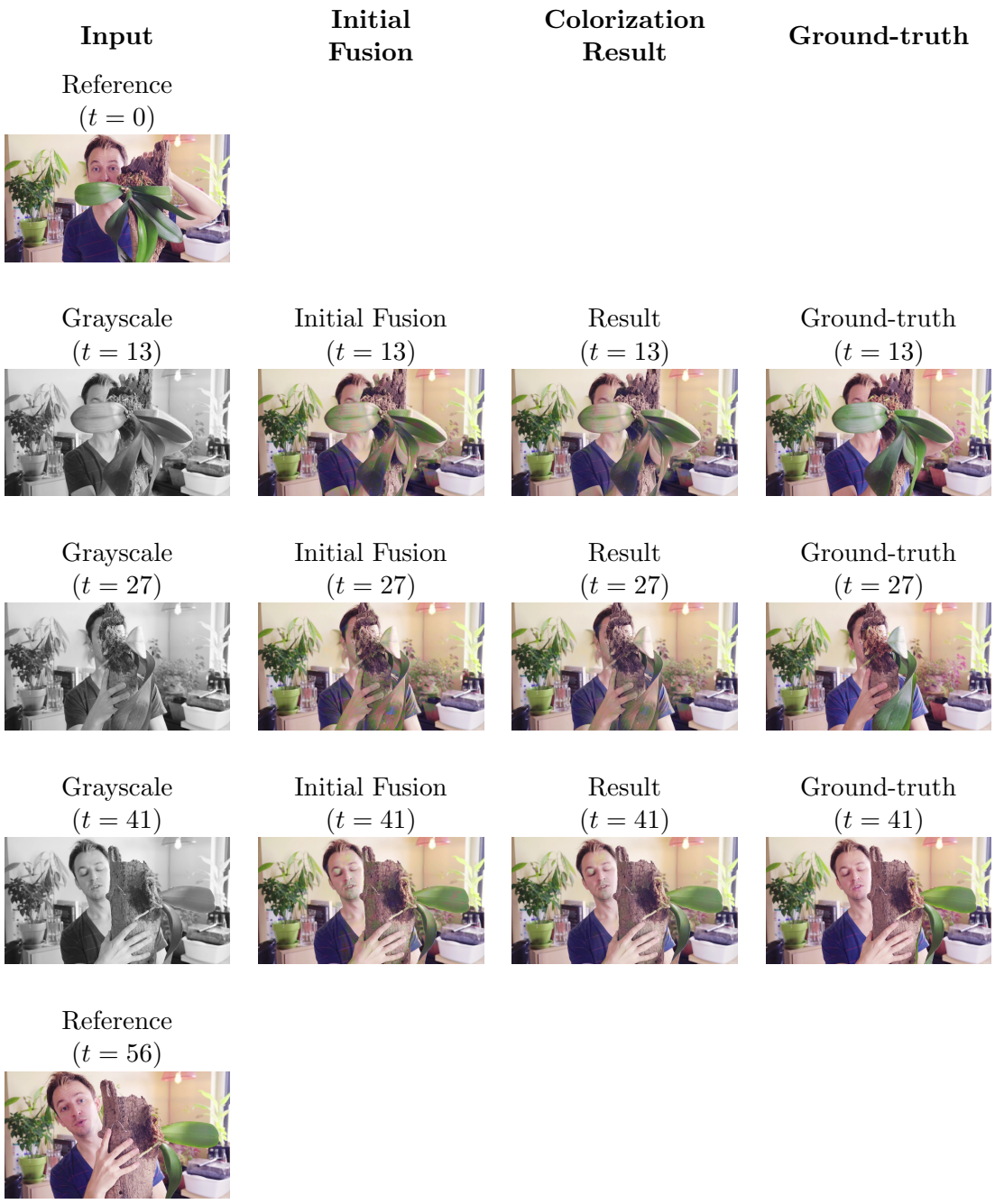


Figure 5.26: Colorization example for sequence orchid. The first column shows the input data. Here, the first ($t = 0$) and last ($t = 56$) frames are already colorized. The second column illustrates the results from the initial fusion. Our proposed method, with the U-net residual model, achieves the results in the third column. The last column visualizes the ground-truth color images. Sequence is taken from the test dataset.

Runs	1	2	3	4
Average *PSNR*	39.00	**39.14**	39.05	38.91

Table 5.2: Comparison of the U-net residual model for different runs regarding the average *PSNR* over all frames. The *PSNR*-values (dB) of the U-net residual model for different runs calculated on the test set are listed.

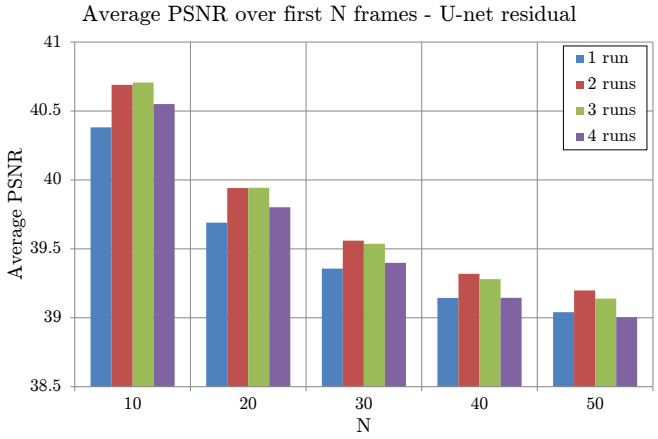


Figure 5.27: Comparison of different runs for the U-net residual model regarding the average *PSNR* over first N frames. The average *PSNR* (dB) over the first N frames is calculated.

the branches with higher runs, but artifacts appear on various leaves. The results show that for the sequence golf the colorization is getting worse with an increasing run number. Wrong colorizations from the initial fusion get worse with higher run numbers. For the sequence gym, it can be seen that the color of the training bag is getting more intense with higher runs and artifacts from the initial fusion start to disappear. The sequence ocean-birds undergoes smaller alterations. The color of the water changes to a slightly more blue hue with increasing runs.

5.3 Conclusion

In this chapter we introduced a novel approach for colorizing grayscale sequences. Our considerations are based on the work of Schaub et al. [33]. We modified the global and local color transfer and added an additional backward path to the colorization routine. To improve the final colorization we introduced a confidence map for each individual

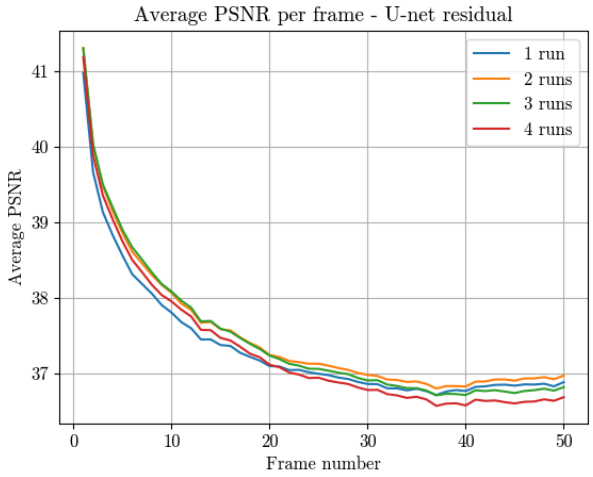


Figure 5.28: Comparison of the average *PSNR* per frame for different the U-net residual model for different runs. The average *PSNR* (dB) is calculated per frame for the U-net residual model over different runs on the test set.

sub-colorization. We added and tried different fusion *CNNs* to combine the separate colorization results and their associated confidence. In terms of *PSNR* measures between ground-truth and estimated colorization, the proposed method with the U-net fusion network and the residual estimation method achieved the best results. On our test set this network achieves 39.30dB. From the results in Table 5.1 we can see that each individual fusion network boosts the results of the initial fusion. As can be seen in the colorization results for the U-net residual model, Figure 5.23 - 5.26, the learned fusion is able to remove artifacts that are part of the initial fusion. The training of the U-net fusion network and the residual estimation method for two successive runs (one forward run and one backward runs) achieves best results with an estimation of the colorization after 2 runs. Here we achieve a *PSNR* measure of 39.14dB. After the second run we steadily achieve worse results. The achieved *PSNR* values for the run variation are all lower than the *PSNR* measures attained with the U-net and U-net residual method trained for 1 run, see Table 5.1. This may be due to the repeated residual estimation method that occurs after each run. The results in Figure 5.34 affirm the worsening of the colorization after each run. It can be seen that artifacts start to appear for some sequences and with each run they get worse. In each run, the fusion from the previous run is changing and the network may not be able to sufficiently learn this input variation. Also, the subsequent local color transfers rely on the previous fusion results, which may lead to an error propagation. The introduction of a confidence for the initial and learned fusion may help to overcome this behavior. Also a training without the need of computing intermediate results may improve these recurrent method. By comparing the *PSNR* measure for single sequences we can see that we achieve high numbers at the beginning and the end of sequences. This is due to the fact that the

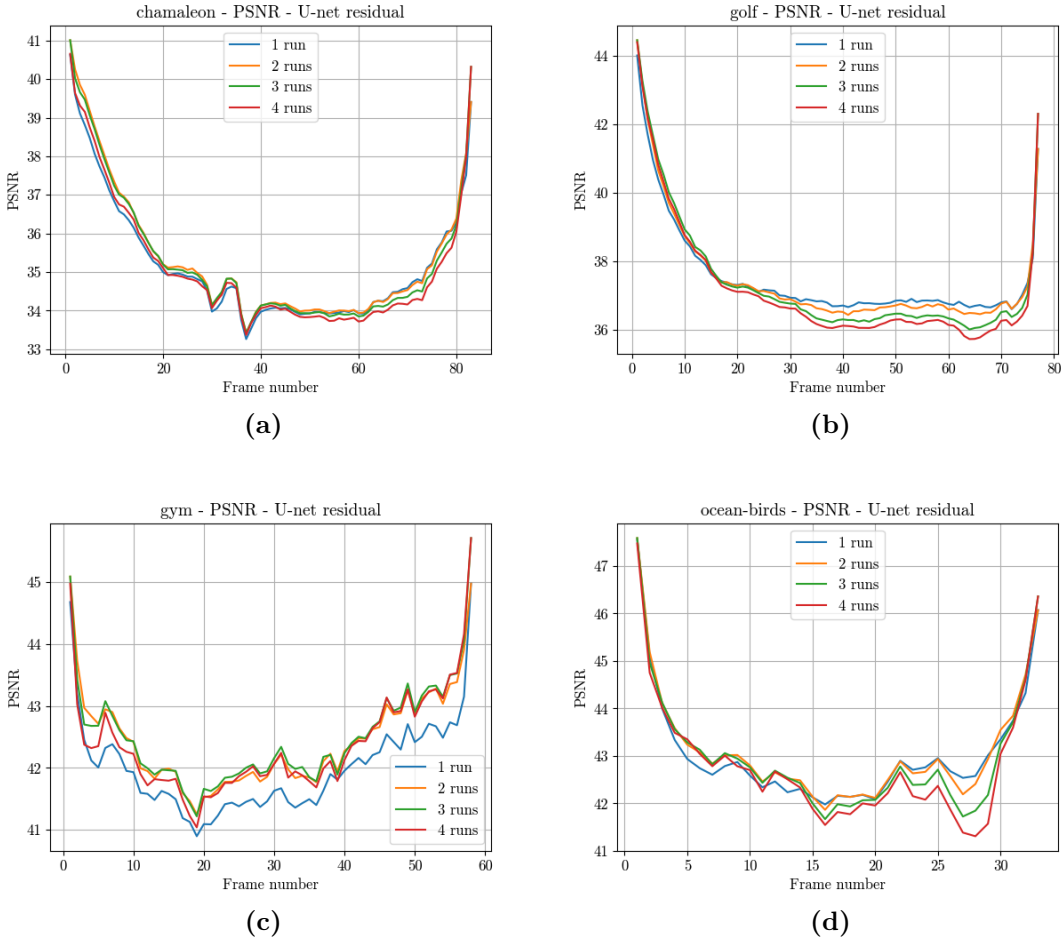


Figure 5.29: *PSNR* value per frame for individual sequences for the U-net residual model for different runs. (a): chameleon, (b): golf, (c): gym, (d): ocean-birds. *PSNR* in dB.

first and last image of the sequence are already colorized and the color propagation around these images is more correct. The additional reference image at the end of the sequence improves the color fading. This behavior can be seen in the "bathtub"-like looking curves in Figure 5.22 and Figure 5.29. More complex fusion networks may achieve better results, but we were limited by the memory size of our *GPU*. During training the memory for two versions of our *CNN* was allocated. The use of *GPUs* with larger memory, and therefore more complex *CNNs*, may achieve better results. For all sequences we can say that the colorization achieves better results regarding the *PSNR*, the less motion there is. Additionally, if new objects appear and disappear during the sequence, the proposed method is not able to colorize these objects in the right way. The method uses color from similar semantic looking objects. To sum up, our proposed method achieves visually appealing

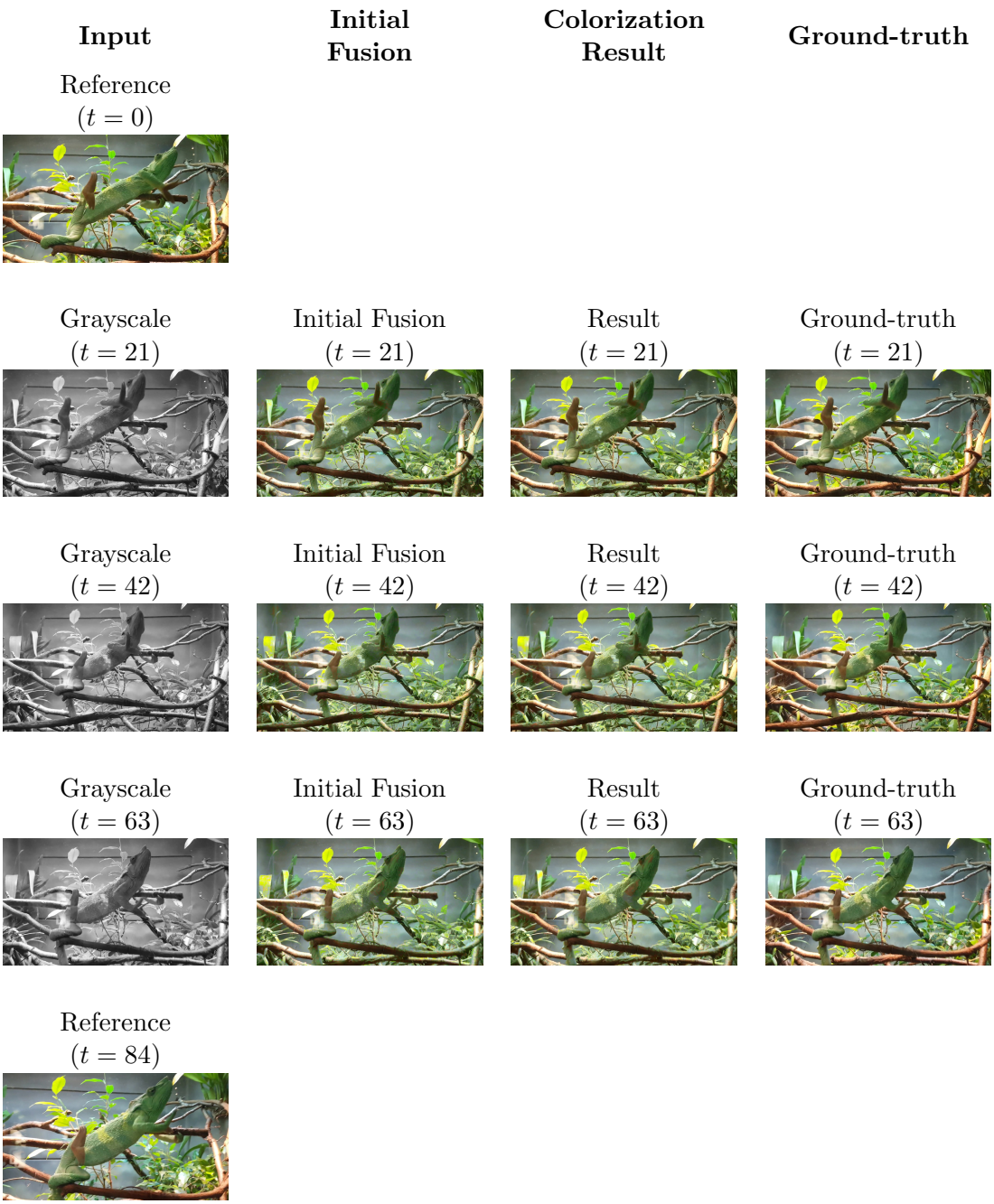


Figure 5.30: Colorization example for sequence chamaleon. The first column shows the input data. Here, the first ($t = 0$) and last ($t = 84$) frames are already colorized. The second column illustrates the results from the initial fusion. Our proposed method, with the U-net residual model and 2 runs, achieves the results in the third column. The last column visualizes the ground-truth color images. Sequence is taken from the test dataset.

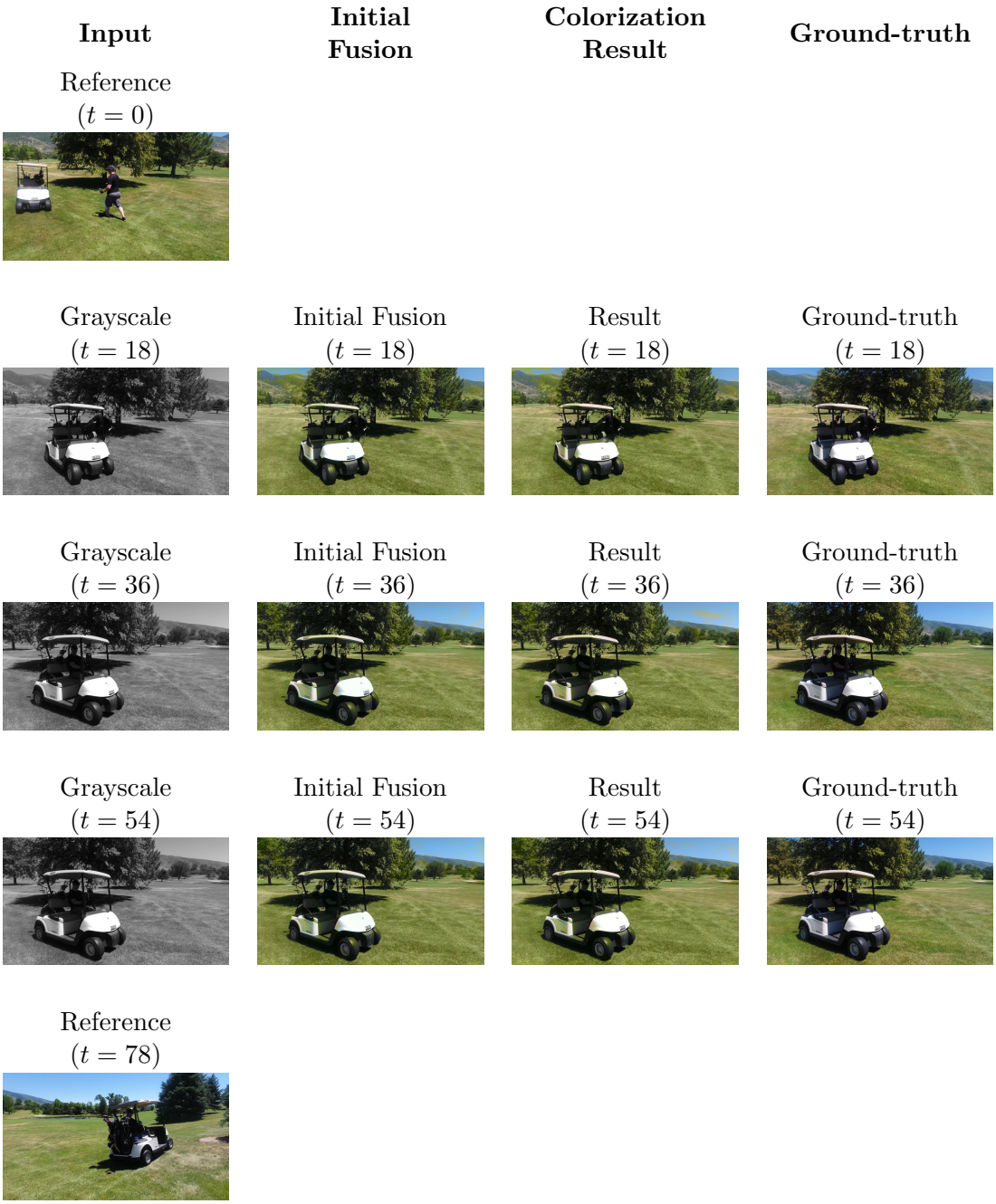


Figure 5.31: Colorization example for sequence golf. The first column shows the input data. Here, the first ($t = 0$) and last ($t = 78$) frames are already colorized. The second column illustrates the results from the initial fusion. Our proposed method, with the U-net residual model and 2 runs, achieves the results in the third column. The last column visualizes the ground-truth color images. Sequence is taken from the test dataset.

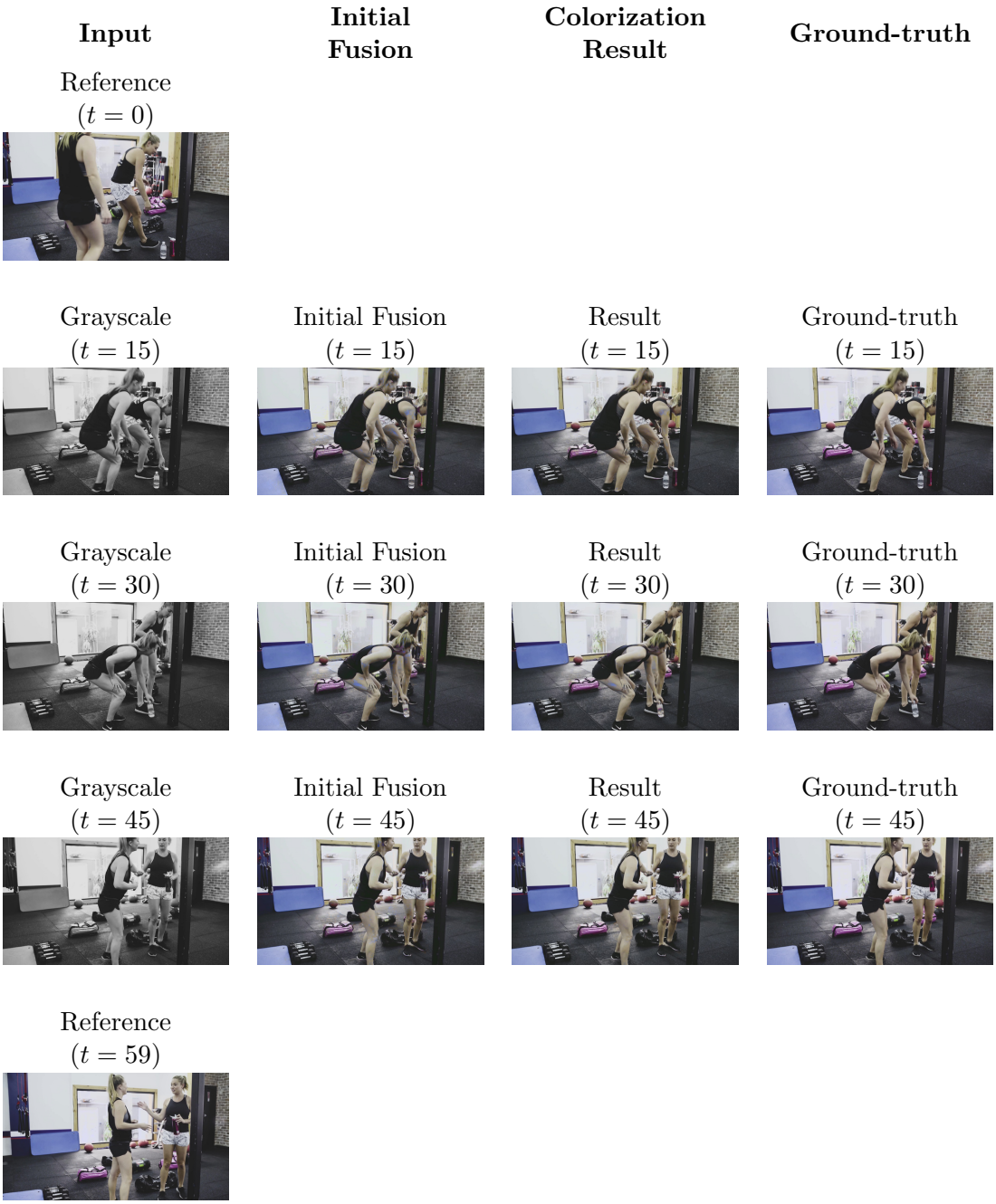


Figure 5.32: Colorization example for sequence gym. The first column shows the input data. Here, the first ($t = 0$) and last ($t = 59$) frames are already colorized. The second column illustrates the results from the initial fusion. Our proposed method, with the U-net residual model and 2 runs, achieves the results in the third column. The last column visualizes the ground-truth color images. Sequence is taken from the test dataset.

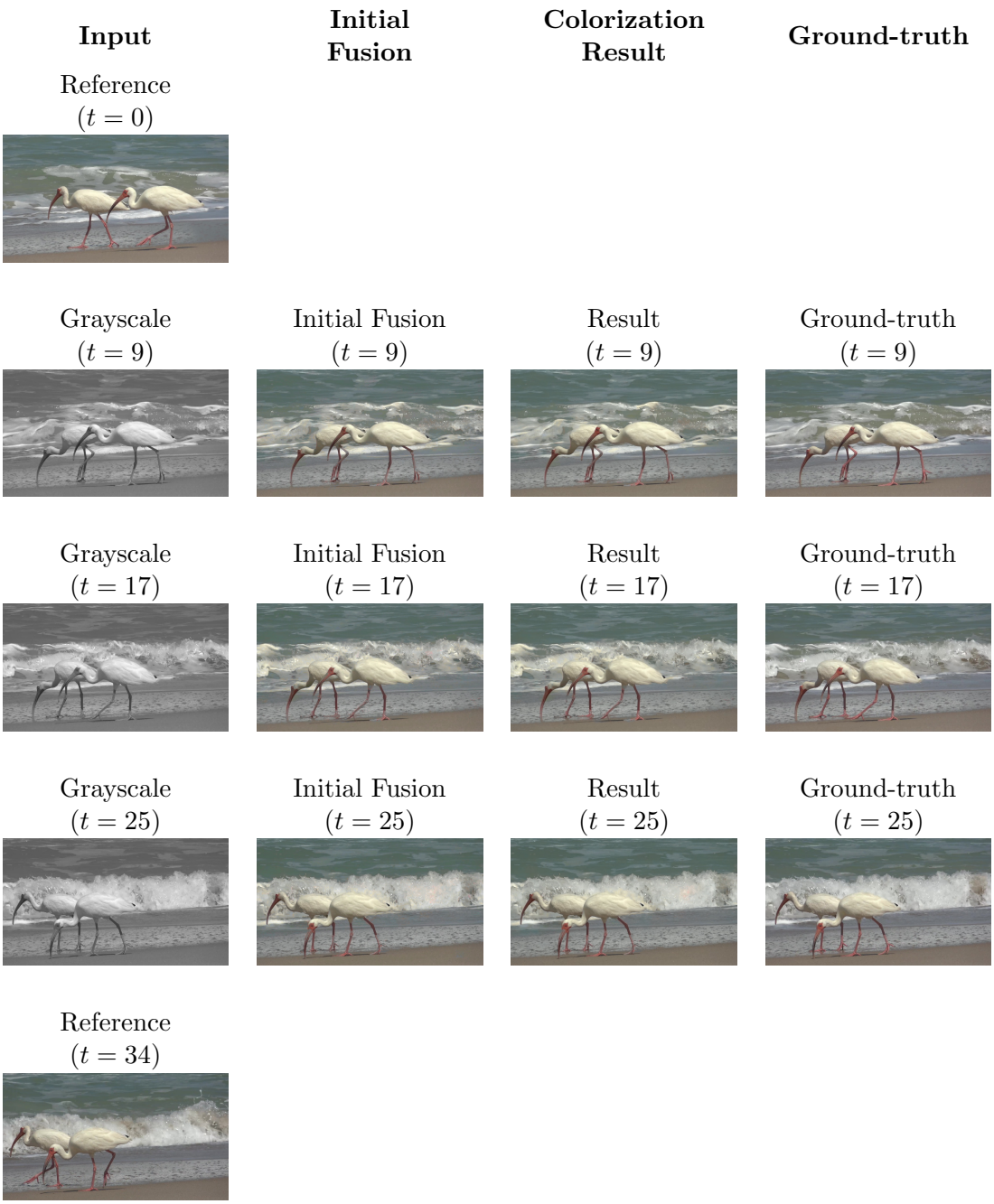


Figure 5.33: Colorization example for sequence ocean-birds. The first column shows the input data. Here, the first ($t = 0$) and last ($t = 34$) frames are already colorized. The second column illustrates the results from the initial fusion. Our proposed method, with the U-net residual model and 2 runs, achieves the results in the third column. The last column visualizes the ground-truth color images. Sequence is taken from the test dataset.

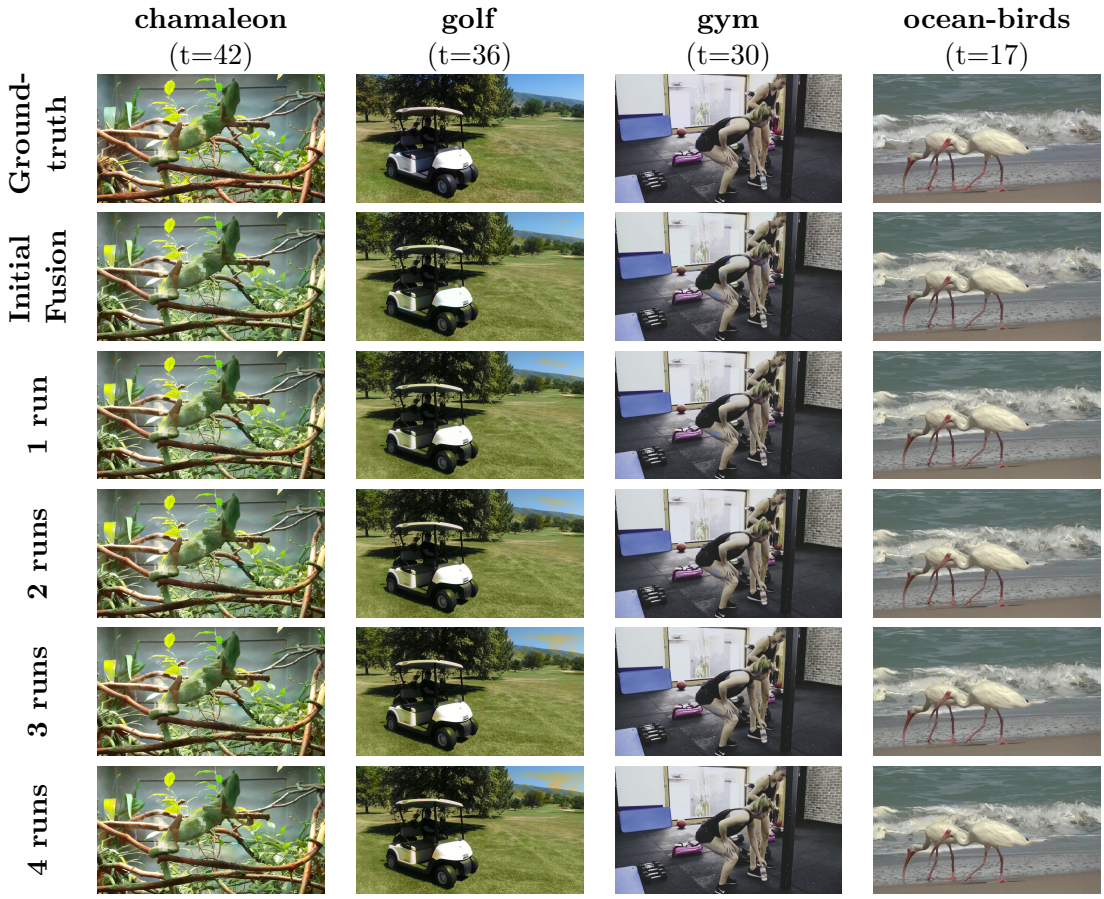


Figure 5.34: Run variation examples. This figure visualizes the colorization of selected frames of various sequences after a certain number of runs. The first row shows the groundt-truth colorizations and the second row the initial fusion. The other rows illustrate the colorization results after different numbers of runs. Each column represents a frame from a different sequence. Sequences: chamaleon, golf, gym, ocean-birds (from left to right). The results were achieved with the U-net residual model trained for 2 runs. Sequences are taken from the test set.

results.

6 Conclusion and Outlook

6.1 Conclusion

In this thesis we defined the problem of image colorization. We analyzed how the loss of information leads to an ill-posed inverse problem. Without additional information the solution of the colorization problem is ambiguous. We stated different methods and gave an literate review of developed approaches to tackle this issue.

We started our considerations for image colorization based on the work of Gupta et al. [9] and added the concept of Optimal Transport (OT). We used *OT* to estimate a bijective assignment to transfer the colors of a reference image to the grayscale image. Based on the initial results we extended this approach to the colorization of grayscale sequences. Here we used the pre-calculated bijective assignment from the first frame of the sequence as a look-up-table to color all of the other images. Due to the high computational effort on the Central Processing Unit (CPU) and not satisfactory results, we decided to switch to a Graphics Processing Unit (GPU) based approach with the use of Convolutional Neural Networks (CNNs).

We used the work of Schaub et al. [33] as a starting point. They used a combination of local and global procedures to achieve a final colorization of a grayscale sequence, given an initial colored first frame. We adapted both sub-strategies. We calculated the local color transfer with the help of an optical flow estimation method and refined the colorization in a local neighborhood. The global color transfer was improved by smoothing the initial coarse colorization with a median filter. We introduced a second reference image as the last frame of the sequence to the colorization pipeline. With this additional color image, we have the opportunity to calculate the colorization in two directions, from the first frame to the last and vice versa. The final colorization is now based on four intermediate results. For each of these sub-results we introduce a confidence map, which is a per-pixel measure that counts for the certainty of the individual colorization. We decided to test various *CNNs* to combine the intermediate results. We improved the final colorization by using a residual estimation method. The fusion *CNN* is trained to compute a change of color.

For this estimation method we introduced an initial fusion, that calculates a colorization based on the sub-results and the confidence maps. The achieved results suggests, that the proposed method is not able to colorize appearing and disappearing objects that are not included in the reference images but apart from that, the results are promising. A recurrent estimation of the colorization with the U-net residual model adds artifacts. The introduction of a confidence for the initial fusion as well as for the results of the learned fusion would probably help to overcome this behavior. Altogether, we introduced a novel approach for coloring grayscale sequences. The proposed method achieves visual pleasing results.

6.2 Future Work

The lack of computational efficiency of the colorization approach with *OT* computed on the *CPU* could be enhanced with a *GPU* implementation. A parallel computation would drastically reduce the run-time of this method. An improvement of the final results may only be achieved by adding extra information from other reference images or using a frame-to-frame colorization.

Additionally, the approach of using *CNNs* may also be improved by using extra color reference images. A recurrent estimation over multiple runs can be enhanced with the introduction of a confidence for the initial and learned fusion. Also, training the model for multiple runs, without the need of computing intermediate results due to memory or time limitations, could enhance the recurrent method. A single network that calculates the whole colorization pipeline would probably reduce computational time. The usage of colorized non-sequence frames would help to colorize objects in the sequence that are not part of the reference images. The colors of newly appearing and disappearing objects can then be transferred from these color images. Another possibility is the direct assignment of color to these objects. Here, the user would have to colorize the objects or make colorful scribbles. The network then should propagate these colors throughout all frames where these objects are included.

A

List of Acronyms

ADAM	Adaptive Moment Estimation
ANN	Artificial Neural Network
API	Application Programming Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
GPU	Graphics Processing Unit
l.s.c.	lower-semicontinuous
LED	Light-Emitting Diode
MSE	Mean Squared Error
OT	Optimal Transport
PDHG	Primal-Dual Hybrid Gradient
PSNR	Peak Signal-To-Noise Ratio
ReLU	Rectified Linear Unit
ROI	Region Of Interest
SIFT	Scale-Invariant Feature Transform
SLIC	Simple Linear Iterative Clustering
SURF	Speeded Up Robust Features
VGG	Visual Geometry Group

B Derivations for Primal-Dual Algorithm

B.1 Proximal Mappings

This section shows the calculation of the proximal mappings, that are needed to calculate the Primal-Dual Hybrid Gradient (PDHG) algorithm Equation (4.18). The needed functions are given in Equation (4.25).

$$\begin{aligned} \text{prox}_{\tau g}(\mathbf{z}) &= \arg\min_{\mathbf{u}} g(\mathbf{u}) + \frac{1}{2\tau}||\mathbf{u}-\mathbf{z}||_2^2 \\ &= \arg\min_{\mathbf{u}} \mathbf{c}^T\mathbf{u} + \delta_{\geq 0}(\mathbf{u}) + \frac{1}{2\tau}||\mathbf{u}-\mathbf{z}||_2^2 \end{aligned} \tag{B.1}$$

Now we make a distinction of cases and evaluate the arg min element-wise:

- for $u_i > 0$:

$$\begin{aligned} \arg\min_{u_i} & c_i u_i + 0 + \frac{1}{2\tau}||u_i - z_i||_2^2 \\ & c_i + \frac{1}{\tau}(u_i - z_1) = 0 \\ & u_i = z_i - \tau c_i > 0 \Rightarrow z_i > \tau c_i \end{aligned} \tag{B.2}$$

- for $u_i = 0$:

$$\begin{aligned} &\Rightarrow c_i + [-\infty, 0] + \frac{1}{\tau}(u_i - z_i) = 0 \\ &\quad u_i = z_i - \tau c_i + \tau[0,\infty] = 0 \Rightarrow z_i = \tau c_i + \tau[-\infty, 0] \\ &\Rightarrow z_i > -\infty \\ &\Rightarrow z_i < \tau c_i \end{aligned} \tag{B.3}$$

- for $u_i < 0$:

$$\Rightarrow \infty \Rightarrow \text{n.def.} \tag{B.4}$$

Combining the previous results, we get:

$$\text{prox}_{\tau g}(z_i) = \begin{cases} 0 \text{ if } z_i > -\infty \text{ and } z_i < \tau c_i \\ z_i - \tau c_i \text{ otherwise} \end{cases}, \tag{B.5}$$

which is calculated element-wise.

The deviation of the other proximal mapping gives:

$$\begin{aligned} \text{prox}_{\sigma f^*}(\mathbf{z}) &= \arg\min_{\mathbf{u}} f^*(\mathbf{u}) + \frac{1}{2\sigma}||\mathbf{u}-\mathbf{z}||_2^2 \\ &= \arg\min_{\mathbf{u}} \langle \mathbf{w}, \mathbf{u} \rangle + \frac{1}{2\sigma}||\mathbf{u}-\mathbf{z}||_2^2 \\ &= \arg\min_{\mathbf{u}} \mathbf{w}^T\mathbf{u} + \frac{1}{2\sigma}||\mathbf{u}-\mathbf{z}||_2^2 \\ &= \mathbf{w} + \frac{1}{\sigma}(\mathbf{u}-\mathbf{z}) = 0 \\ \Rightarrow \mathbf{u} &= \mathbf{z} - \sigma\mathbf{w}. \end{aligned} \tag{B.6}$$

As the final result we get:

$$\text{prox}_{\sigma f^*}(\mathbf{z}) = \mathbf{z} - \sigma\mathbf{w}. \tag{B.7}$$

Bibliography

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. F. S., and Süsstrunk. Slic superpixels. Technical report, 2010. (page 20)

[2] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *Proceedings of the 9th European Conference on Computer Vision*, 2006. (page 20)

[3] W. Burger and M. J. Burge. *Digital image processing: an algorithmic introduction using Java*. Springer, 2016. (page 7, 12)

[4] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011. (page 22)

[5] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, April 2018. (page 38, 52)

[6] Z. Cheng, Q. Yang, and B. Sheng. Deep colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015. (page 4)

[7] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004. (page 25)

[8] O. Frigo, N. Sabater, V. Demoulin, and P. Hellier. Optimal transportation for example-guided color transfer. In *Proceedings of the Asian Conference on Computer Vision*, 2014. (page 4)

[9] R. K. Gupta, A. Y. S. Chia, D. Rajan, E. S. Ng, and H. Zhiyong. Image colorization using similar images. In *Proceedings of the 20th ACM International Conference on Multimedia*, 2012. (page 4, 5, 17, 25, 26, 27, 69)

[10] J. Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton university bulletin*, pages 49–52, 1902. (page 14)

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016. (page 38, 39, 52)

[12] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proceedings of the 28th annual Conference on Computer Graphics and Interactive Techniques*, 2001. (page 19)

[13] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. (page 48, 49)

[14] 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 the 13th annual ACM international conference on Multimedia*, 2005. (page 4)

[15] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (TOG)*, 35(4):110, 2016. (page 4)

[16] S. Iizuka, E. Simo-Serra, K. Kikuchi, K. Yamaguchi, T. Kobayashi, Y. Shinya, and T. Suzuki. Deepremaster: Temporal source-reference attention networks for comprehensive video enhancement. *ACM Transactions on Graphics*, 38, 2019. (page 6)

[17] I. F. Jafar and G. M. A. Sukkar. A novel coloring framework for grayscale images. In *Proceedings of the International Conference on Multimedia Computing and Information Technology*, 2010. (page 4)

[18] V. Jampani, R. Gadde, and P. V. Gehler. Video propagation networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2017. (page 5)

[19] L. Kantorovich. On the translocation of masses. In *Proceedings of the National Academy of Sciences, Comptes Rendus (Doklady) de l'Academie des Sciences de l'URSS*, 1942. (page 21)

[20] H. B. Kekre and S. D. Thepade. Color traits transfer to grayscale images. In *Proceedings of the International Conference on Emerging Trends in Engineering and Technology*, 2008. (page 4)

[21] K. I. Laws. Texture energy measures. In *Proceedings of the Image understanding workshop*, 1979. (page 19)

[22] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics (TOG)*, 23(3):689–694, 2004. (page 4, 24, 30)

[23] W. Li, F. Viola, J. Starck, G. J. Brostow, and N. D. F. Campbell. Roto++: Accelerating professional rotoscoping using shape manifolds. *ACM Transactions on Graphics (TOG)*, 35(4):1–15, 2016. (page 51)

[24] X. Liu, L. Wan, Y. Qu, T. T. Wong, S. Lin, C. S. Leung, and P. A. Heng. Intrinsic colorization. In *Proceedings of the ACM SIGGRAPH Asia*, 2008. (page 4)

[25] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, 1996. (page 20)

[26] G. Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*, 1781. (page 21)

[27] J. R. Movellan. Tutorial on gabor filters. `https://pdfs.semanticscholar.org/bae5/bae884633e7da2c1ec75d158f8849d2183d3.pdf`, 2002. [Online; accessed 4^{th} May 2020]. (page 20)

[28] M. A. Nielsen. *Neural networks and deep learning*. Springer, 2018. (page 33)

[29] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. (page 51)

[30] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. V. Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. (page 51)

[31] J. Rabin and N. Papadakis. Non-convex relaxation of optimal transport for color transfer between images. In *Proceedings of the International Conference on Geometric Science of Information*, 2015. (page 4)

[32] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, 2015. (page 49, 50)

[33] S. Schaub-Meyer. *Video Frame Interpolation and Editing with Implicit Motion Estimation*. PhD thesis, ETH Zurich, 2018. (page 5, 33, 34, 60, 69)

[34] R. Sinkhorn and P. Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967. (page 24)

[35] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. (page 44, 45)

[36] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *European Conference on Computer Vision*, 2010. (page 42)

[37] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2011. (page 7, 37)

[38] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. (page 49)

[39] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy. Tracking emerges by colorizing videos. In *Proceedings of the European Conference on Computer Vision*, pages 391–408, 2018. (page 5)

[40] T. Welsh, M. Ashikhmin, and K. Mueller. Transferring color to greyscale images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002. (page 4)

[41] S. Xia, J. Liu, Y. Fang, W. Yang, and Z. Guo. Robust and automatic video colorization via multiframe reordering refinement. In *Proceedings of the IEEE International Conference on Image Processing*, pages 4017–4021. IEEE, 2016. (page 5)

[42] L. Yatziv and G. Sapiro. Fast image and video colorization using chrominance blending. *IEEE transactions on image processing*, 15(5):1120–1129, 2006. (page 4)

[43] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. (page 48)

[44] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *Proceedings of the European Conference on Computer Vision*, 2016. (page 4, 5)

[45] R. Zhang, J. Y. Zhu, P. Isola, X. Geng, and A. S. Lin. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics*, 36(4):119, 2017. (page 4)