RWTH AACHEN UNIVERSITY

Diese Arbeit wurde vorgelegt am
Lehr- und Forschungsgebiet Informatik 8 (Computer Vision)
Fakultät für Mathematik, Informatik und Naturwissenschaften
Prof. Dr. Bastian Leibe

Master Thesis

Digital color reconstruction of a historical film format

vorgelegt von

Chenfei Fan

Matrikelnummer: 390189

2020-01-08

Erstgutachter: Prof. Dr. Bastian Leibe
Zweitgutachter: Prof. Dr. Konrad Schindler

Eidesstattliche Versicherung

Chenfei Fan	390189
Name	Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Masterarbeit mit dem Titel

Digital color reconstruction of a historical film format

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen, 2020-01-08	
Ort, Datum	Unterschrift

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten dementsprechend.

Die vorstehende Belehrung habe ich zur Kentnis genommen:

Aachen, 2020-01-08	
Ort, Datum	Unterschrift

Abstract

The lenticular film is a unique color film technique for amateur filmmaking. The most popular product was released by Kodak in 1928, under the name `Kodacolor`. Color films produced with this technique provide precious records of everyday life in the first half of the twentieth century. The film's surface is engraved with a microscopic array of cylindrical lenses (the lenticules). Within each lenticule, the color of a vertical slice of the image is spatially encoded in grayscale. It requires a specific projection set-up to regain the color images on the screen. However, we can also reconstruct the colors from a high-resolution scan of the film material by digital processing.

As a legacy of the work done in the framework of the doLCE project (University of Basel, 2012), the University of Zurich has a software solution to reconstruct lenticular colors. The first step of color reconstruction is a precise recognition of the boundaries between the lenticules. Its precision suffers from common imperfections on the film, such as misalignment, defocusing, and noise. Since lenticules are very narrow, this often leads to false or inconsistent colors in reconstruction.

Inspired by deep learning approaches for a wide range of edge detection tasks, at the EcoVision Lab, ETH Zurich, we develop a practical lenticule boundary detector using the U-Net architecture. The training dataset is composed of successful color reconstruction results of the doLCE software. Since the training samples are limited, we use data augmentation methods, such as affine transformation and random erasing, to extend the dataset.

To get even sharper lenticule boundaries, we introduce an iterative approach to refine the recognition results based on the constraint that the width of the lenticules is approximately constant within one image. Without losing much color information, this increases color consistency and speeds up the process of color reconstruction.

Preliminary results of the color reconstruction of the analyzed lenticular films show that the proposed deep learning method improves the result provided by

doLCE, reconstructing more truthful and consistent colors.

Keywords: lenticular film, color reconstruction, edge detection

Acknowledgement

I would like to thank Dr. Giorgio Trumpy of University of Zurich for providing me with knowledge in lenticular film and physics about film colors, and preparing the large dataset for training the network. And I would like to thank Dr. Stefano D'Aronco for the thorough supervision and assistance during the thesis which helped me to gain knowledge in computer vision and get familiar with Python programming, and the great ideas in color reconstruction. Also thanks to Dr. Joakim Reuteler, Dr. Rudolf Gschwind and AMIA Open Source for the fundamental work on doLCE.

In addition I would like to thank Dr. Jan Dirk Wegner for looking over the project, and Prof. Konrad Schindler of the Institute of Geodesy and Photogrammetry, ETH Zurich, for providing the resource and funding for my thesis and living in Zurich. Also thanks to Prof. Bastian Leibe for the recommendation. Finally, I would thank my family for the great support during my study.

Contents

1 Introduction

The basic idea of the lenticular film was developed by Raphaël Liesegang in 1896, and applied to still photography by Rodolphe Berthon in 1908 [Flu12].

The first commercial lenticular film was marketed by the Eastman Kodak Company in 1928 under the name `Kodacolor` and provided amateur filmmakers with the chance of recording everyday life, which serves as precious record of life in the first half of the twentieth century.

As is shown in Figure 1.1 from [Wei33], when light emitted from the light source goes through the lens, it is focused on a very small area on the film, where the surface of the film is engraved with a microscopic array of cylindrical lenses (the lenticules), which have curvatures such that light going through any small area on the filter will be focused on a corresponding small area in the silver emulsion. These tiny lenses function as color filters that allow different color components to pass at different positions. As the diagram sketched by [CS28] illustrates, in the top one-third of the lens, only the red component of the light is allowed to pass, while from the center third, only the green components, and for the low third, only the blue components, such that records of the red, green, and blue exposures are obtained separately in each lens.

The film can be projected on screen by reversal. As is shown in Figure 1.1, light is projected through the film, a projection lens, and a color filter, in the same manner as the process in taking the film, reversing the optical path of the light from the light source to the lenticules. The engraved lenses focus the light which are coded in intensities corresponding to the red, green, and blue exposures onto the red, green, and blue sections of the filter shown in Figure 1.3. The projection lens focuses the film image onto a screen to obtain a colored picture.

Figure 1.4 show another illustration of capturing and projection of the lenticular film.

It used to require a specific projection set-up to regain the color images on the screen, but the work done in the framework of the doLCE project [R$^+$14] proposes a software solution to reconstruct colors. The lenticular film is first scanned and digitized into gray-scale images and then fed into the software as

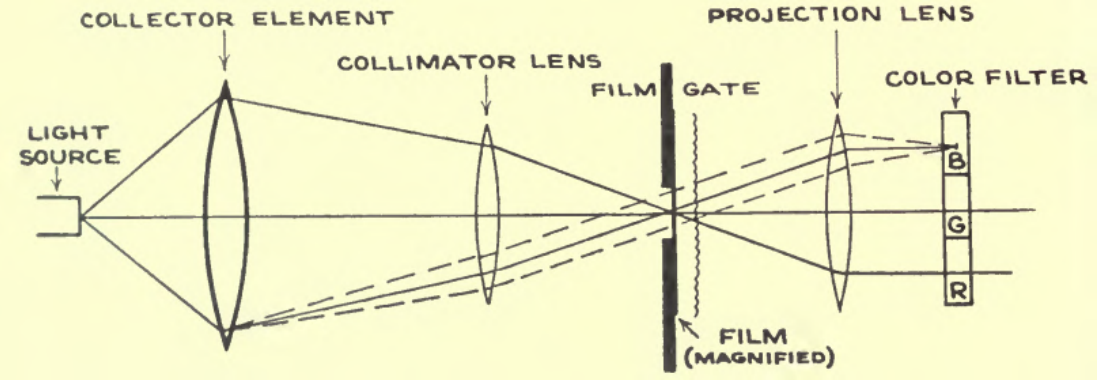


Figure 1.1: Diagram of projection optical system for lenticular color-film [CMW37]

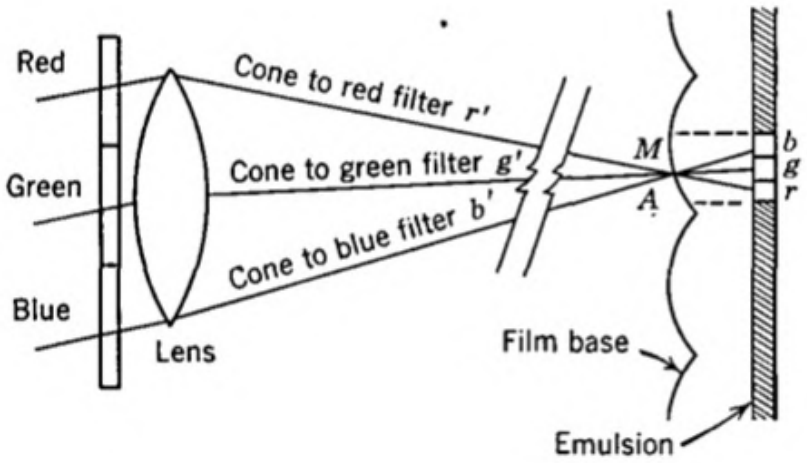


Figure 1.2: Diagram tracing light-beam paths in lenticular processes [CS28]

the input. Figure 1.5 shows the two steps of the doLCE software: first, it localizes lenticules by separating them with vertical lines. Then it convert gray levels in each lenticule into color.

Comparing doLCE's reconstruction with the original analog color reconstruction, the result is satisfactory for many lenticular images, but it could also fail to reconstruct truthful colors when the condition of th lenticular film is bad, for example, when the lenticules are not perfectly vertical, and when there are blemishes or dark areas on the image.

These failures in doLCE's color reconstruction are assumed to result from the algorithm it uses to detect boundaries of the lenticules. It sums all the color intensities along the height dimension of the image and detect the peaks along the width dimension to find the locations of the boundaries, but it first assumes that lenticules have perfectly vertical boundaries - not tilted, nor curved or distorted, which is only approximate, but often not the exact case in many projection settings in real life. What is more, if there are vast dark areas in the image, summing

Figure 1.3: color filter used in Kodacolor [CS28]

along each vertical line does not always guarantee correct peaks. Given that the lenticules are microscopic, such small errors in boundary detection will lead to visible artifacts or inconsistencies in colors.

We have investigated into algorithms like canny edge detector and Hough transform to detect these lenticule patterns, but they do not provide satisfactory results for this given problem. As a result, inspired by deep learning approaches, at the EcoVision Lab, ETH Zurich, we develop a practical lenticule boundary detector using a deep learning architecture, which will be discussed in the following chapters.

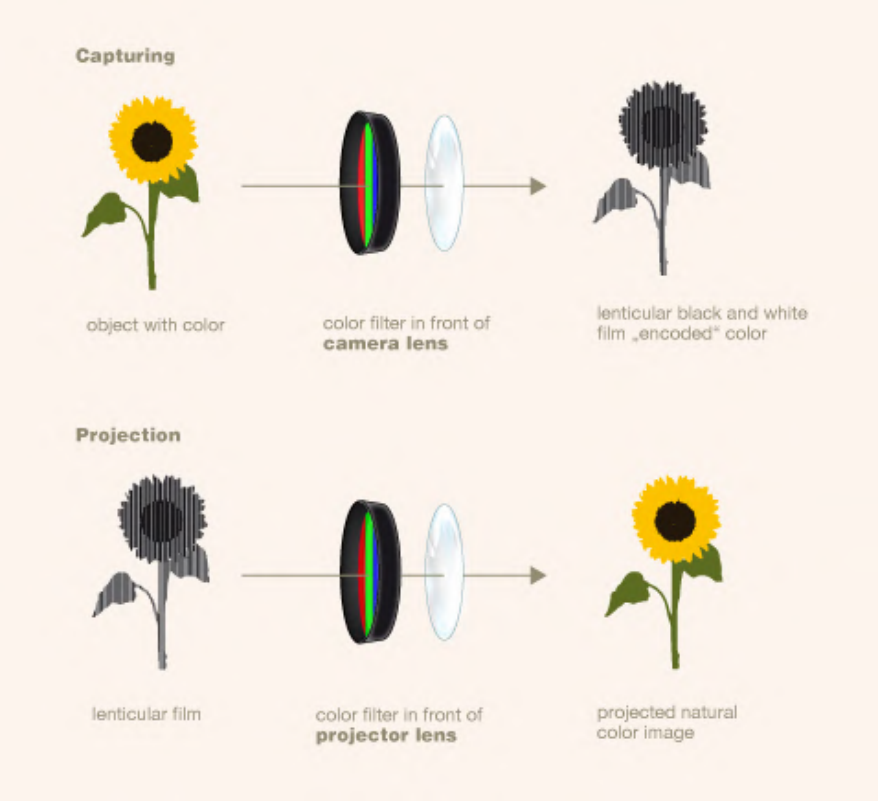


Figure 1.4: Principle of capturing and projecting lenticular film. Credit: Joakim Reuteler and Rudolf Gschwind, Digital Humanities Lab, University of Basel, Switzerland. Illustration by Sarah Steinbacher, Multimedia & E-Learning-Services, University of Zurich.

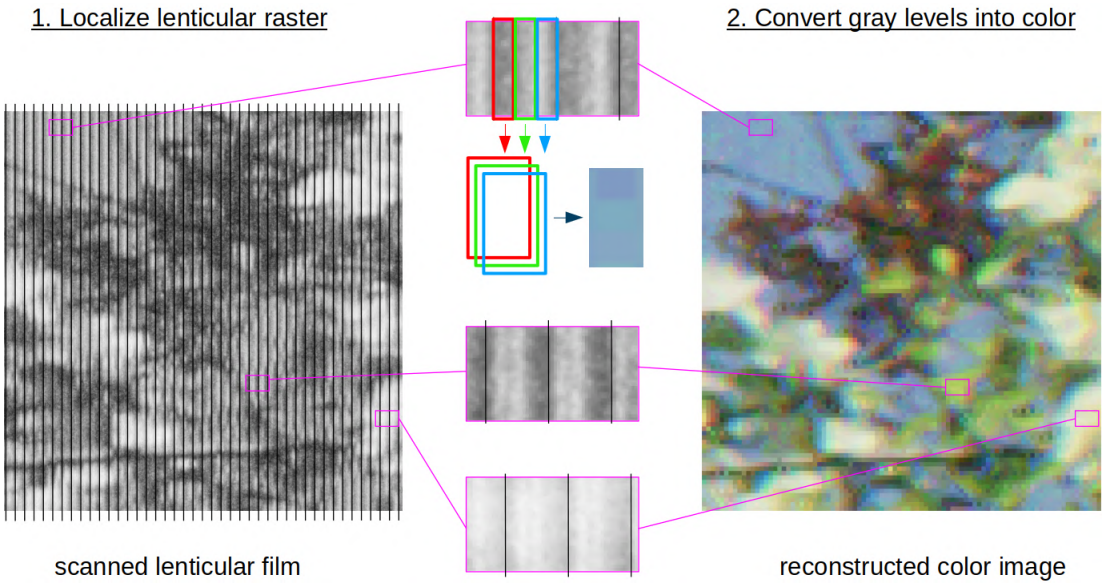


Figure 1.5: 2 steps of the doLCE software [R+14]

Figure 1.6: comparing doLCE's reconstruction with the original analog color reconstruction. Credit: D. Pfluger and G. Trumpy

2 Lenticule Detection

The first and most critical step of color reconstruction for digitized lenticular film images is detecting the precise location of the lenticules. Due to large gradients on the lenticule boundaries, this task is expected to be solved by edge detection techniques which have been applied to other similar problems. However, a local gradient operator would not be the best approach because lenticule boundaries in dark areas of images are hard to find, e.g. Figure 2.1.

Figure 2.1: a picture in the doLCE dataset with large dark areas where lenticule boundaries are hard to recognize

This is where deep learning comes into play. Because lenticule boundaries are vertically continuous, we can guess their unknown parts by extending them from clearer parts of the image. This requires not only local or neighboring gradients but also information from far upper or lower pixels.

2.1 System design

In the first step of processing an image, first we cut the image into slices and feed them to our designed network by batches as inputs. Then the outputs of the network are horizontally concatenated into a whole image. Figure 2.2 illustrates the pipeline of image processing. The result is a probability map of lenticule boundaries, which is later used for color reconstruction.

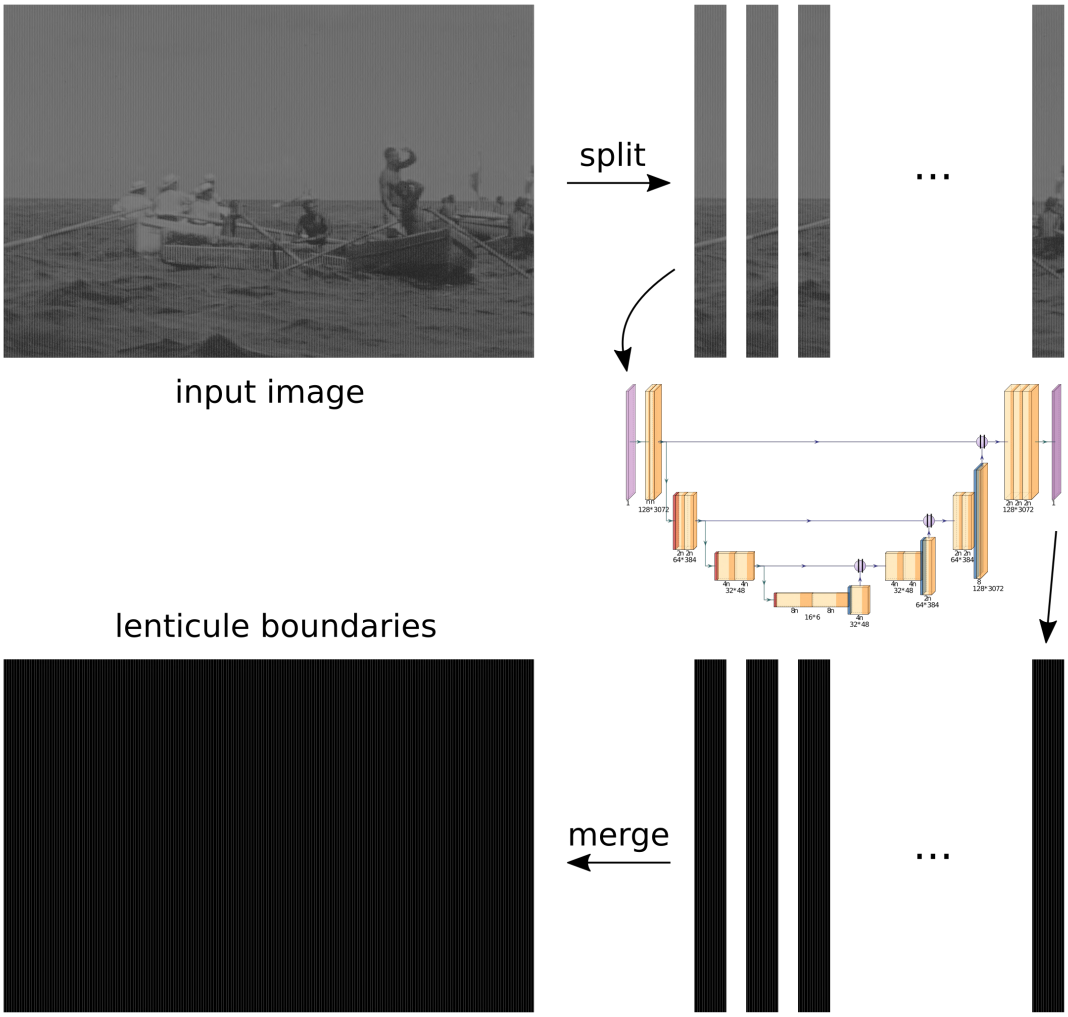


Figure 2.2: The image processing pipeline from the input image to the lenticule boundaries

2.1.1 Model architecture

The U-Net [RFB15] is a convolutional network architecture introduced for fast and precise biomedical image segmentation. It has also achieved promising results on a wide range of segmentation tasks. Unlike the traditional sliding-window approach which predicts the class label of each pixel locally, it consists of a usual contracting path to capture larger context and a expanding path that enables precise localization. The contracting path alone can be applied to classification tasks and extract features. In the expanding path, pooling operators are replaced by upsampling operators to increase the resolution of features, and the output of each layer is concatenated with the features of the same resolution in the contracting path, forming a symmetric U-shape. Since the U-Net is capable of capturing very large context, we implement a modified U-Net to train on the samples generated from the doLCE dataset.

Our network architecture is illustrated in Figure 2.3.

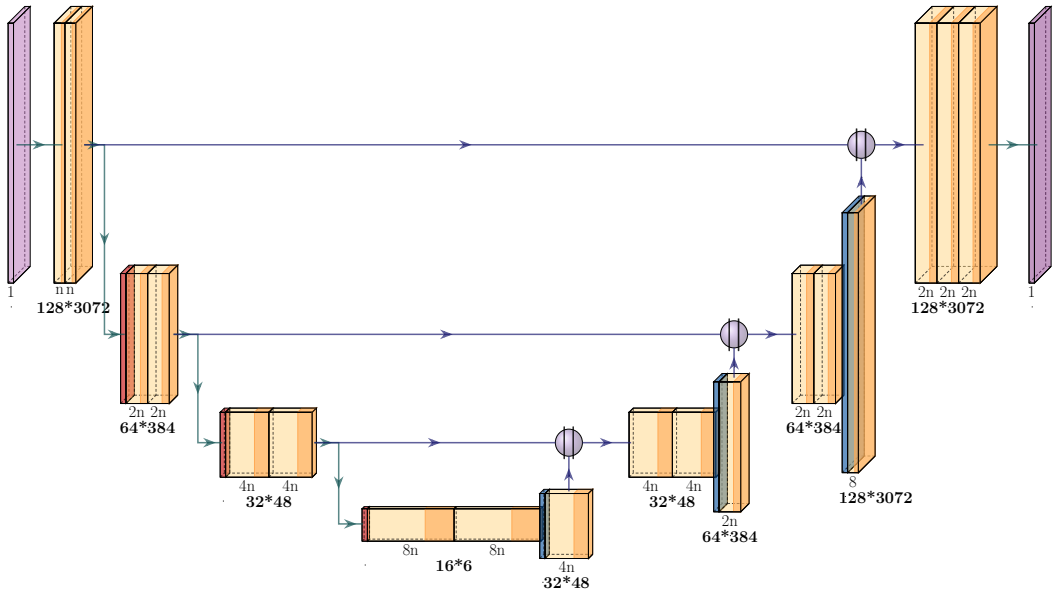
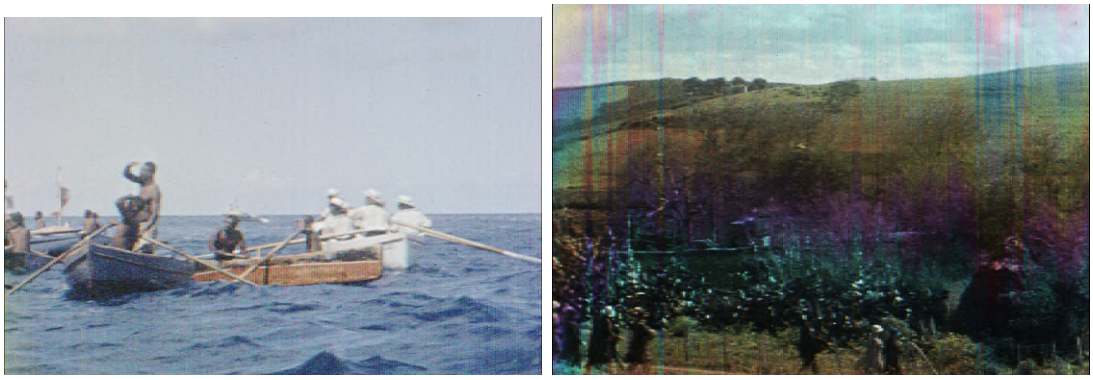


Figure 2.3: U-Net architecture trained on samples of size 128*3072 which are generated from the doLCE dataset

Dataset

The dataset for training and validation of our U-Net is composed of 2010 samples selected from successful reconstructions of doLCE project. The assessment of reconstructed images is subjective but in general it favors images with consistent colors. Figure 2.4 shows examples of successful / unsuccessful reconstructions of doLCE. Samples like Figure 2.4a are included in the dataset although there is

still some greenish hue at its bottom left corner, as most colors look consistent and appear truthful, while those like Figure 2.4b are removed because there are serious color inconsistencies and purple trees which are clearly far from real life.



(a) successful reconstruction by doLCE (b) unsuccessful reconstruction by doLCE

Figure 2.4: Examples of successful / unsuccessful reconstructions of doLCE.

As Figure 2.5 shows, each sample consists of the input image, which is the scanned lenticular film in gray scale, and the target image, which is the ground truth labels of lenticule boundaries. These ground truths are intermediate results of doLCE's pipeline. They are considered true based on the visual consistency of color reconstruction, so they are expected to be less accurate than manual labeling. However, given the large dataset, we still expect that most ground truth labels are accurate enough to draw out the outliers.

(a) the input image (b) the target image (ground truth labels)

Figure 2.5: Input and target images of a sample

Although the U-Net has a capacity for bigger images, lenticular film scans are still too large with approximately 4000 pixels in width and 3000 pixels in height, which makes training on whole images too costly. Furthermore, our database is very limited. But since we mainly consider information sharing in the vertical

direction, while lenticule patterns are horizontally repeated, it is possible to cut each image into (tall and thin) slices with similar patterns, and train the network on those slices instead of the whole images. This method reduces computational resources and enlarges the training dataset.

2.1.2 Data augmentation

Even with training on slices of images, the dataset is still so small that overfitting is likely to degrade the performance of the trained network. Moreover, the dataset only contains samples that are considered successful reconstructions by doLCE. Given the working principle of doLCE, they are more likely to come from images with clear and vertical lenticule boundaries. Images with obscured or irregular lenticule boundaries, therefore, are much less likely to appear in the dataset. However, our network is designed to cope with these "bad" conditions on lenticular films. We need more samples with both obscured or irregular lenticule boundaries and truthful detection results.

To extend the dataset with our desired samples, we need to apply data augmentation. For each sample where the input and the target come from the same slice of a lenticular film scan and its corresponding lenticule boundaries detected by doLCE, the input image undergoes a random affine transformation and a random erasing that occludes a part of the image, while the target image only undergoes the same affine transformation as the input, without random erasing. This is done by the `PyTorch` library, `torchvision.transforms`, and an example of the results is shown in Figure 2.6.

2.2 Training and evaluation

2.2.1 Loss

The detection of lenticule boundaries is a binary image segmentation problem, or to say a binary pixel-wise classification problem which determines whether each pixel is a lenticule boundary pixel or not. The Binary Cross Entropy (BCE) Loss is commonly used for this type of tasks. Since the truth labels are either 0 or 1, the output of our U-Net should also be scaled to [0, 1], so we apply a Sigmoid (logistic) function before computing the loss. For better numerical stability, we use the `PyTorch` class `torch.nn.BCEWithLogitsLoss` which combines the Sigmoid and the BCE Loss into one operation.

The idea behind weighted loss is to emphasis on rarely seen labels in highly unbalanced training sets. Although it performs well for many edge detection algorithms such as Holistically-Nested Edge Detection (HED) [XT15], Richer Convolutional Features for Edge Detection (RCF) [LCH+17], and other recent works such as [DSL+18], from our experiment results, it does not work better than a vanilla BCE loss. By looking at the validation results we speculate that

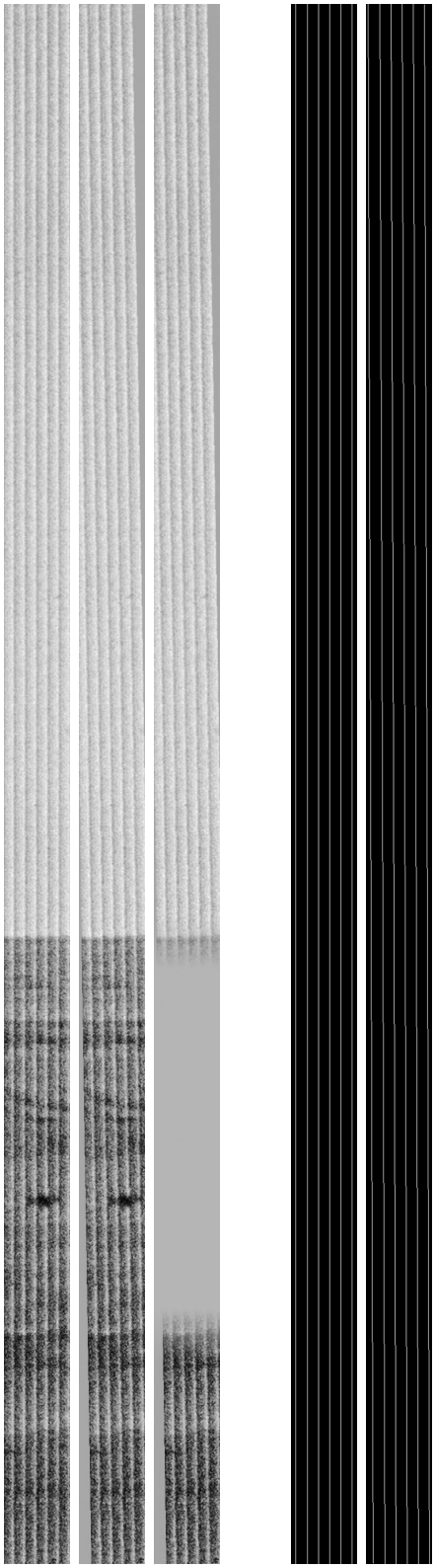
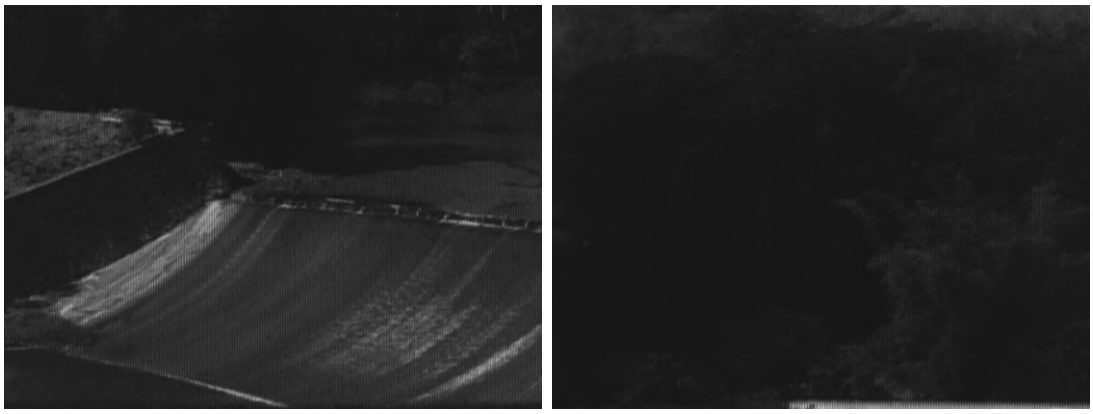


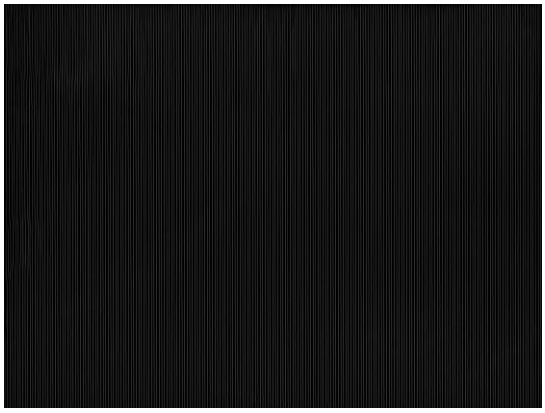
Figure 2.6: Data augmentation: (a) the original slice of input (b) input after affine transformation (c) input after random erasing
(d) the original target (e) target after affine transformation

if we weigh more on true labels, the punishment on false positives, which refers to noises in background, is not sufficient.

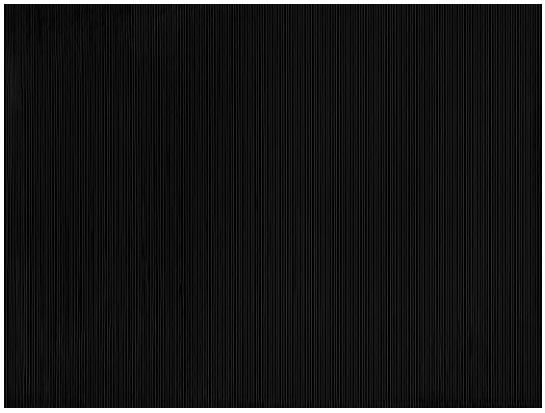


(a) input image 1

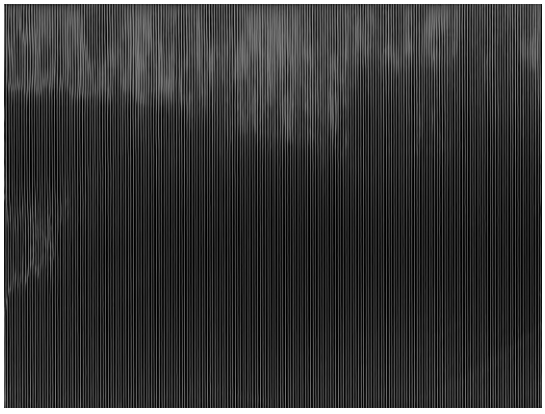
(b) input image 2



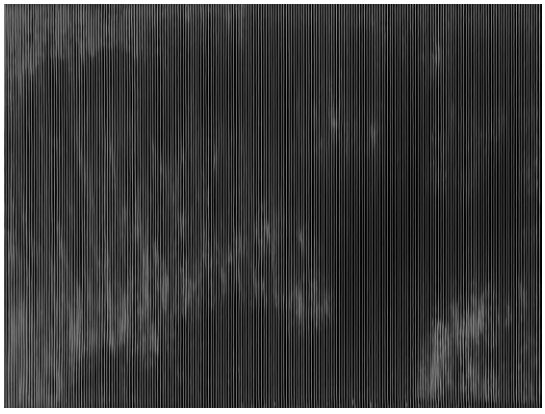
(c) output image 1(trained with BCE loss)



(d) output image 2(trained with BCE loss)



(e) output image 1(trained with weighted BCE loss)



(f) output image 2(trained with weighted BCE loss

Figure 2.7: Input and target images of a sample

As a result, we use BCE Loss without weights, which trades off recall with precision. As is shown in Figure 2.7, lenticule boundaries of the validation result appear to be sharper but also dimmer without training with weighted loss. Although we can further fine-tune the weights, as is shown in the next chapter, a constrained optimization of the lenticule boundaries can compensate the noises, minimizing the impact of different loss functions.

It is worthy noting that only comparing the value of the loss does not help in choosing the loss function because they are calculated by different criterion. Comparing the output images of the validation sets is always necessary.

2.2.2 Metrics

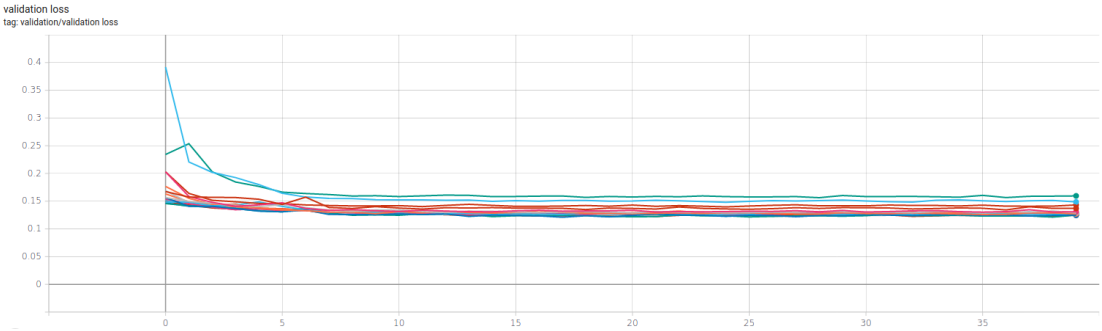
By deciding on a vanilla BCE Loss function without weights, we have agreed that precision and recall are equally important for our model. As a result, we monitor them separately as evaluation metrics, and also calculate dice coefficients [SLV+17], which is commonly used for highly unbalanced segmentations, respectively.

2.2.3 Experiments

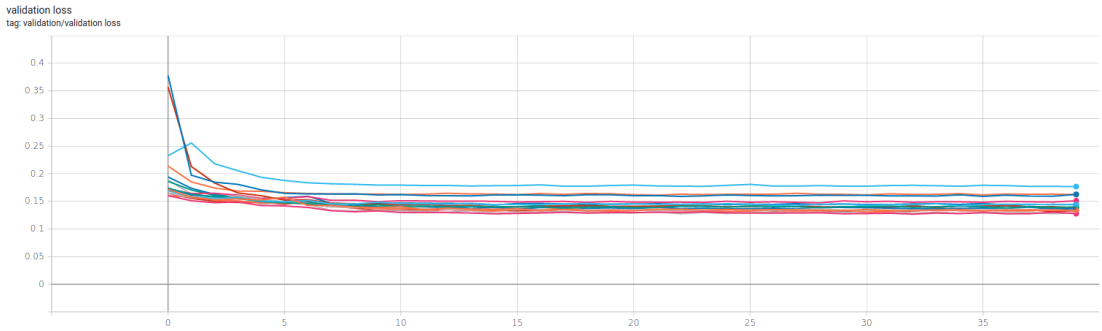
We implement our network using `PyTorch` and train it on a single NVIDIA GeForce GTX 1080 GPU on the Leonhard cluster provided by ETH Zurich. We conduct a grid search to find the best setting of parameters. We choose the number of features in the first convolutional layer ("n" in Figure 2.3) from {4, 8, 16, 32}, the initial learning rate from {0.01, 0.001}, the optimizer from Adam (weight_decay=0) and SGD (momentum=0.99), and whether or not to apply random erasing in data augmentation. By this grid search, we have 32 experiments with different parameter sets. In each experiment, the network is trained for 40 epochs. The learning rate is multiplied by 0.1 after every 7 epochs. The validation losses of these experiments are plotted using TensorBoard. Figure 2.8a and Figure 2.8b show the curves of validation loss of experiments without random erasing and with random erasing, respectively.

By comparison of validation metrics together with visual outputs, we find that the recall is usually satisfactory (for most experiments over 0.99), while the precision, which is only 0.17 in the best case, would determine the sharpness of lenticule boundaries. Therefore we favor the model parameter sets yielding larger precision.

Contrary to expectation, the curves of loss and metrics do not show a clear tendency with different parameter sets. We choose the parameter set with the largest dice coefficient (and also the largest precision) shown in Figure 2.9, with random erasing in data augmentation, 16 features in the first convolutional layer, an initial learning rate of 0.001, SGD as optimizer, and use this fine-tuned U-Net in Chapter 3 for color reconstruction.



(a) validation loss of experiments without random erasing



(b) validation loss of experiments with random erasing

Figure 2.8: curves of validation loss plotted in TensorBoard

with random erasing:

optimizer:	learning rate:	adam	sgd	learning rate:	adam	sgd	Init-features:
precision	0.01	0.16961445	0.14389413	0.001	0.13059174	0.10063663	4
recall		0.99403111	0.99083705		0.99653035	0.99436264	
dice coefficient		0.28921658	0.25056795		0.23031062	0.1821939	
precision		0.15474041	0.16733625		0.1694409	0.14186837	8
recall		0.99348474	0.99325436		0.99512131	0.99233821	
dice coefficient		0.26657555	0.28577342		0.28910351	0.24750613	
precision		0.1558541	0.13981607		0.15591188	0.17219637	16
recall		0.99392158	0.87858448		0.99478857	0.99603115	
dice coefficient		0.26832177	0.23959842		0.26834132	**0.2931336**	
precision		0.15902256	0.16585552		0.1690813	0.1711038	32
recall		0.9930488	0.99436231		0.99534507	0.99523297	
dice coefficient		0.27320454	0.28374186		0.28853501	**0.2913407**	

without random erasing:

optimizer:	learning rate:	adam	sgd	learning rate:	adam	sgd	Init-features:
precision	0.01	0.15088931	0.13804768	0.001	0.13804768	0.10608004	4
recall		0.99416646	0.99366472		0.99366472	0.9935771	
dice coefficient		0.26078929	0.2415368		0.2415368	0.19122045	
precision		0.15794376	0.16315652		0.16503353	0.14472263	8
recall		0.9933806	0.99336238		0.99598985	0.99191025	
dice coefficient		0.2712274	0.27941944		0.2823169	0.2516257	
precision		0.14472263	0.16276811		0.16533037	0.15507221	16
recall		0.99191025	0.99402496		0.99605833	0.99548852	
dice coefficient		0.2516257	0.27878165		0.28280527	0.26702268	
precision		0.15507221	0.15845378		0.1653144	0.16278717	32
recall		0.99548852	0.99557339		0.99667689	0.99565253	
dice coefficient		0.26702268	0.27244253		0.28269157	0.27887966	

Figure 2.9: metrics of the validation set from different parameter sets

3 Color Reconstruction

We speculate that the U-Net architecture proposed in the last chapter provides more truthful lenticule boundaries, but no conclusion can be drawn until we see the results of the color reconstruction.

It seems that given the output images of lenticule boundaries, we can already reconstruct color inside each lenticule by following the physics of lenticular film explained in Chapter 1. Having a closer look at Figure 1.5, the color of each pixel depends on the only one row and the only one lenticule it belongs to, so the color reconstruction can be done by an iteration over each pixel to extract the color pixel by pixel, though apparently this process could be further optimized.

However, details in implementation are still left to be discussed in this chapter.

3.1 A straightforward implementation

Since color reconstruction is always localized in rows, i.e. we do not consider a pixel's color also depends on its neighbors in adjacent rows, we can reconstruct color by iterating the same method over rows, where in each iteration we work on a single row of image with 1D "flat" lenticules.

For a straightforward implementation, the lenticule boundaries are defined such that, for one row of the image, if $x \in [x_1, x_2]$ marks one lenticule, then the lenticule to its right needs to satisfy $x \in [x_2 + 1, x_3]$. Between each two lenticules the boundary is sharp, with no gap or overlap.

However, considering the output of the U-Net as a heat map showing the probability of each pixel belonging to the boundary class, we still need a non-maximum suppression method where only pixels with local maximum probabilities would stand out to mark the boundaries of two lenticules. Here we use the method `scipy.signal.find_peaks` to extract these local maxima. For a better and more stable performance, the required minimal horizontal distance is set to 7.

After extraction of "sharp" boundaries, the problem is reduced to an interpolation task inside each lenticule given its left and right boundaries. For ex-

ample, if $x \in [x_1, x_2]$ marks one lenticule, then its three channels are represented by $x \in [x_1, x_1 + (x_2 - x_1)/3]$, $x \in (x_1 + (x_2 - x_1)/3, x_2 - (x_2 - x_1)/3]$, $x \in (x_2 - (x_2 - x_1)/3, x_2]$, respectively (conversion from floats to integer pixel values are omitted for notation simplicity). Then for each color channel, we can use, for example, the `scipy.interpolate.interp1d` method to map the color intensities in one-third of the whole interval to the whole interval and interpolate the pixels in between. As is visualized in Figure 1.5, this interpolation stretches the three color channels to the full size in width, and then stack them in depth.

Examples of color reconstruction by the straightforward implementation are shown in Figure 3.1. Figure 3.1a is successfully reconstructed without artifacts, while Figure 3.1b has some green stripes on its left part, which can be further improved.

(a) a successful reconstruction (b) an unsatisfactory reconstruction

Figure 3.1: Examples of successful / unsuccessful reconstructions of the straightforward implementation.

To summarize, a straightforward implementation contains two for-loops. First we loop over rows, then in each row we loop over lenticules to reconstruct colors one lenticule by one lenticule, by interpolation. This implementation already produces valid results, but it is inefficient and hard to utilize the GPU with the for-loops. In our experiment on the validation set, it takes several minutes to reconstruct one image, which is unsatisfactory in practice.

What is more, the method `scipy.signal.find_peaks` does not always find the correct peaks and needs manual fine-tuning. With one parameter set, it skips some peaks, while with another parameter set, it recognizes false ones, which strongly affects color accuracy. As a result, in Figure 3.1b, the green stripes come from lenticule boundaries that are not correctly detected. Therefore, we still need a more robust boundary detector which produces more truthful and coherent boundaries in an automated fashion.

Nevertheless, it already proves the success of our U-Net architecture, and provides a baseline for the following optimization.

3.2 Reconstruct color by convolution

In order to speed up the straightforward implementation of color reconstruction, we need to get rid of the for-loops and replace the interpolation inside for-loops by a similar but more efficient approach. Since interpolation is related to transposed convolution, it is easy to think of applying `PyTorch`'s built-in convolution functions which are optimized on GPUs, and designing a kernel which mimics interpolation.

Figure 3.2 illustrates the steps of reconstructing color by convolution. To clarify the idea, we take only one row of a film scan as an example of the input image, so that we can reduce one dimension for visualization. Furthermore, we use a row of pixels in random colors to represent a row of a real film scan since it is too large and complex.

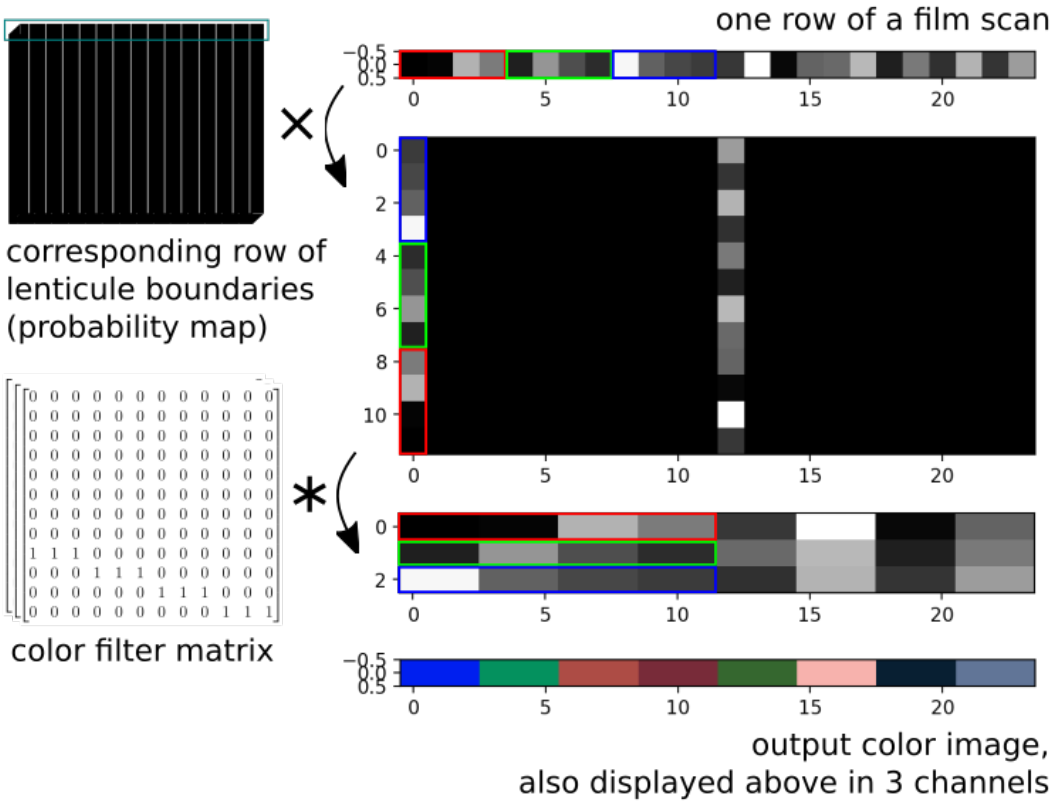
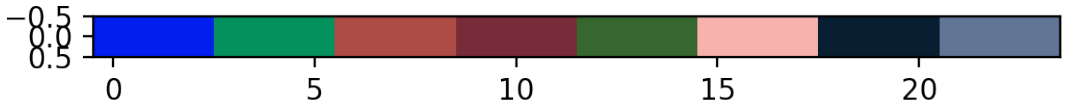
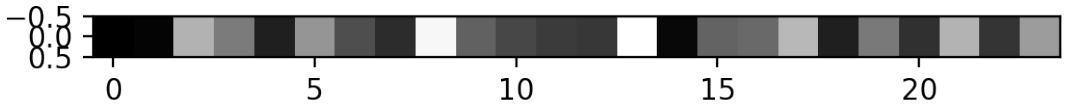


Figure 3.2: steps of reconstructing color by convolution

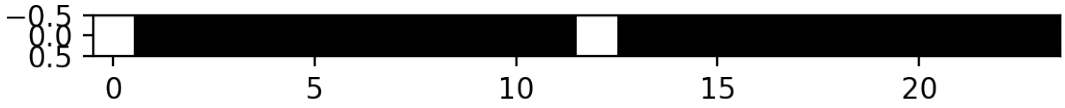
To explain this in detail, Figure 3.3 shows an example of one row of an artificial image which is 1 pixel in height and 24 pixels in width. First, we use the original color image Figure 3.3a to generate an artificial lenticular film image Figure 3.3b, where each pixel is split into three adjacent color channels, each of which is 1 pixel in width. Assuming each lenticule is 12 pixels in width, we can mark their boundaries in Figure 3.3c.



(a) one row of a color image



(b) image stored in lenticular film format



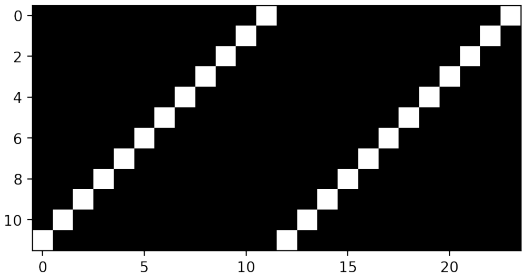
(c) boundaries of lenticules

Figure 3.3: a minimal image to illustrate convolution

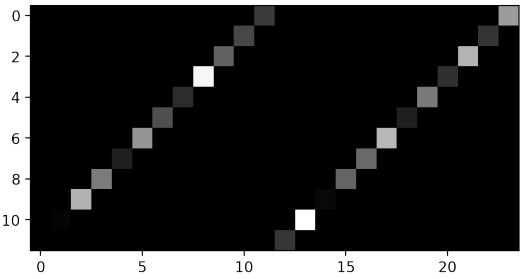
In Figure 3.4 we illustrate the reshaping filter which expands the channel dimension of the image while copying appropriately the color information. First we design a tensor of size `[w, 1, 1, w]` where the diagonal elements on the plane of the first and the last dimension are ones, as shown in Figure 3.4a. Apply it as a convolution kernel on the lenticule boundaries to stack them in space, and its shape becomes `[1, w, 1, W]`, where lenticule width $w = 12$ and image width $W = 24$ in this example.

Then we multiply the lenticule boundaries stacked in space with the input lenticular film image, as in Figure 3.4b, and apply on top another convolution filter of size `[w, w, 1, w]` where the space diagonal elements on the cube of the 1st, 2nd, and 4th dimensions are ones, to align the input image along widths of lenticules. Now this reshaped input image also has the size of `[1, w, 1, W]`.

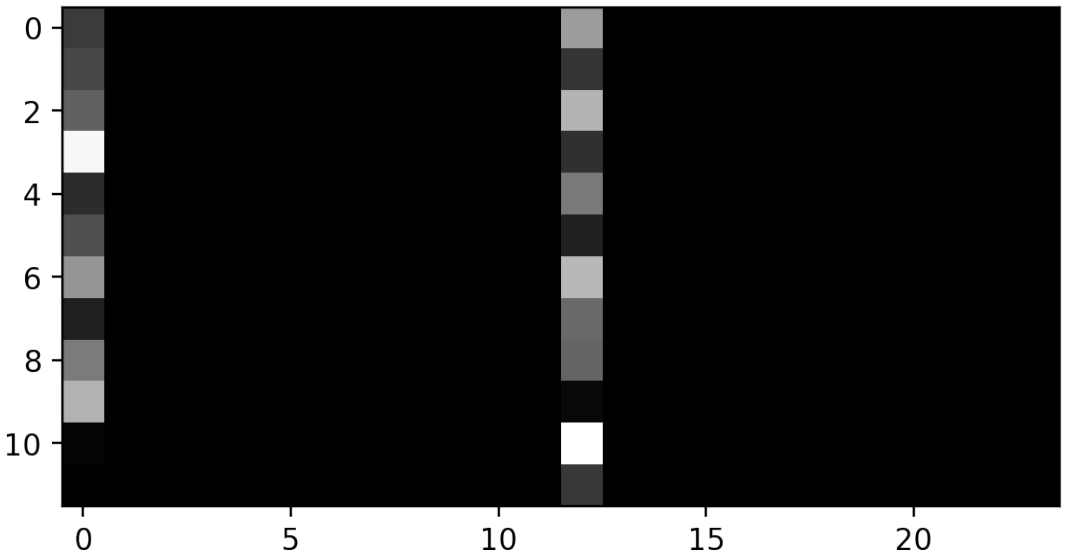
Finally, in Figure 3.5 we apply the color filter of size `[3, w, 1, w]` on the stacked input image of size `[1, w, 1, W]`, to get the reconstructed image of size `[1, 3, 1, W]`, which is shown in Figure 3.5a in separated color channels and Figure 3.5b in colors.



(a) the reshaping filter (visualize 1st, 4th dimension)



(b) multiply the filter with the image (visualize 2nd, 4th dimension)

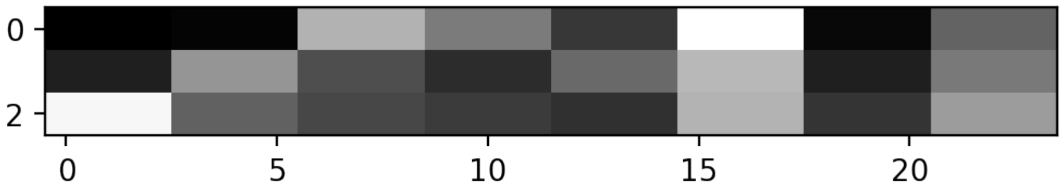


(c) align the stacked image (visualize 2nd, 4th dimension)

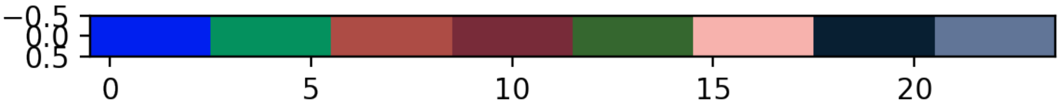
Figure 3.4: the reshaping filter

The 2nd and 4th dimensions of the color filter is three `w*w` matrices for color reconstruction of 3 color channels. The color filter matrix for the red channel of this simple image is as follows:

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1
\end{bmatrix}$$



(a) separated color channels of the reconstructed image



(b) reconstructed image

Figure 3.5: the (linear) color filter

where the last one-third of rows works like the linear interpolation used in the straightforward implementation. The other two channels have similar matrices but they locate at the first and second one-third of rows, respectively. Efficiency of the convolution method is achieved by replacing interpolation inside every lenticule in a for-loop by applying this convolution filter globally for all lenticules.

Another advantage is the flexibility of designing color filter matrices. Without having to develop interpolation methods, we can design the color filter by assigning arbitrary values, as long as the color filter matrices satisfy the constraint that all elements in each column sum up to 1. Based on this constraint, users can then modify it for different color qualities that they desire. For example, we can design the matrix (here only the red channel is shown) in a seemingly simpler way:

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

which is named as a "midpoint" filter. The idea behind this midpoint filter and its reconstruction results are discussed in Section 3.4.

Coherent lenticule boundary corrector

It is important to note that the convolution kernel has a fixed size, where the dimensions of the stack filter and the color filter are equal to the width of the lenticule, namely w in the notation of Section 3.2. However, this is based on the assumption that all lenticules have the same width in one image, which is not the case for almost all images. A straightforward solution is to use the mode of all distances between peaks to represent the global width of lenticules of an image. With minor modification, a better formulation is to use the mode of distances between peaks only in the middle row of the image, which is more efficient. Regardless of which method to use, they both find one integer value that approximates all widths of lenticules in an image. As a result, if we apply these filters directly to the image, there will be many areas with color inconsistencies where the widths of lenticule are not equal to the assumed "global" width.

To achieve better color coherence in compliance with the convolution kernels of a fixed size, we develop a pre-processing function to apply on the lenticule boundary heat map before convolution, where we constrain that all lenticules have the same width in one image. This is done by numerical optimization where the target is that every w consecutive pixels in one row should sum up to 1, so that for each w consecutive pixels, there is one one pixel belonging to the boundary class. We implement this by applying an all-one convolution kernel of size w on the heat map which does the sum, as part of the loss function. At the same time, the optimized heat map of lenticule boundaries should still be similar to the original one, which is the other part of the loss function. Numerical optimization is carried out in an iterative manner by built-in optimizers (e.g. `torch.optim.Adam`) and convolution functions provided by the `PyTorch` library.

Results show that the new implementation making use of convolution is much faster than the straightforward implementation, taking only several seconds for each image. However, it is based on the assumption that all lenticules have the same width, which is only true in the ideal condition. Since conditions of films vary, this assumption is generally not fulfilled. For images containing a relatively large area where lenticule widths are different from those in other areas, it will lead to visual artifacts.

Figure 3.6 shows the limitation of this faster implementation when lenticule widths vary inside one image. In this case, we need a method that can adapt to different lenticule widths while maintaining the efficiency of the convolution method.

3.3 Adapt to flexible lenticule widths

According to empirical experience provided by the doLCE project, for most lenticular films, lenticule widths inside one single image generally have the difference of no more than one pixel. For example, for an image where the majority of

Figure 3.6: "rainbow" color inconsistencies in an image reconstructed by convolution

lenticules are w pixels wide where w is an integer, there is probably an area in the same image where most lenticules have the widths of $w-1$ or $w+1$, but it is unlikely that they have the widths of bigger than $w-1$ or smaller than $w+1$. If we just take the mode of w as the unitary width to design the filter for convolution, there will be artifacts shown in Section 3.2, so it would be ideal if the convolution kernel can adapt to a different size wherever it needs to. However, since the convolution kernel has a fixed size and takes in a whole image in one go, we need at least two separate convolution kernels with different sizes for different lenticule widths.

3.3.1 Soft constraints on widths

Consider an image where the majority of lenticules have the width of w, but also many of them have the width of $w+1$. Compared to representing them all by an integer w, it is more reasonable to have a decimal number between w and $w+1$ that approximates the mean width of lenticules. Since directly applying the built-in peak finder `scipy.signal.find_peaks` does not always output truthful results, especially when the detected boundaries are not clear enough, we adapt the idea from [AB02], [PZW16], and [CFW+19] where images are processed in frequency

domain to remove stripes patterns, but instead of removing these patterns, we only need to calculate the global frequency and period of the lenticule boundaries heat map. First, by applying a 2D fast fourier transform `scipy.fft.fft2` along the width dimension of the image and summing it up along the height dimension, we have a frequency map, e.g. Figure 3.7, along the dimension of repeated stripes. Then we divide the minimum distance (W/w) by the width of the image (W) to get the period of the stripes (w).

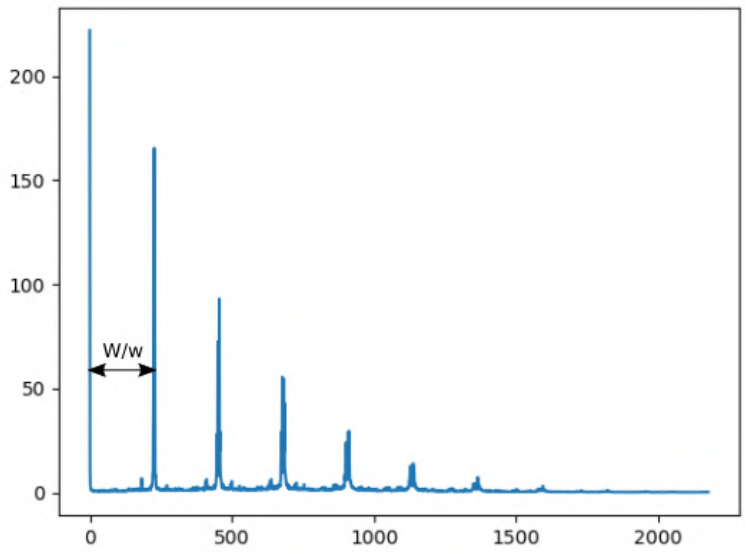


Figure 3.7: an example of the frequency map of a lenticule boundaries heat map

Apart from having a decimal result to represent the global lenticule widths, analyzing the image in the spectral domain is more robust and efficient than finding peak distances row by row in a large image and calculating their means. There is only one parameter that needs manual selection, which is the maximum lenticule width, which we set to 25 according to empirical results by doLCE.

If the decimal global period of stripes is x, the lenticule widths specified in our two filters will be $w = floor(x)$ and $w+1 = ceil(x)$ respectively. They determine the dimension of the convolution kernels applied on the image.

Then the coherent lenticule boundary corrector introduced in Section 3.2 is modified in order to comply with the idea of soft constraints on lenticule widths. Rather than constraining that every w consecutive pixels in one row should sum up to 1, here w is merely replaced by either w or $w + 1$, such that in numerical optimization, the smaller loss between using a convolution kernel of size w and size $w + 1$ is taken into account for each pixel. Other implementation details remain unchanged.

3.3.2 Two filters and width mask

After applying the two filters on the image separately and getting two images, we still need a mask to combine them, which marks where the widths of lenticules are nearer to w and where they are nearer to $w + 1$ (using the notation in Section 3.3.1). In Figure 3.8 we visualize these two images used for combination.

(a) assuming lenticules have size w (b) assuming lenticules have size $w + 1$

Figure 3.8: images reconstructed with different assumptions about lenticule widths

Here we find that, since the nature of convolution is summing over filters applied on the image, when we use the "small" filter to do convolution on the image, where we assume all lenticules are of width x, as shown in Figure 3.8a, in many places there are black stripes where there is no color information, because the lenticule to their left are so narrow that they fail to sum to their position. Similarly, on the image processed by the "large" filter where all lenticules are assumed to have the width of $w+1$, some areas are white or pink as shown in Figure 3.8b, because they are summed up by the colors from two adjacent lenticules, for example the rightmost channel of the lenticule to its left, and the leftmost channel to its right.

We expect a mask ensuring that each pixel is covered by exact one lenticule, which is similar to the idea of the coherent lenticule boundary corrector introduced in Section 3.2 and Section 3.3.1. So we calculate the loss terms for two different convolution kernels again as in Section 3.3.1, which represent how close for w or $w + 1$ consecutive pixels their values sum up to 1, but instead of picking the smaller value for each pixel, we store the arguments in a Boolean mask, which determines for each pixel which size of convolution filters should be applied. The result after masking and combination is shown in Figure 3.9, where there are no more color inconsistencies like in Figure 3.6, Figure 3.8a and Figure 3.8b.

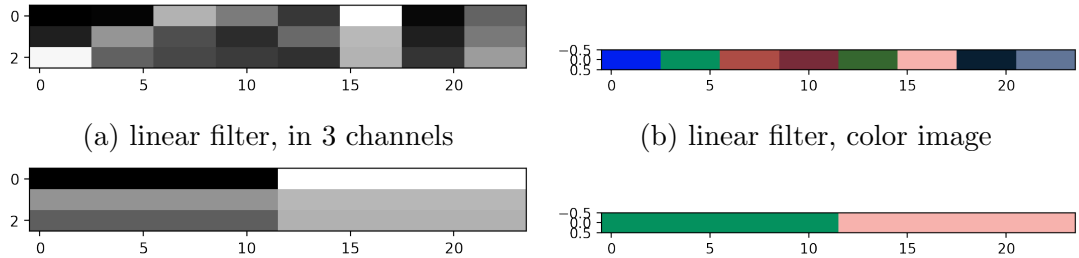
Figure 3.9: a successful color reconstruction by combining two filters

3.4 Color filter design

The only detail left to be discussed is the design of color filter. Since even the lenticule boundaries in the original film scan are not precise, there are always ambiguity in our detected lenticule boundaries, and also the boundaries between color channels inside one lenticule. As a result, the idea of our default filter is that we only pick the pixel at the midpoint of each channel for color reconstruction to avoid misidentifying that a pixel belongs to its neighborhood channel or lenticule. Although it loses some color information, it produces more confident and truthful colors. On the other hand, we can also avoid color loss by applying a linear or other smoothed filters if we are very confident about the detected lenticule boundaries.

Figure 3.10 show the comparison of colors between applying a linear filter and a midpoint filter on the simple image in Section 3.2. Compare Figure 3.10c to Figure 3.10a, and Figure 3.10b to Figure 3.10d, it is clear that the midpoint filter only picks the second block of each lenticule to represent the whole lenticule. Although it seems most color information are lost, this idea could be beneficial for large images.

Figure 3.11 shows the comparison of colors between applying a linear filter and a midpoint filter on an image from the test set. The image reconstructed by the



(a) linear filter, in 3 channels

(b) linear filter, color image

(c) midpoint filter, in 3 channels

(d) midpoint filter, color image

Figure 3.10: comparison of the linear / midpoint color filter on a simple artificial image

midpoint filter seems smoother and cleaner than the one reconstructed by the linear filter. It contains less details but portrays more confident colors.

(a) linear filter

(b) midpoint filter

Figure 3.11: comparison of the linear / midpoint color filter on an example image

4 Evaluation

4.1 Comparison with doLCE

The purpose of the thesis consists in providing a better solution for lenticular film color reconstruction than the former doLCE project which failed in color reconstruction due to hard conditions on the film such as irregular lenticules or blemishes on the film material.

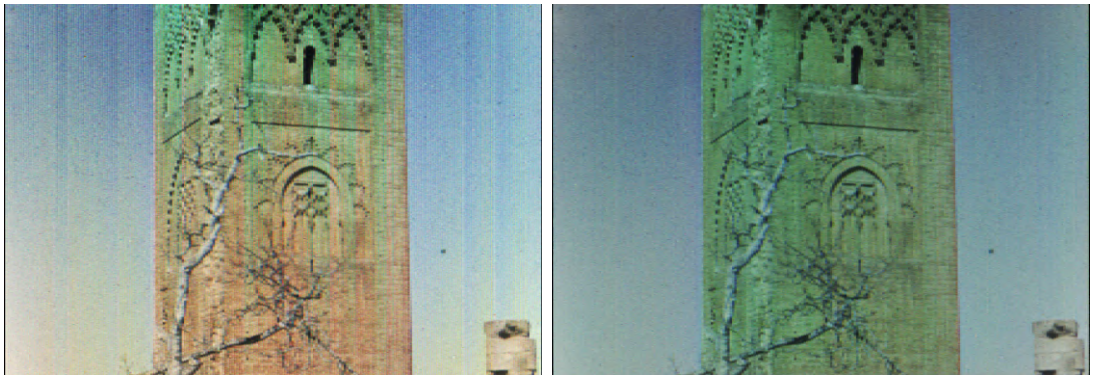
We tested our method with the U-Net architecture proposed in Section 2.1.1, the parameter setting specified in Section 2.2.3, and the midpoint color filter introduced in Section 3.4, on selected images from the dataset provided by Dr. Giorgio Trumpy from the Department of Film Studies, University of Zurich. The results are horizontally flipped to match the setting of the original reconstructions of doLCE. Note that there are also differences in post-processing which determines brightness and saturation of colors, which is not covered by the thesis topic.

In Figure 4.1, the result from doLCE has serious inconsistencies and unrealistic colors, but our method (deep-doLCE) provides a smoother and more truthful result without these artifacts.

(a) doLCE (b) deep-doLCE

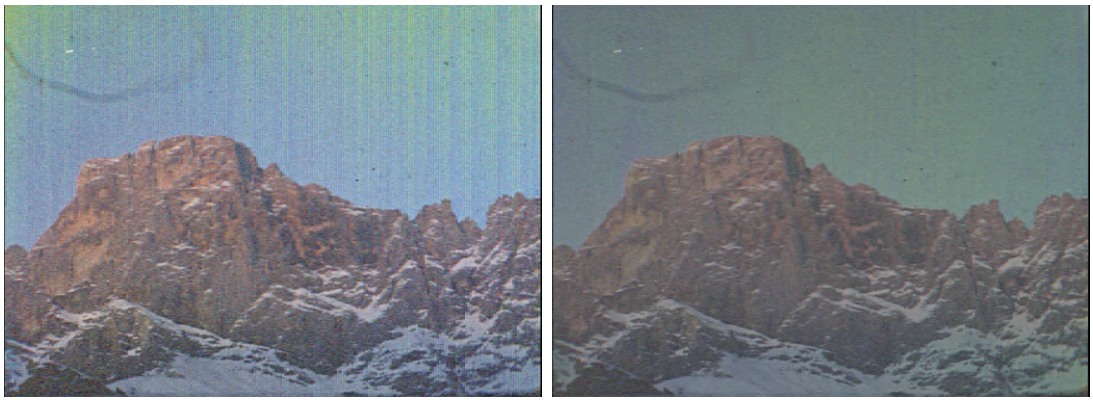
Figure 4.1: `AfrikaK2_1186.png`

In Figure 4.2 and Figure 4.3, the green stripes appearing in results from doLCE are removed in deep-doLCE.



(a) doLCE (b) deep-doLCE

Figure 4.2: `AfrikaK2_1249.png`



(a) doLCE (b) deep-doLCE

Figure 4.3: `SpinabadAS_0013.png`

Figure 4.4 and Figure 4.5 both contain a wide stripe where the color is inconsistent (the red stripe to the right of the standing woman in Figure 4.4a; the blue stripe at the middle right part in Figure 4.5a). These color stripes are both removed in the reconstructions of deep-doLCE.

4.2 Discussion

Most color reconstruction of deep-doLCE are satisfactory, but there are still some images that are not successfully reconstructed. An example is Figure 4.6, where both methods fail, but with different artifacts.

(a) doLCE (b) deep-doLCE

Figure 4.4: `MarleneAS_0097.png`

(a) doLCE (b) deep-doLCE

Figure 4.5: `RolleaAS_0221.png`

In Figure 4.6a, there are many green stripes in the sky where it should always be blue. In Figure 4.6b, these stripes are removed by deep-doLCE but the right part of the image is overall greenish. In fact, Figure 4.3b also has this problem of green hue in the sky area, although it is less severe.

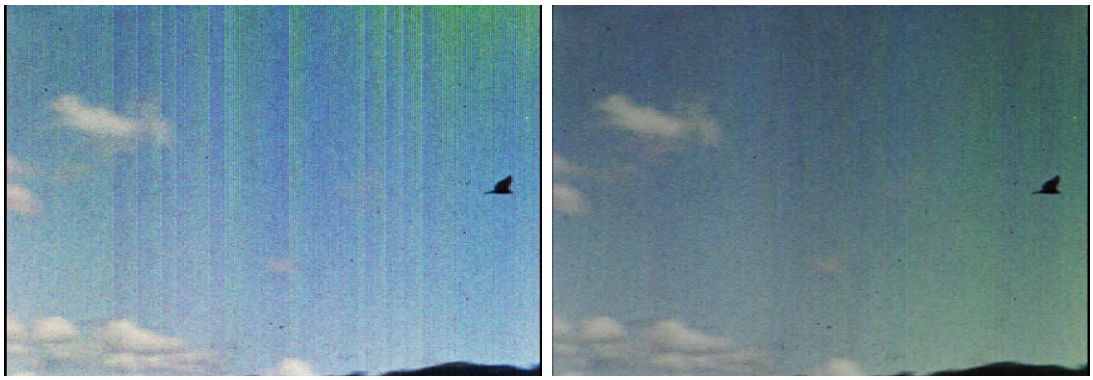
We inspect the lenticular film scan of this failed image to find that the lenticule boundaries (which are represented by black vertical lines on the image) appear thicker in its right part than its left part, and also much thicker than the lenticule boundaries in successful reconstructions. We need extra speculation to locate the exact boundaries of the lenticules inside these thick lines.

Taking the area with a bird shown in Figure 4.7 as an example, we compare the ground truth of lenticule boundaries drawn by hand with the results of our detection. First, since our detector is trained by thousands of samples to looks for edges, they are likely to find the most probable lenticule boundaries towards the middle of black lines, which is marked in red. Second, since this picture is blue

overall, we expect general higher color intensities in its blue channels. As we know from the essence of the lenticular films that brighter color in the gray-scaled film scan corresponds to higher intensity in a color channel, we conclude that these brighter parts in the original film scan belong to blue color channels. According to this, we can draw the ground truth of lenticule boundaries approximately by hand, which is marked in green. Difference of the two color marks shows that our detection has a right offset compared to the ground truth, which results in a false match between the brighter parts in the film scan and the green color channel, so the reconstructed image appears overall greenish.

Note that the drawing in Figure 4.7 is exaggerated to show the problem, which means our detection is only around 2 pixels away from the ground truth. However, since each color channel is only several pixels wide, there are already visual defects.

Knowing the source of defects does not automatically provide us with a solution because we can only estimate the correct boundaries knowing that this image is overall blue. Without this prior knowledge, any pixel inside the dark parts of the film scan could be a boundary pixel, but our current detector can only take a general guess based on its training set, which is not guaranteed to find the exact lenticule boundaries. One possible solution is to take into account prior statistics of color in training. There are models to estimate color statistics from images of gray scale and we may apply them alongside our current detector.



(a) doLCE (b) deep-doLCE

Figure 4.6: `AfrikaK2_0339.png`

4.3 conclusion

Evaluation results have shown that our proposed method, namely deep-doLCE, outperforms the solution by the legacy doLCE project in color reconstruction for lenticular film images with bad conditions such as irregular lenticules and dark areas. For areas on images where the lenticule boundaries are thick, further

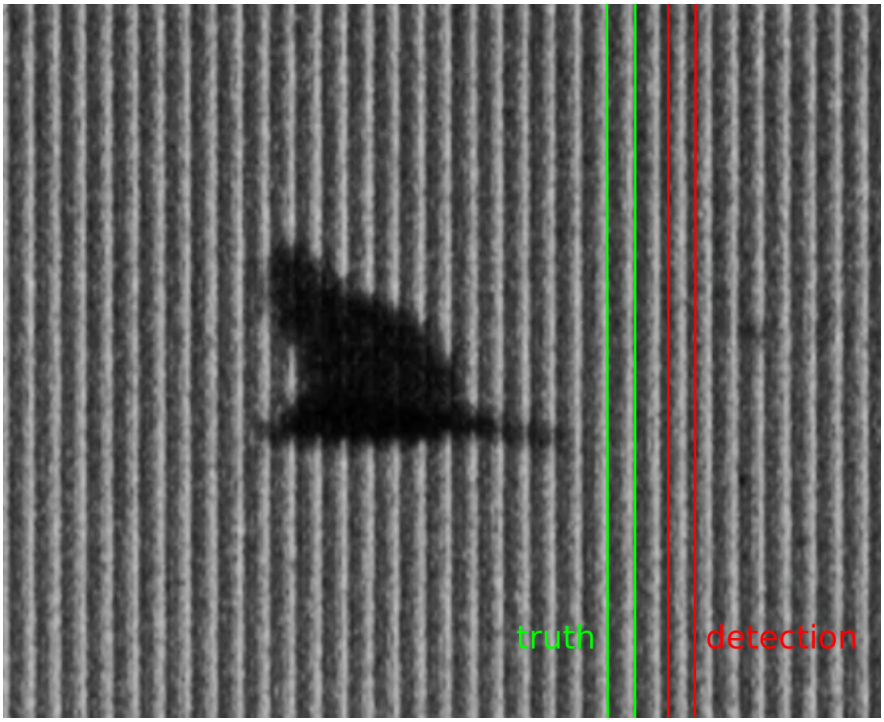


Figure 4.7: the lenticular film scan of `AfrikaK2_0339.png`

modification in designing the convolution filters are needed to remedy the loss of color information on the occluded color channels.

Bibliography

[AB02] Igor N Aizenberg and Constantine Butakoff. Frequency domain medianlike filter for periodic and quasi-periodic noise removal. In *Image Processing: Algorithms and Systems*, volume 4667, pages 181–191. International Society for Optics and Photonics, 2002.

[CFW+19] Boyang Chen, Xuan Feng, Ronghua Wu, Qiang Guo, Xi Wang, and Shiming Ge. Adaptive wavelet filter with edge compensation for remote sensing image denoising. *IEEE Access*, 7:91966–91979, 2019.

[CMW37] JG Capstaff, OE Miller, and LS Wilder. The projection of lenticular color-films. *Journal of the Society of Motion Picture Engineers*, 28(2):123–135, 1937.

[CS28] JG Capstaff and MW Seymour. The kodacolor process for amateur color cinematography. *Transactions of the Society of Motion Picture Engineers*, 12(36):940–947, 1928.

[DSL+18] Ruoxi Deng, Chunhua Shen, Shengjun Liu, Huibing Wang, and Xinru Liu. Learning to predict crisp boundaries. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 562–578, 2018.

[Flu12] Barbara Flueckiger. Timeline of Historical Film Colors, 2012.

[LCH+17] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. Richer convolutional features for edge detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3000–3009, 2017.

[LZW19] Qingyang Li, Ruofei Zhong, and Ya Wang. A method for the destriping of an orbita hyperspectral image with adaptive moment matching and unidirectional total variation. *Remote Sensing*, 11(18):2098, 2019.

[PZW16] Jihong Pei, Mi Zou, and Lixia Wang. A local adaptive threshold noise detection linear interpolation filter (lalif) for stripe noise removal

in infrared images. In *2016 IEEE 13th International Conference on Signal Processing (ICSP)*, pages 681–686. IEEE, 2016.

[R+14] Joakim Reuteler et al. Die farben des riffelfilms: digitale farbrekonstruktion von linsenrasterfilm. *Rundbrief Fotografie*, 81:37–41, 2014.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[SLV+17] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 240–248. Springer, 2017.

[TDGF18] Giorgio Trumpy, Josephine Diecke, Rudolf Gschwind, and Barbara Flueckiger. Cyber-digitization: pushing the borders of film restoration's ethic. *Electronic Media & Visual Arts*, pages 190–195, 2018.

[Wei33] F Weil. The optical-photographic principles of the agfacolor process. *Journal of the Society of Motion Picture Engineers*, 20(4):301–308, 1933.

[XT15] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.