University of Bremen

Masterarbeit

A Deep Learning Approach to Color Spill Suppression in Chroma Keying

Pascale Maul

Course of Study:	Computer Science
Examiner:	Udo Frese, Hans Meine
Supervisor:	Udo Frese
Commenced:	April 04, 2019
Completed:	October 18, 2019

Abstract

This thesis proposes a deep learning approach to color spill. Color spill is the effect that occurs when lighting is reflected from the background onto the foreground. This phenomenon occurs mainly in chroma key settings, i.e. upon shooting movies with green-screen scenes, which are then replaced by for example an animated background. Here, the main concerns are creating an image matte and suppressing spill affected areas. As of yet, this process requires heavy user interaction and is very time consuming.
One particular difficulty is to distinguish between mixed and spilled pixels. Both appear in form of pixels with a green tinge. In contrast to spilled ones, mixed pixels are caused by foreground-background interactions that should be translated to the new image composition. Blurred areas are to remain translucent, while the background should be let through. The same applies to areas where the foreground object is translucent. In contrast, spilling artifacts should be removed and the original color of the foreground object should be restored. Not doing so results in inconsistencies between the lighting conditions of the foreground and new background, which are easily detected by the human eye.
Color spill is a phenomenon with a ground truth, that is difficult to acquire. Therefore, no data sets containing relevant information exist. Rendering synthetic images to construct such a data set is the first major contribution of this work. The second one is to train a convolutional neural network on the acquired data set, with a custom loss function.
The network generates an RGBA image from an RGB image, by simultaneously predicting the alpha map and RGB pixels with a suppressed spill. While the custom accuracy for predicted green screen images doesn't exceed 80%, the network shows great generalization capabilities as its predictions translate well to natural images.

Contents

List of Figures

List of Tables

List of Abbreviations

CNN	convolutional neural network. 11
FNN	feedforward neural network. 11
ML	machine learning. 13
MSE	mean squared error. 24
ReLu	rectified linear unit. 15
RMSProp	root mean squared propagation. 19
SAD	sum of absolute differences. 24
SGD	stochastic gradient descent. 19
SIM	spill with identity mapping. 42
WMSE	weighted mean squared error. 37

+

1 Introduction

1.1 Chroma Keying and Color Spill

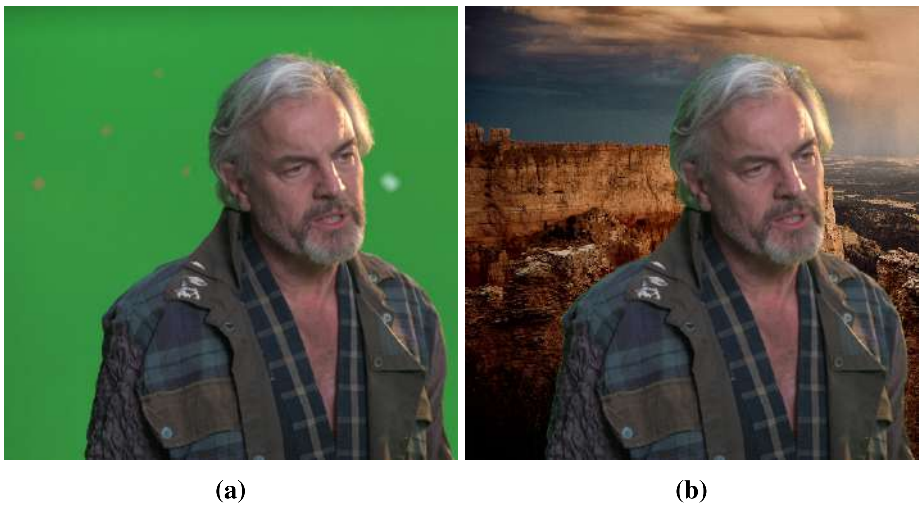
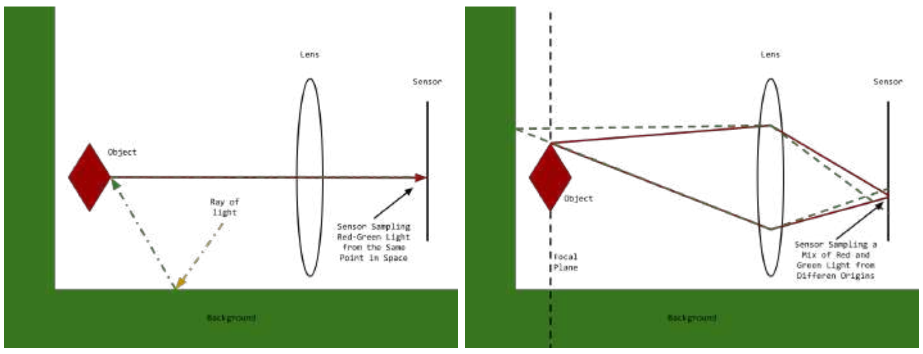


Figure 1.1: Image (a) is a typical green screen image. It originates from the Tears of Steal movie [13]. Image (b) shows the same image composed onto a new background. Besides separating the foreground from the background, the image has not been processed any further. The composite clearly shows spilled and mixed pixels. While the mixed pixels appear in the border area, spilled pixels appear on the left side of the actor's jacked, on his beard, and neck. The background is a photography from Mark Boss and was downloaded from Unsplash.

Green screens are commonly known from their usage in weather forecasts. The forecaster stands in front of a green background which is then replaced with the weather map in post-production. Other popular uses of green screens include movie productions that rely on separately filmed or animated background shootings.
The method of replacing an uni-colored background with a different one is referred to as *chroma keying*. In principle it works for any background color, but green is often chosen as it doesn't interfere with natural hair or skin colors.
Problems arise, when light reflects from the background onto the foreground, or if blur or transparency in foreground objects causes the back- and foreground colors to mix. The foreground contains areas with a green tinge, which becomes obvious once the final composition is in place. It can be said that with the background color spills onto the foreground, hence the term *color spill*. Pixels, whose colors are affected through background reflections are called *spilled pixels*. Pixels affected through blur/transparency are referred to as *mixed pixels*. Figure 1.1 shows an original

green screen image and the foreground image composed onto a new background, illustrating the concept of mixed and spilled pixels.
Mixed pixels are the result from merging color information that originates from different points in space into one image point. For example transparent objects are permeable to lighting originating in the background, causing the camera to sample color from both objects at one sensor point. The mechanism for mixed pixels that are created through blur is also based on the sampling of light that originates from different points in space but is more complex and involves the mechanics of an optical lens. This lens is responsible for the projection of light onto the camera sensor. The light passing through the lens is bend to a matching point behind it. The distance between the matching point and the lens depends on the distance between the portrayed object and the lens. Only light originating from the called focal plane have their matching point aligned with the lens-sensor distance. Light originating from points in space further away than the focal plane have their matching point before the sensor, while the ones from a closer distance have their matching point behind the sensor. [16] In both cases this results in the light being dispersed onto a sensor area instead of a sensor point, creating blur in images. Figure 1.2 illustrates the light paths responsible for the appearance of (blurred) mixed pixels and spilled pixels.


Lens
Sensor
Object
Ray of light
Sensor Sampling Red-Green Light from the Same Point in Space
Background
Lens
Sensor
Object
Focal Plane
Sensor Sampling a Mix of Red and Green Light from Differen Origins
Background

(a) Spill Effect **(b)** Blur Effect

Figure 1.2: Here the difference between color spill and blur is illustrated. Figure (a) sketches how light is reflected from the floor onto the object, dyeing the object in the background's color. This area then appears to have a green tinge and is sampled accordingly on the camera's sensor. Figure (b) illustrates how light originating from different points in space are projected differently through the lens onto the sensor, creating mixed pixels.

1.2 Spill Suppression

This thesis proposes a deep learning approach to color spill correction for images with uni-colored backgrounds. The term color spill refers to undesired color artifacts that appear most often in green or blue screen scenes and are caused by reflections of the background onto the foreground. Even slight reflections may become obvious once composing the foreground onto a new background. Here, the human observer quickly detects inconsistencies between the color of the composition's background and foreground areas that are affected by reflections of the original background. These inconsistencies are inevitable in any chroma key setting, and to date there exists no system that is capable of removing all reflection artifacts without user interaction.
The basic steps to acquire the an image composition with a color corrected foreground scene are:

1. foreground/background definition
2. foreground/background separation
3. color spill suppression
4. computation of the final composite

Point 1 and 2 of the pipeline are closely related to the *image matting* problem. Image matting is the process of extracting the foreground image by separating it from the background. The term *natural image matting* describes the extraction of a foreground from a natural scene. Here, the background is dependent on the use case, and thus everything is a potential fore- or background. Which is the reason why natural image matting systems require a coarse partition of the image, known as *trimap*, into foreground, background, and undefined regions.
The product of the matting process is the *matte* or *alpha map*, which defines every pixel's translucency or opacity. This possibility of marking pixels to partially belong to the fore- and background is what distinguishes image matting from image segmentation, where each pixel obtains an unambiguous label. The need for translucent pixels becomes apparent for areas where foreground/background interactions are either caused by the object's (semi-)transparency or borders, where blur of varying degrees causes a mix of fore- and background colors. In these cases the foreground's property of interacting with any given background should be preserved, to retain transparency and blur. The compositing equation, first referenced in [10] captures this separation in the following way:

$$C = (1 - \alpha)B + \alpha F \tag{1.1}$$

The final composite image C is a mix of background B and foreground F, diminishing the influence of the background image on the foreground by factor $(1 - \alpha)$.

This examination of the image matting problem reveals a particular difficulty in the described pipeline: There exists different lighting interactions between fore- and background that manifest in the same manner (pixels with a green/blue tinge) but should be resolved differently. Pixels dyed in the background's color, also referred to as mixed pixels, because of the foreground object's transparency or blur, should also simulate the same lighting interactions with the new background. Thus, the main concern is to find an appropriate alpha value. In contrast opaque areas, where the pixel color is altered through background reflections, should retain their opacity while the RGB value needs to be corrected to represent the object's “true„ shade.

1.3 Project Scope and Contributions

While Chapters 4 and 5 give a detailed description of the project's individual components, this chapter summarizes this project's individual contributions and presents the research question at hand.

As the goal is to train a deep learning system, the first concerns is data acquisition. A suitable data set would provide

- a large quantity of images with uni-colored background,
- the corresponding alpha mattes,

- and the ground truth consisting of foreground RGB data.

To date there exists no appropriate data set which fulfills the named requirements. Especially none which provides the RGB data that is unaffected by light from the background. Which is the why one major contribution of this work consists of creating such a data set. It preferably provides a variety of image structures and enough examples so that the system has the opportunity to learn the difference between mixed and spilled pixels. Thus, it should contain a large number of mixed, and an even larger quantity of spilled pixels. Spill suppression is especially challenging in cases where the whole foreground is affected by background reflections. Therefore, images with a heavy spill effect should be included in the data set. But not all images should show large spill artifacts, because this might prime the system to expect spill in large quantities, while most chroma keying setups already minimize the spill effect through clever lighting.
Taking the above points into account, a large data set has been created by rendering images with Blender. The advantages of synthetic over natural images, are their accuracy and the ease of creating ground truth data. It is not viable to either remove lighting interactions between fore- and background in natural images or to create a large quantity of natural green/blue screen images. The feasibility of using synthetic image data has already been demonstrated by Tremblay et al. [22]. Here, the researchers trained a network on synthetic data and acquired results which where comparable to a network trained on real-world data. The usage of synthetic data is discussed in greater detail in Section 4.1.

The state of the art in spill suppression is mainly defined by image matting and color unmixing publications, as it is not an active research topic. The results of Xu et al. [23] in employing the deep learning framework for natural image matting, reinforced this works choice in using the deep learning framework. The fact that the deep learning framework is capable of taking into account high level features such as object shapes and textures, was one of the main reasons for the system's success, according to the authors.
As such, the second contribution of this project is to develop a deep learning system that is proficient in creating an alpha matte and simultaneously generating RGB data with spill suppression. This system uses a convolutional neural network (CNN) which computes the spill suppressed image with the corresponding alpha channel (i.e. RGBA output) from a RGB image. Given the architecture used by Xu et al. this project used the well founded U-Net architecture (see Section 2.5). The trained system uses a custom loss function which reinforces the difference between mixed and spilled pixels (Section 5.3).

In summary, this work investigates the feasibility of using synthetic data to learn color spill with the help of the deep learning framework. The developed system is then evaluated on a quantitative and qualitative level. The quantitative evaluation is based on custom metrics, while the qualitative one uses non-synthetic green-screen images to demonstrate the system's performance on the generated data set and its applicability to real-world data.

1.4 Thesis Outline

To begin with, the theoretical background of the deep learning framework with focus on embedding the introduced methods in machine learning principles is presented. The main points here are feedforward neural networks (FNNs), CNNs, the training/learning process, and regularization. Additionally, the earlier addressed U-Net architecture is presented in detail.

The following chapter outlines the current state of the art in color spill. To do so three publications are presented and analyzed.
It follows a chapter that describes the process of the data set creation. Another chapter is dedicated to the development of the spill suppressing deep learning system.
The results are presented and analyzed in the subsequent chapter. As mentioned above, the analysis is two-fold: First the system is evaluated with custom metrics. Second image examples are used to present the strengths and weaknesses of the system.
A closing chapter summarizes the project and draft future possibilities in the spill suppression domain.

2 Deep Learning

A computer program is said to **learn** from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.

Tom Mitchell

This definition of learning by Mitchell [15] is crucial to every machine learning (ML) application and also the core of every deep learning system. As such, this chapter begins by introducing fundamental machine learning concepts. Then, fundamental types of neural networks are presented: Firstly feed forward ones and then the more complex convolutional neural networks. The following section then focuses on the training of neural networks by presenting the backpropagation and gradient descent algorithm. Finally, the U-Net is presented, which is the network architecture used in this work.

2.1 Machine Learning Principles

ML applies statistical principles to automatically approximate complex functions [9]. The functions may describe image classes, specify color schemes for black and white pictures, or be a mapping between languages which is used for automatic machine translation. This automatic approximation is acquired through *learning*. The citation in the beginning of the chapter recites the definition of learning by Mitchell in 1997 [15].
Using the color spill problem as an example, then the task *T* would be separating foreground and background with simultaneous un-spilling of the foreground scene. The performance measure *P* would be the loss function, while the experience *E* would be the training of the neural network with the constructed data set. Machine learning tasks are frequently categorized as *regression* or *classification* problem. A problem is defined as a regression problem, if it maps its input onto a numerical value. It is on the other hand defined as a classification problem if it maps its input onto one of *N* categories. For some problems, the category depends on the design of the *target function*, i.e. the problem formulation. The segmentation of an image may for example be formulated as both, a classification or regression problem. It is a classification problem if the algorithm labels individual pixels as $\{1, 0\}$ for foreground and background, but would be a regression problem if the algorithm assigns each pixel a percentage of it being a foreground/background pixel.
The choice of experience *E* has a great influence on the later performance of the ML system and should be chosen carefully. How well the training experience represents the real distribution of

examples is a crucial factor. If, for example, this work's data set would be missing images with blur, then it is unlikely that the trained algorithm would be usable in real world applications, because it would not be able to un-spill blurred regions.

Choosing the performance's estimate depends on the system's design, setting and purpose. In one setting it may be more important for a binary classifier to correctly classify all positive examples at the expense of negative ones, while in another setting a good overall performance is more important. In this works application it would not be a sensible choice to let the algorithm un-spill the background pixels, it can just ignore them and focus on the foreground pixels (What would be the ground truth?).

The components of every machine learning system are: a task, target function, data set/experience, performance measure, and the learning algorithm [15] .

To explain the following terms, the example of foreground - background separation is used. The target function is represented in the image matting problem by the neural network. It labels each pixel as either fore- or background. It is trained by the gradient descent algorithm with backpropagation. Here, the backpropagation algorithm computes the gradient of a loss function which is in turn used by the gradient descent algorithm to train the network by changing its parameters. The problem of determining to what extend each processing step is responsible for the final outcome/error is not trivial and is known as the *credit assignment* problem [15].

By choosing a model, it is assumed that this particular model has the representational power, also called *capacity*, to represent the true target function. The capacity is a model's ability to fit a variety of functions [9]. If the target function's complexity exceeds the model's capability of what it can represent, then its capacity is insufficient and the performance of poor quality. If the model's capacity exceeds the complexity of the problem at hand, then the model is likely to learn noise present in the training set, instead of the true underlying distribution. The above points are tied to the concepts of *generalizazion*, *underfitting* and *overfitting*. The model representing function f is said to *overfit*, if there exists a function f' that performs worse on the training set but has a smaller error over a set that reflects the true distribution of instances [15]. In contrast the model is said to *underfit*, if it cannot acquire a satisfying small error on the training and test set. The ability to perform well on previously unobserved samples is called generalization. To test the generalization of a model, the dataset is split into training set and test set. The training set is used to optimize the model and the test set is used to compute the generalization error. It is thus desirable to have a small training error (avoiding underfitting) and to minimize the gap between training and testing error (avoiding overfitting) to acquire good generalization. Underfitting can be avoided by adjusting the model's capacity, while a there exists a whole set of separate methods to avoid overfitting, which are collectively known as regularization methods.

2.2 Feedforward Neural Networks

FNN are function compositions described by a weighted directed acyclic graph. The graph's nodes are called *neurons* and constitute computational units that analyze whether a certain feature is present in their input signal or not. Neurons are organized into sets called *layers* as shown in Figure 2.1. There are three types of layers: input, hidden, and output layers. The dimensionality of the hidden layers defines the network's *width*, the number of layers is referred to as the network's *depth*.

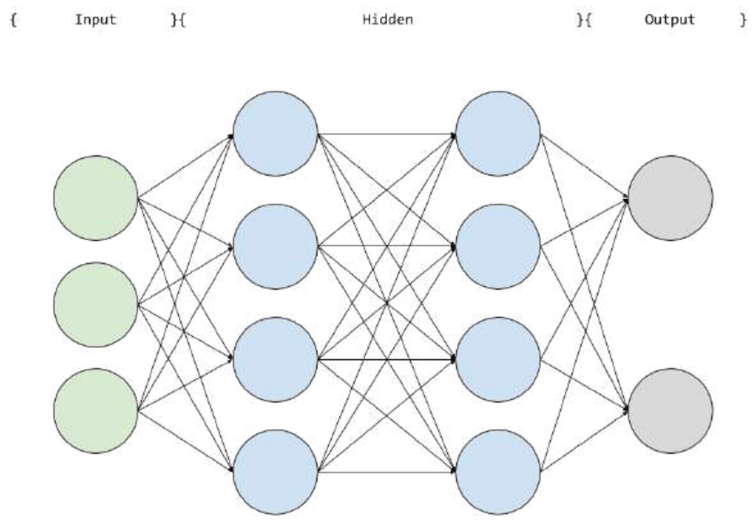


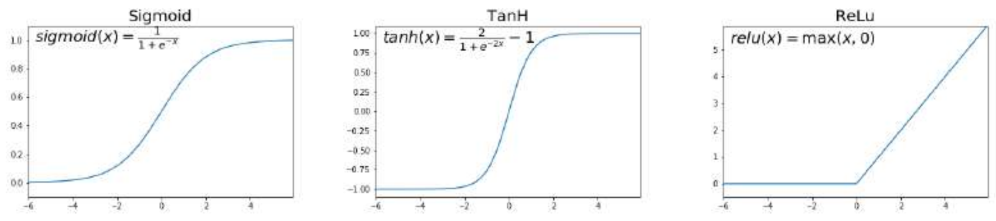
Figure 2.1: FNN with all-to-all connections. The network has one input, two hidden and one output layer.

Those two factors influence to a large degree the network's capacity. A network is usually labeled as deep if it has at least one hidden layer, but modern literature on deep learning often revolves around models having many of hidden layers.
Let $v_1, ..., v_m$ denote neurons that are connected to neuron v_k and let $w_{1k}, .., w_{mk}$ denote the weights of the corresponding connections. The output y_k of neuron v_k is then computed the following way:

$$y_k = \phi\left(b_k + \sum_{i=1}^{m} w_{ik} v_i\right) \tag{2.1}$$

Where b_k is called *bias* and ϕ is called *activation function*.
The main purpose of the activation function is to introduce nonlinearity into the model so that it is able to approximate non-linear functions. Desirable traits in activation functions are the support of sparsity (see section 2.3), computational efficiency, and differentiability. If the function is used to model a probability distribution, the activity should be limited to $[0, 1]$. Different types of activation functions (see figure 2.2) are available, with the standard choice for hidden layers being the ReLu. Although not differentiable in 0, it promotes sparsity and computational efficiency. Sparsity is easily introduced upon initializing the network, i.e. computing the initial weights, by randomly sampling from the standard normal distribution [8]. The choice of the activation function of the output layer depends on whether the model is solving a regression or classification problem. For regression problems a linear function is usually assigned to the output neurons.
For classification problems, the output of the final layer is not computed as shown in equation 2.1. The standard is to use the softmax function which predicts the probability for each of the n classes:

$$softmax(x)_k = \frac{\exp(x_k)}{\sum_{i=1}^{n} \exp(x_i)} \tag{2.2}$$



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

Figure 2.2: The above figures show different activation functions. Prior to the rectified linear unit (ReLu) the sigmoid and the hyperbolic tangens were popular choices for hidden units. The main reason for the frequent use of the ReLu is, that pre-training becomes dispensable. Additionally the sigmoid saturates for very high and low values which obstructs learning. If the ReLu is not an option, it is recommended to use the hyperbolic tangens instead. It behaves approximately linear near 0 and saturation can be avoided as long as the input values are kept small. [8]

Where x_k is the linear part of the neurons activity:

$$x_k = b_k + \sum_{i=1}^{m} w_{ik} v_i \tag{2.3}$$

2.3 Convolutional Neural Networks

2.3.1 The Convolution Function

Convolutional neural networks are especially well suited to process data with grid-like topology, such as images. They are a type of FNN that uses the convolution operation to propagate neural activity. Three important concepts are supported by using the convolutional operation: equivariance, parameter sharing, and sparsity.
The principle of equivariance is especially important to image processing as it reinforces the idea that structures in images are the same, no matter where in the image they appear. From a mathematical perspective function f is defined as equivariant to function g if $f(g(x)) = g(f(x))$. In the context of neural networks, the convolutional operation is equivariant to translations. It follows that applying the convolution function and then shifting the image n pixels to the right gives the same result as first shifting the image and then applying the convolutional operation. This is especially useful as it means that the location of a particular does not matter for its detection. It is noted that convolution is not equivariant to changes in scale or rotations of the image.
In [8] multiple arguments are named in favor of sparse interactions. For one, the size of the representation can be varied by controlling the number of active neurons. This way the size of the representation can be adapted to the amount of information carried by the input. Sparsity is also linked to linear separability as only a subset of the dimensions are used to form a representation. Additionally, sparse representations are statistically more efficient than dense distributions. In CNNs this concept is realized through a sparse connectivity matrix.
The convolutional operation entails multiple units sharing one set of weights, also referred to as *parameter sharing*. This is useful, because storage handling is a problem in deep networks and

parameter sharing lessens the memory requirements. It also entails that instead of learning a different parameter set for every location, one set is learned for all sites.
The convolution of a 2D gray scale image I with a 2D kernel K is defined as:

$$(I * K)(i, j) = \sum_{m} \sum_{n} I(m, n)K(i - m, j - n) \tag{2.4}$$

In a CNN the kernel is formed through the neurons' weights, which are shared across the layer (see Figure 2.3).
As such, each neuron applies the same function but at different locations of the input. The collective output of neurons that share one set of weights is often referred to as *feature map*. A feature is a particular structure in the input signal that evokes a strong neural response. An edge of a certain orientation is an example for a feature which is often learned in low-level layers.
Convolutional layers can also be used to downsample the input. This is done by varying the *stride* of the convolution. The stride defines how often and where the kernel is applied to the input. With a stride of 1, kernel K is applied to every pixel position (i, j). For a stride of size s in i-direction, K is only applied to (i, j) where $i \in \{0 * s, 1 * s, ..., (I - 1) * s\}$. The stride may vary in i and j direction. Naturally, increasing the stride decreases the spatial resolution of the resulting feature map.
Contrary to fully connected FNN, there is not always a path that leads from every input neuron to every neuron located in one of the subsequent layers. This leads to the concept of a *receptive field*. The receptive field indicates what portion of the input affects a particular neuron. The size of the receptive field of the hidden neurons from Figure 2.3 is 2. A special role plays here the pooling operation: It drastically increases receptive field's size. For 2x2 pooling, the neuron, has a receptive field four times the size of its precedents.
How the bias is used in CNNs is dependent on the type of layer used. Generally one bias is assigned to each set of weights. The standard convolution has thus just one bias for each set of units.

2.3.2 Pooling and Upsampling

A convolutional layer is often paired with a subsequent pooling layer. Pooling is the process of applying a summary statistic to each feature map of the layer. One variant is max pooling, where in a rectangular neighborhood only the maximum activity is propagated. Other versions include applying the L^2 norm or the weighted average based on the distance from the central pixel. Pooling promotes invariance to local translations and is particularly useful if the presence of a feature is more important than its exact location.
In contrast to pooling, which compresses the image and contained information, builds the upsampling operation, which expands the image. The simplest method in doing so is to in crease one pixel to a nxn (typically 2x2) image patch and giving one pixel the the value of the pixel the patch originated from while filling the other pixels with zeros, while retaining the original topology of the image. Individual upsampling methods may differ in how they choose the fill value. Other possibilities include copying the value from the nearest neighboring pixel or interpolation.

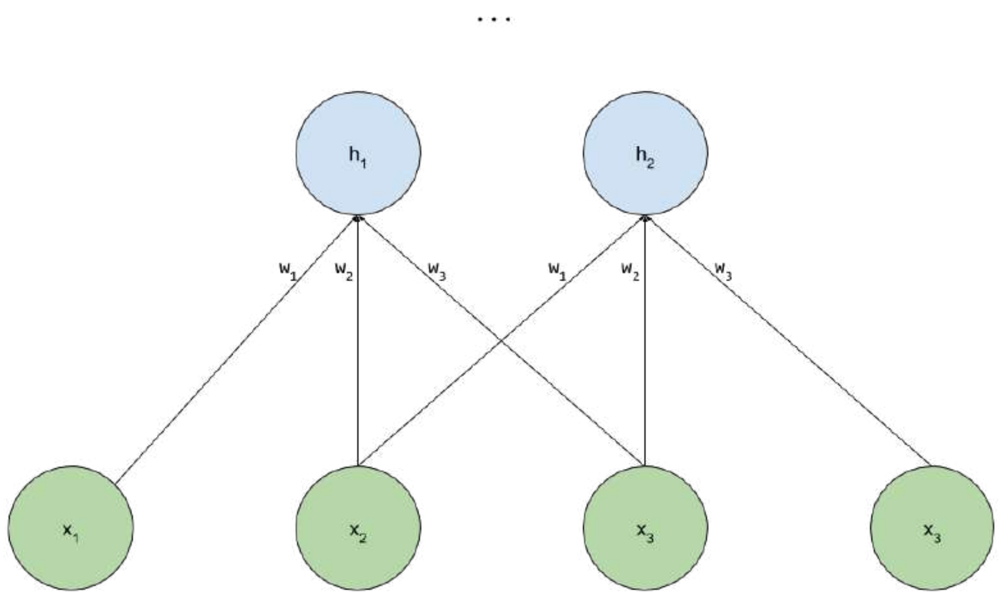


Figure 2.3: The realization of valid convolution with stride 1 in CNNs. The input neurons are colored green and the convolutional layer is colored blue. Each neuron of the convolutional layer uses the same set of weights but at different locations of the input. The size of the hidden units' receptive field is 2.

2.4 How Neural Networks Learn

This section is dedicated to the training process. It firstly presents the gradient descent algorithm, the backpropagation algorithm, and variations of the gradient descent algorithm.

2.4.1 The Gradient Descent Algorithm with Backpropagation

The gradient descent algorithm is the component responsible for the learning process. It is an optimization algorithm that uses the gradient of the objective function J for learning. The objective function is derived from the loss function. The difference between the two is what is considered as function input and what is considered the function's parameter. The aim of the loss function is to evaluate the network's output given a specific data point. It is used to reinforce important principles in the output. The classical example in regression losses would be that small errors matter less than big ones. The objective function in contrast views the data input as a parameter and computes the loss with the network parameters (i.e. weights and biases) as input and the learning algorithm modifies the model's parameters θ to minimize the objective function. In short one's purpose is to measure the distance between network output and ground truth, or certain aspects of the ground

truth, while the other's purpose is to maximize the quality of the networks performance.
The principle of gradient based learning is to use the gradient of the error surface of the network, that is the error given the network parameters θ, to improve the networks output by changing θ. Because the gradient of the objective function always points into the direction of steepest ascent, it is guaranteed that $J(\theta) > J(\theta - \epsilon * \nabla_\theta J(\theta))$ for small enough ϵ. The algorithm thus „descents "the error surface by moving θ in small steps. New points are calculated by updating θ with

$$\theta \leftarrow \theta - \epsilon \nabla_\theta J(\theta) \tag{2.5}$$

Here ϵ is a positive scalar, determining the size of the step, and is called the *learning rate*. Gradient descent converges when every element of the gradient is zero. Difficulties for this method arise in critical points, where $\nabla_\theta J(\theta) = 0$. Those occur at local/global minima, maxima and saddle points. [9]
While the gradient descent algorithm defines an implementation of the learning process, the backpropagation computes the actual gradient. In the backpropagation framework every network is viewed as a composition of functions applied to some input x (note that x here represents a data point and not the network parameters). To compute the gradient for every network parameter (weights, bias, activation function parameters, etc.) in θ, it is sufficient to calculate the partial derivative for every network parameter p: $\frac{\partial J(\theta)}{\partial p}$.

2.4.2 From Gradient Descent to Adam

For a single update, the gradient descent algorithm uses all the available data to compute the next step. A variant of the gradient descent is the stochastic gradient descent (SGD) algorithm. Here, each update is computed using only a single data point. This is beneficial in critical points. While the gradient descent has no possibilities to rely on another update value to move on the error surface, SGD updates still have the possibility to escape stagnation by relying on updates using data points that result in a gradient that is not zero. This comes at the expense of computational efficiency. A compromise between gradient descent (computational efficiency) and SGD (escaping critical points) represents the mini-batch gradient descent. Here, m data points of the data set are used to compute the next update. Another improvement is the concept of momentum. The idea behind momentum is to implement a velocity term that integrates previous gradients via exponential decay. The update rule using a batch of size m is given by the following equation where v is the velocity term:

$$\begin{aligned} v &\leftarrow \alpha v - \epsilon \nabla_\theta \left(\frac{1}{m} \sum_{i=0}^{m-1} J_i(\theta) \right) \\ \theta &\leftarrow \theta + v \end{aligned} \tag{2.6}$$

root mean squared propagation (RMSProp) is yet another variant of the gradient descent algorithm presented in [12]. It keeps an exponentially weighted average of the squares of the gradient elements. The inverse of the average is then used to scale the gradient. The update rule is then given by:

$$\begin{aligned} g &\leftarrow \frac{1}{m} \nabla_\theta \sum^{m} J_i(\theta) \\ r &\leftarrow \beta r + (1 - \beta) g * g \\ \theta &\leftarrow \theta - \frac{\alpha}{\sqrt{r}} \nabla_\theta J_i(\theta) \end{aligned} \tag{2.7}$$

Where g is the gradient given ith batch and $*$ represents element-wise multiplication. This method favors progress in more gently sloped directions of parameter space [9] avoiding sharp slopes and potential oscillations within the affected dimensions. The Adam (adaptive moment estimation) algorithm [14] now combines both approaches, momentum and scaling, in one algorithm and has proven to be a variant that performs well in various settings.

2.4.3 Regularization

Regularization is the term of various techniques, whose goal is to avoid overfitting. More precisely, it is any technique that aims to reduce the error on the test set (generalization error), but not on the training set [9].

Batch Normalization

This technique tackles the problem that in practice the networks layers are changed simultaneously, while the theoretical assumption is that all the other layers do not change. Batch normalization normalizes the given layer which reduces the problem of coordinating updates across layers [9].

Early Stopping

Early stopping [17] is a regularization method that determines at what point the training should stop. In this approach the loss or any other quantity monitoring the networks behavior is tracked. When the given quantity does not improve more than δ within N epochs the training is halted. The theory behind this approach is, that the network learns the true data distribution within M iteration, and starts overfitting after. Early stopping is now avoids overfitting by stopping the training at the appropriate time.

Cross Validation

This methods is applied in the model selection process, i.e. when tuning the system's hyperparameters. When the size of the data set is limited, then there is the danger of one system variant coincidently fitting the test set better than the other variants. To avoid that problem, the set is divided into k folds. The same system is then trained k separate times with the training set being $\{fold_1, ..., fold_{i-1}, fold_{i+1}, ..., fold_k\}$ and the test set being $fold_i$. The average performance of the estimators k variants on the test set is now compared to the one of the other estimators to select the best variant. [9]

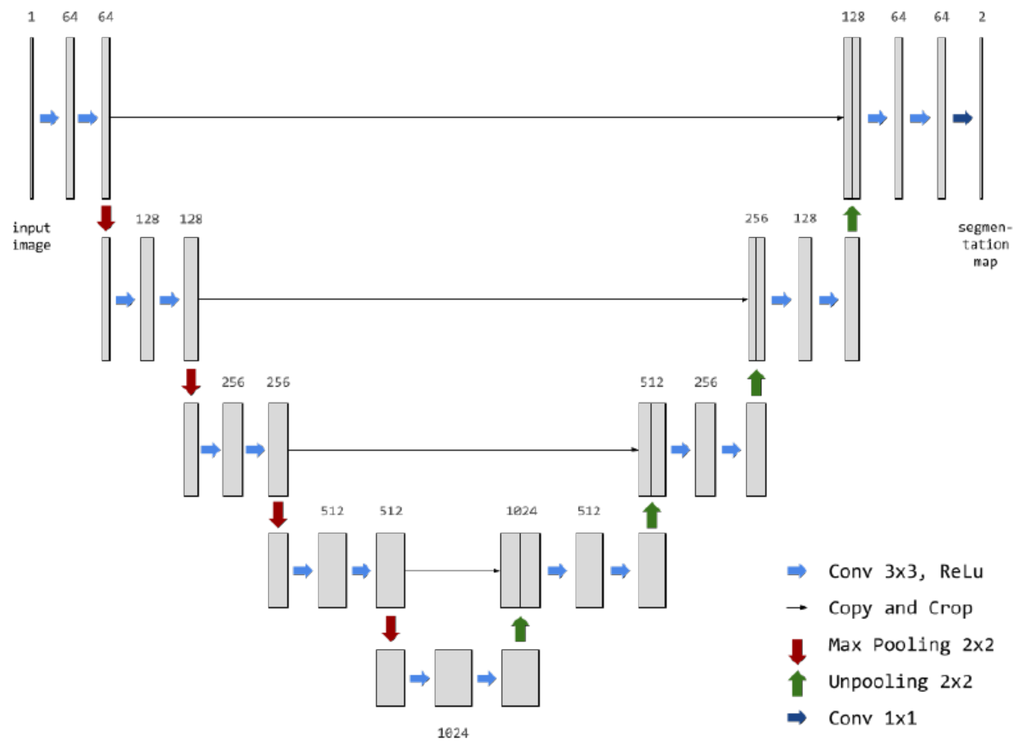


Figure 2.4: Illustration of the U-Net architecture as in [19]. The left hand side shows the contractive path, the right hand side the expansive path. This network differentiates between background and foreground, which is why 2 feature maps provide the output. It is possible to train the network to differentiate between different objects of the same class by providing borders in the training data.

2.5 The U-Net

The *U-net* is the network architecture used in this work and a popular tool in (bio-medical) image segmentation applications [19]. As such it learns a function mapping images to images. In the bio-medical domain data sets are often scarce and expensive to obtain and thus the U-net was designed with small data sets in mind. Before the U-net, images where segmented by feeding image patches to a classifier to assign a label to the center pixel. The segmentation was then obtained by combining the results from several patches. The authors' goal was to replace that technique by labeling the image pixels simultaneously and thus providing a fast image segmentation with use of context and good localization.

The U-net is a fully convolutional neural network, which means that no fully connected layer is used and it is possible, due to the convolution operation, to process images of varying size. Its main components are a contracting path followed by an expansive path. The contracting path follows the typical structure of a CNN and is composed of multiple convolutional blocks. One block is composed of two subsequent $3x3$ convolutions which use the ReLu as activation function. These are followed by a $2x2$ max pooling operation with stride 2. After each block the number of feature channels is doubled. The expansive path consists of similar convolutional blocks, only that the max pooling operation is replaced by an up-sampling operation and the number of channels are halved after each block. Additionally, each block not only receives inputs from the preceding one, but also from the corresponding block in the contracting path. The last block of the expansive path does not use up-sampling but uses $1x1$ convolutions to map the feature vectors to the output classes. See Figure 2.4 for an illustration of the original network architecture. Although the U-net was designed for small data sets, the architecture cannot fully compensate this constraint. For a successful application, intensive data augmentation is crucial. Examples for image augmentation methods are shift, rotation, deformations, and varying intensity values. The network is trained by applying the softmax function pixel-wise and using cross entropy as loss function. Adding boarders to the training set enables the network to learn the separation of overlapping objects belonging to the same class. This networks key components are the convolutional blocks, the contractive/expansive path and the copy and crop connections. Hereby the role of the convolutional blocks is to extract features from the image/feature maps. The subsequent max pooling operations compress the extracted information at the expense of the feature maps' resolution. The role of the copy and crop connections is hereby to combine the resolution of the lower-level features with the information of the higher-level features. This combination of factors allows to segment images at a high resolution.

3 State of the Art

This chapter focuses on establishing the state of the art in color spill correction. Because the subject itself is not a distinct research domain, this chapter reviews publications from bordering domains, namely: natural image matting, color decomposition and illumination decomposition.

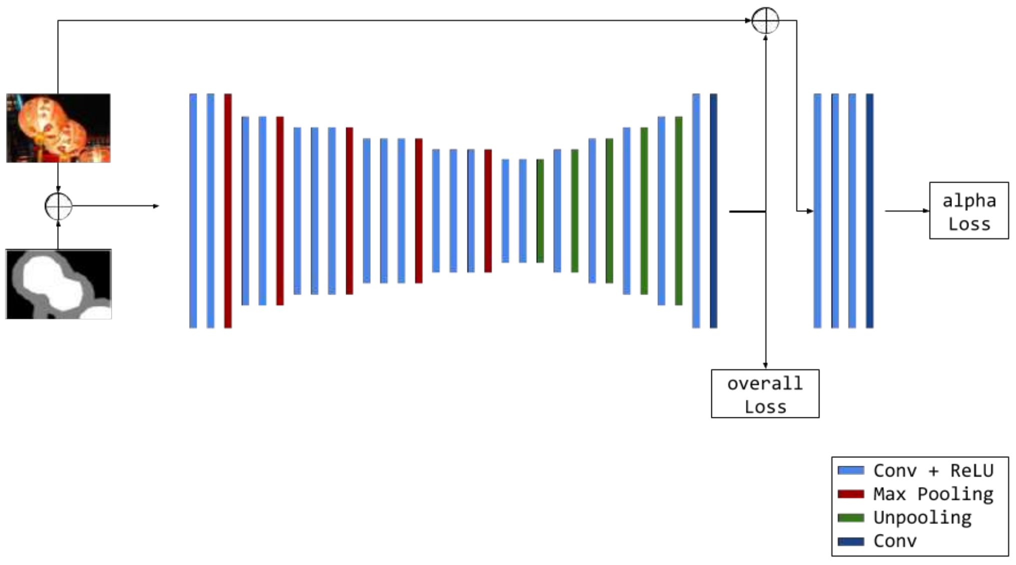


Figure 3.1: Visualization of the network structure used by Xu et al. in [23]. On the left-hand side an exemplary input consisting of an RGB image and a rough trimap is shown. This 4-channel input is fed into the encoder-decoder structure in the middle. The encoder consists of a succession of convolutional layers that use the ReLu as an activation function and max pooling layers. The encoder is followed by a decoder structure that consists of unpooling operations and, again, convolutional layers combined with the ReLu function. The final layer of the encoder-decoder network is a simple convolutional layer whose output is the 1-channel alpha matte. Because the output of the encoder-decoder network is overly smooth, the researchers concatenated the alpha matte with the original RGB image and fed the 4-channel image to the refinement network depicted on the right-hand side of the graphic which then computes the final alpha matte.

3.1 Using Deep Learning for Natural Image Matting

The first publication presented here focuses on natural image matting. This type of image matting separates designated scene objects in a natural setting from the rest of the scene. The resulting matte is then composed onto new backgrounds to obtain new images. Because it is use-case dependent on which objects should be extracted, most image matting algorithms require two inputs: the original image and the trimap. The purpose of trimaps is to label image regions as foreground, background, and unknown. Unknown regions are usually the borders between the foreground object(s) and the background. In 2017 Xu et al. [23] were the first to apply deep learning to the natural image matting problem. For most computer vision problems the greatest hurdle in applying deep learning lies in the amount of available data. This was also the case for the natural image matting problem, which Xu et al. resolved by finding a method to massively increase the amount of available training data.

This publication has been selected because it also uses deep learning in an image matting problem. The creative solution to the lack of data in the natural image matting domain also played an important role in the selection of this paper. Furthermore, the model type used in this project was inspired by this work's choice of architecture.

The author's data acquisition method uses existing image mattes and pastes them onto new backgrounds. For more details on the creation of the data set the reader is referred to Section 4.3. The authors used a CNN with an encoder-decoder structure to which a subsequent refinement stage was appended. The input to the encoder-decoder network is the RGB image and its trimap. The refinement network receives the decoder's output, the preliminary matte, and the original RGB image as input and computes the final alpha matte image. The network's structure is portrayed in more detail in Figure 3.1. The encoder-decoder network is trained separately from the refinement network. The encoder-decoder is trained on an *overall loss*, while the refinement one is trained separately on the *alpha loss* only. All losses are only applied to regions which are labeled as unkown in the trimap:

$$w(i) = \begin{cases} 1, & \text{if the } i\text{th pixel is within an unknown region in the trimap} \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

The pixel-wise alpha loss $\mathcal{L}_\alpha^i$ is based on the absolute difference between ground truth g and prediction p:

$$\mathcal{L}_\alpha^i = w^i \sqrt{(\alpha_p^i - \alpha_g^i)^2 + \epsilon^2} \tag{3.2}$$

whereof $\alpha_g^i, \alpha_p^i \in [0, 1]$ and ϵ is a small summand to avoid division through zero during backpropagation. The sum of the compositional and alpha loss forms the overall loss. According to the authors, the purpose of the compositional loss is to improve the alpha matte by taking into account how well the composite fits perceptually into the scene. In particular this affects the foreground regions ($\alpha > 0$) whereof the background colors mix with the foreground colors.

$$\mathcal{L}_{compositional}^i = w^i \sqrt{(c_p^i - c_g^i)^2 + \epsilon^2} \tag{3.3}$$

at which c_* is the RGB vector for pixel i. The overall loss then is

$$\mathcal{L}_{overall}^i = 0.5 * \mathcal{L}_{alpha}^i + 0.5 * \mathcal{L}_{compositional}^i \tag{3.4}$$

Results were reported with regard to the image matting benchmark [18], the author's composition-1k testing dataset, and a newly collected real image dataset. At the time the author's method ranked first on the alpha matting benchmark with respect to the sum of absolute differences (SAD) and second when using the mean squared error (MSE). The freely accessible code of other methods participating in the alpha matting benchmark, allowed the authors to test them on the 1k test set. The performance of the other methods on the 1k test set could not compete with the results of the authors' method. Besides evaluating the performance, the authors also tested the dependency on varying degrees of trimap dilation. While the other methods' SAD error rate shows a positive correlation with the degree of dilation, the author's method doesn't show much variance. The authors also conducted a user-study on Amazon's Mechanical Turk [1] to compare the test subjects' preference of different image matting methods with each other on real images. The participants were presented with two choices. These two images were the same composites, acquired through different methods. The result was that the majority of the users ($> 80\%$) preferred the author's method above any other one.

Besides visual examples, the authors relied mainly on the SAD and MSE metric to describe and compare the model's performance. The SAD is used when the difference between zero and nonzero elements is important. The MSE on the other hand is designed to emphasize greater differences more than small differences and (like all means) is more susceptible to outliers within the data. The results of the alpha matting benchmark are remarkable but also difficult to assess, because the test set only comprises 8 images. The 1k test set results seem even more remarkable, but the performance discrepancy between the author's and the other methods on the 1k test set is much higher than the discrepancy between the methods on the alpha matting test set. Possible reasons could be, that the deep learning system is adapted itself to this type of data sets and learned, in addition to the foreground - background separation, to detect artifacts left behind from the compositing process. Or, it is simply because the difficulty of the 1k test set is much higher. Either way, because of the overwhelming preference for the authors method, it is more likely that their method outperforms the other ones.

3.2 Color Unmixing for Foreground - Background Separation

The next work presented here directly addresses color spill in terms of the color unmixing framework. In 2016 Aksoy et al. [1] developed an interactive method to segment images and build global and local color models. These models are then used to separate the fore- from the background. The aim of this project was to develop a pipeline which targets the bottleneck in film productions, that is caused by manually attaining the alpha matte and removing color spill artifacts. Advantages of their method are, that it easily scales to any resolution and that the matte and unmixing parameters from a single frame can be propagated throughout a whole video sequence.

This publication is the only one that could be found that, which directly addresses color spill in chroma keying. It also views the color spill from an unmixing framework, showing a different perspective of the problem.

[1] a platform to outsource any tasks that can be performed virtually/online such as content moderation, survey participation, or - in this case - conducting perceptual experiments [2]

The pipeline for this unmixing process consists of three stages: The acquisition of the global color model through user interaction, the derivation of local color models and the pixel-wise color unmixing, which dismisses any spilling artifacts. The global color model is determined with the help of user interaction, in which N scribbles are placed over dominant scene colors. For each scribble, the corresponding Normal distribution $\mathcal{N}(\mu_i, \Sigma_i)$ is computed. In the second stage, for each image region a set of active color distributions is identified. This subset builds the local color model. In the third stage, this subset is then used to approximate the respective superpixel colors. Superpixels are small image regions, clustered together by merging adjacent pixels/regions until the color difference is below a certain threshold or some other stopping criterion. Superpixels are used in this scenario to speed up the computation. The approximation is based on the idea, that a color c is a mixture of underlying colors u_i. The amount each color contributes to the final color is determined by factor α_i.

$$\sum_i \alpha_i u_i = c \tag{3.5}$$

Based on this equation, the authors formulate an energy function $\mathcal{F}$ which is used to find α:

$$\mathcal{F} = \sum_i \alpha_i \mathcal{D}_i(u_i) \tag{3.6}$$

Underlying color c is substituted by the Mahalanobis distance $\mathcal{D}_i$ which measures the distance of underlying color u_i to the ith distribution. To reinforce the feasible α values, the following constraints are imposed:

$$\sum_i \alpha_i = 1, \quad \sum_i \alpha_i u_i = c, \quad and \quad \alpha_i, u_i \in [0,1] \tag{3.7}$$

The final minimization function integrates equations 3.6 and 3.7. This formulation is parallelizable to a degree at which it is possible for the user to interactively manipulate the initial scribbles to adapt the results and other parameters (f.ex. the size of the superpixels, which influence the resolution of the final result) as desired.

To evaluate their approach, the authors investigated how close two color distributions can be, before the algorithm is unable to unmix the corresponding image regions. According to the authors, the algorithm successfully unmixed the colors to a point at which it is even difficult for humans to discern the two regions. The authors' main tool in evaluating their results was to compare the algorithms' output to the results of a paid professional compositing artist [2] which used distinct professional image manipulation tools in conjunction, and individually - with the only constraint of not repainting individual pixels. The result was, that the individual application of the tools could not remove color spill and create an alpha matte. It was only by using all tools in conjunction that the artist was able to create a matte *and* remove the color spill artifacts. The results of the authors' method are objectively comparable to the ones of the compositing artist. The key difference is that the manipulation of one image takes 13 seconds for a 1080p image, not counting subsequent alterations of the input scribbles and parameters, and a total of 10 minutes if the results are refined. In comparison, the artist needed up to two hours for the same image. The authors also compared their method with other natural matting algorithms. But rather than evaluating the results on a test set, they presented an objective comparison.

[2] An image manipulation expert that specializes in removing the green-screen background and removing color spill artifacts.

The advantages of these unmixing methods are the low processing time and the subjectively superior results which seem to live up to industrial standards. Another advantage is, that the results from one frame are transferable to other frames of the same sequence through superpixel tracking. Limitations of the algorithm come to light, when the illumination changes drastically in video sequences and if an underlying color does not appear in its pure form in at least one frame. The objective evaluation of this method is difficult as no ground truth data is available. Nevertheless, the image examples the authors provided within the publication are impressive. But, as the authors compared the performance of their algorithm with the one of other natural image matting methods it would have naturally been interesting to see the performance of the unmixing algorithm on a natural image matting benchmark such as the then available alpha matting benchmark. Another area of interest would be to measure the degree of correlation two frames need for the algorithm to work. Altogether, the authors devised a very successful method that is suitable for image matting in front of simple backgrounds that according to their tests even works when the fore- and background color distributions are very similar.

3.3 Using Illumination Decomposition to Create Realistic Image Manipulations

The final publication discussed here was presented by Carroll et al. in 2011 [5]. It studies the problem of maintaining consistent interreflections upon material recoloring. The authors approach the problem in terms of *intrinsic images*, which separates each image pixel into a component due to illumination/shading and one due to reflectance. The main contribution of this work is to present a framework that decomposes the illumination component (also illumination intrinsic image) into a linear combination of secondary lighting sources (i.e. sources of interreflections). This user-guided decomposition is then used to compute the change in the illumination component when recoloring materials.

This work is presented here, because it incorporates real world assumptions into the scene. Although in terms of performance the work of Aksoy et al. is superior, this work shows a new perspective onto the color spill problem.

At first, the image is white balanced to render the procedure suitable to all lighting conditions. Next, the RGB image $I(x, y)$ is decomposed [3] into intrinsic images S (shading/illumination) and R (reflectance):

$$I(x, y) = S(x, y)R(x, y) \tag{3.8}$$

The user then selects materials in the images that contribute to indirect illumination in the scene. It follows the authors algorithm for illumination decomposition. The previously selected materials serve here as secondary lighting sources and individual pixel values of S are assumed to be a linear combination of those n sources:

$$S(x, y) = \sum_{i}^{n} T_i(x, y)b_i \tag{3.9}$$

[3]The authors use the algorithm from [3] to decompose the image.

in which the transport term T_i is a factor that regulates the effect of sources b_i. This assumption and additional constraints are then incorporated into energy function E to estimate T. E consists of three components that ensure data fidelity E_{df}, non-negativity E_{nn}, smoothness E_{sm}, and sparsity in the absolute contribution of each source E_{sp}:

$$E = E_{df} + \lambda_{nn} E_{nn} + \lambda_{sm} E_{sm} + \lambda_{sp} E_{sp} \tag{3.10}$$

At which λ are experimentally found weights.
E_{df} is derived from equation 3.9:

$$E_{df} = \left\| S - \sum_i T_i b_i \right\|^2 \tag{3.11}$$

Non-negativity is important to ensure the plausibility of the resulting factors, as there is no feasible explanation for a basis source to have a negative contribution to a pixel in intrinsic image S. E_{sp} also plays a special role as it can be modified by the user to restrict unrealistic contributions from a basis source i to pixel (x, y):

$$E_{sp} = \sum_{i,x,y} w_i(x, y) |T_i(x, y)| \tag{3.12}$$

The weights w are set by default to 1 and may be changed by the user. The last step consists of the user recoloring the sources. The change in reflectance is then computed using equation 3.9.

The authors evaluate their algorithm using the Cornell box [4]. They choose the following configuration: a red (left), green (back), and blue (right) wall and two cuboids (yellow/purple) of different sizes. The color similarity between the back and right wall is then increased (the green back wall becomes increasingly purple). When the colors are too similar, the algorithm cannot separate the indirect illumination from the material source and detected interreflections collapse with unrelated material sources. Furthermore, the authors show a series of real world images whereat one object has been recolored and the algorithm modified the reflectance.

Drawbacks of this method are, that the decomposition doesn't necessarily reconstruct the input illumination, as this constraint is not enforced in the optimization process. The user interaction is also non-trivial, because the user has to learn how to draw efficient strokes. Nevertheless, the system as per the given examples, proves to work well with real world examples. Altogether the concept of intrinsic images is important for consistent color manipulation operations for which the authors provided a feasible framework.

[4] A commonly used 3D test model originally designed to test the of interaction of light in rendering machines [10]. The basic environment consists of a white light source in the center of a white ceiling, differently colored walls, white floor and optional objects placed on the floor.

4 Data Set Creation

As the goal of this project is to train a deep learning system on a synthetic data set, one main contribution of this work is the creation of a data set that fits the problem at hand. Thus, this chapter discusses related data sets, presents the proposed method for the data set construction and highlights its properties. Several examples from the data set are provided.

4.1 The Data Acquisition Problem in Deep Learning Applications

One of the biggest challenges of supervised learning is the acquisition of sufficiently large data sets that fit the problem at hand. Particularly, the acquisition of ground truths is often a time-intensive and expensive process, causing researches to explore new ways to efficiently acquire data sets. One example would be the ImageNet project [20] whose goal is to construct a large data set for image classification. The aim is to provide, on average, 1,000 examples for the more than 100,000 classes. It is not feasible to construct this type of data set by letting the researchers annotate them manually. Instead, the researchers opted for a crowd sourcing approach. Besides annotating existing data, the usage of synthetic data is also a valid approach. The advantage of producing synthetic data is having full control over various aspects of the data (f.ex. real world data sets struggle with finding sufficient data points for rarely occurring instances) and a detailed knowledge of the ground truth. The disadvantage is, that the data set will always be biased towards its architects mental image of the problem at hand, which might not necessarily reflect real world conditions. One example of a synthetic data set is the popular Sintel MPI data set [4] where Butler et al. used the open source animated film *Sintel* [6] to construct an optical flow data set. Here, the data set was generated by re-rendering the film and computing the ground truth in the process. For optical flow data the usage of synthetic data sets is all the more advantageous, because there exists no sensor that is able to directly measure optical flow. Using exclusively synthetic data inevitably raises the question whether algorithms trained/tested on such data sets are suitable for the application to images/films of natural scenes. In the case of the Sintel MPI data set, Butler et al. analyzed the data set and found that its statistics matched those of natural scenes and concluded that this suggest that the data set is sufficiently representative of natural movies. The work of Tremblay et al. [22] also suggest that synthetic data is indeed suitable to train an algorithm for real world problems. Here, the researchers used virtual 3D models which were rendered with a combination of random textures, lighting angles, camera angels, colors, background images etc. to construct a data set for object detection. The CNN trained on this data set achieved competitive results. Additional fine-tuning on non-synthetic data caused the network to outperform the version trained on natural images only. The following subsection gives a quick overview of different existing data sets related to image matting before finally describing the data set creation process applied in this work.

4.2 The Problem of Creating a Ground Truth for Color Spill

Color spill is not only present in artificial scenes, but also in natural scenes where the effect is much more subtle. Here the lighting interactions are between various components of the scene and it is physically impossible to encounter a scene without those interactions. This is also the root of the problem of creating real-world ground truth data for color spill. In the real world it is possible for an object to change locations and therefore possible to change the foreground background lighting interactions. But it is not possible to remove those interactions completely: the only alternative is to substitute them. Even if someone where to decide that some sort of lighting substitute would produce a suitable ground truth, it would be still very difficult to swap the entire background and take a second picture without loosing any precision. In theory it would also be possible to manually create a ground truth by repainting pictures, but the drawback would be that it introduces a bias towards the artist perception and that it would be time and cost intensive. The main drawback with these methods is, however, that it is not feasible to apply them to attain a sufficient amount of data points. As shown above, this is a problem that is especially prominent in the field of deep learning area. These are the reasons this project opted for the creation of a synthetic instead data set. Here, the lighting conditions are simulated and it is possible render the exact same image with and without background, thus removing the color spill. The feasibility of using synthetic data has already been demonstrated by Tremblay et al., which lead the author to believe, that this is the best method for generating a ground truth.

4.3 Data Sets Related to the Color Spill Problem

The most popular image matting benchmark and data set was created by Rhemann et. al in 2009 [18]. Although previous data sets existed, the authors decided to build their own data set because of the previous data sets' quality and their bias towards certain types of algorithms. Additionally none of the previous data sets were re-used in subsequent works and thus failed to establish themselves as a standard. The data set consists of 35 images. 27 of those are available for the training or adjustment of algorithms and 8 are reserved to compute the final scores publicized in the benchmark evaluation results. All images were taken in an indoor setting. The training set is composed of 23 images that show objects photographed in front of a monitor displaying natural images and 5 images that depict objects in front of a natural (indoor) scene. The authors describe the testing set to be considerably more challenging than the training set. The ground truth was acquired through triangulation. To do so the authors took one picture of the complete scene, four pictures of the foreground object with a monitor between fore- and background that displayed black, red, green, and blue respectively, and one picture of the background without the foreground object. The images were taken with different depth of fields and contain color ambiguity. The foreground objects were selected such that the set contained objects with hard and soft boundaries, objects with translucent/transparent areas, objects with varying topology, and objects with varying boundary width. As most matting algorithms need user input in form of a trimap, multiple trimaps were constructed by dilating the original image to different extends and by hand-drawing one for every test case.

Another popular data set was constructed in the course of the earlier presented work (see Section 3) by Xu et al. in 2017 [23]. Its construction was motivated by the poor generalization of the above data set to natural scenes. The authors merged two existing data sets and added new images for

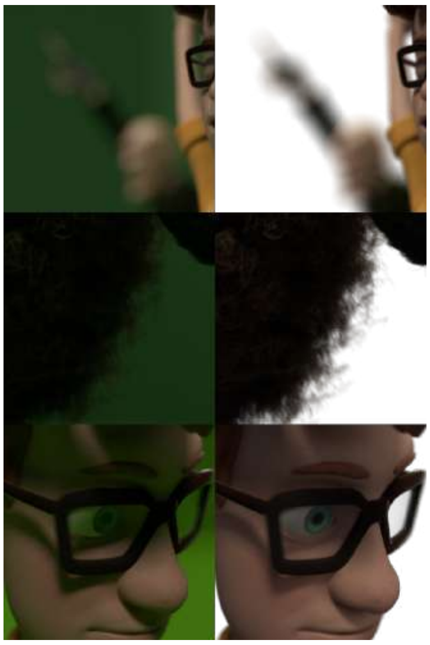


Figure 4.1: Illustrations of phenomenons that where taken into account for the image set creation. The presented images are only zoomed in sections. The left side shows spilled images, while the corresponding ground truth is presented on the right side: The first row shows blur, simulated using the depth of field. The following row shows translucent areas simulated through hair. The third row shows interreflections.

which they manually created alpha mattes. They compiled the data from the above described data set and from the video matting data set presented in [21]. Here, the authors extracted every fifth frame from each video. The authors augmented the existing data by randomly sampling background images from the MS COCO data set and composing them with the extracted foregrounds from their own data set. This resulted in a training set with 493 unique foreground objects which were combined with 100 different background images, leading to a total of 49,300 images. The testing set consists of 50 unique foreground objects that were combined with 20 sampled background images which lead to a total of 1,000 testing images. The authors did not specify how individual video frames contribute to the unique foreground object count.

However, none of these data sets cover color spill, as they concentrate on the matte problem. They only relate to this work, because the trained system not only performs color spill correction, but also creates an alpha matte. The comparison of these work's performance with this project would have been desirable, but unfortunately none of these benchmarks covers uni-colored backgrounds or addresses specifically the color spill problem.

4.4 Proposed Method for Data Set Construction

This work's data set was created using Blender version 2.79, which is a free and open 3D creation suite. The data set contains two types of data. One type of images shows automatically generated „random forms "which are nothing more than a randomly distorted spheres. The other type of images shows freely available 3D character model from the Blender Cloud.

All images were saved in the RGBA format with an image size of 512x512 pixels. The background formed the inside of a cube and every image was rendered trice: with a green and blue background for the spilled images and without background for the ground truth. The green shade was randomly chosen with $r \in [0, .1), g \in [.2, .25), b \in [0, .1)$. The blue shade was chosen similarly only that $r \in [0, .03), g \in [.05, .1), b \in [0.2, .25)$. The ground truth has been created by removing the background scene and rendering it with an alpha map. This removes any illumination interaction of the foreground and background, which has been identified as the source of color spill.

The spheres are distorted by repeatedly applying one of the following operations to a randomly selected face:

- rotating it
- moving it along its normal vector

Additionally, images from the Blender Cloud were used to derive a texture that contain a bump and displacement map, a base color, a specularity, and roughness map. The images are available under the *Creative Commons Attribution 4.0* (see terms and conditions of the blender cloud). The images were generated using a camera that completes one circulation around the sphere forming a video sequence of 100 frames. Throughout the process the depth of field and distance from the sphere were varied. Altogether this method resulted in a total of 7100 data points rendered with 71 different textures and forms.

The images with 3D models use 8 models of „animate"objects as well as some inanimate objects to increase the complexity of the foreground scene. Theses models are all available on the Blender Cloud under different variations of the Creative Commons license (see Table 8.3 in the appendix) and are free for modification and redistribution. The models were created as part of the Blender Open Projects. While they were already available, a considerable amount of work went into creating different poses for them. Additional work went into modifying the models to restrict the final render time without decreasing the quality of the final result. Although these precautions were taken, the render time of 100 images could still amount to more than 15h. For some scenes the depth of field has been varied, but more often the focus has been shifted between objects and the camera has been animated to follow various paths. Altogether 30 different scenes (one scene is one image sequence/video that shows the foreground objects from different distances and angles) were rendered which resulted in ca 2.300 data points.

As mentioned before, this work's main topic is color spill but the problem formulation in fact addresses three distinct phenomenons:

- blur
- translucent areas of the foreground object
- interreflections between back- and foreground.

It is only the last phenomenon, that causes the color spill. But nevertheless, all three points have been addressed in the data set.
In the proposed method, blur has been simulated by varying depths of fields. Translucent areas of foreground objects are present in models whose hair has been modeled through individual strands. Interreflections naturally occur in all images. For examples from the data set, the reader is referred to Figure 4.1.

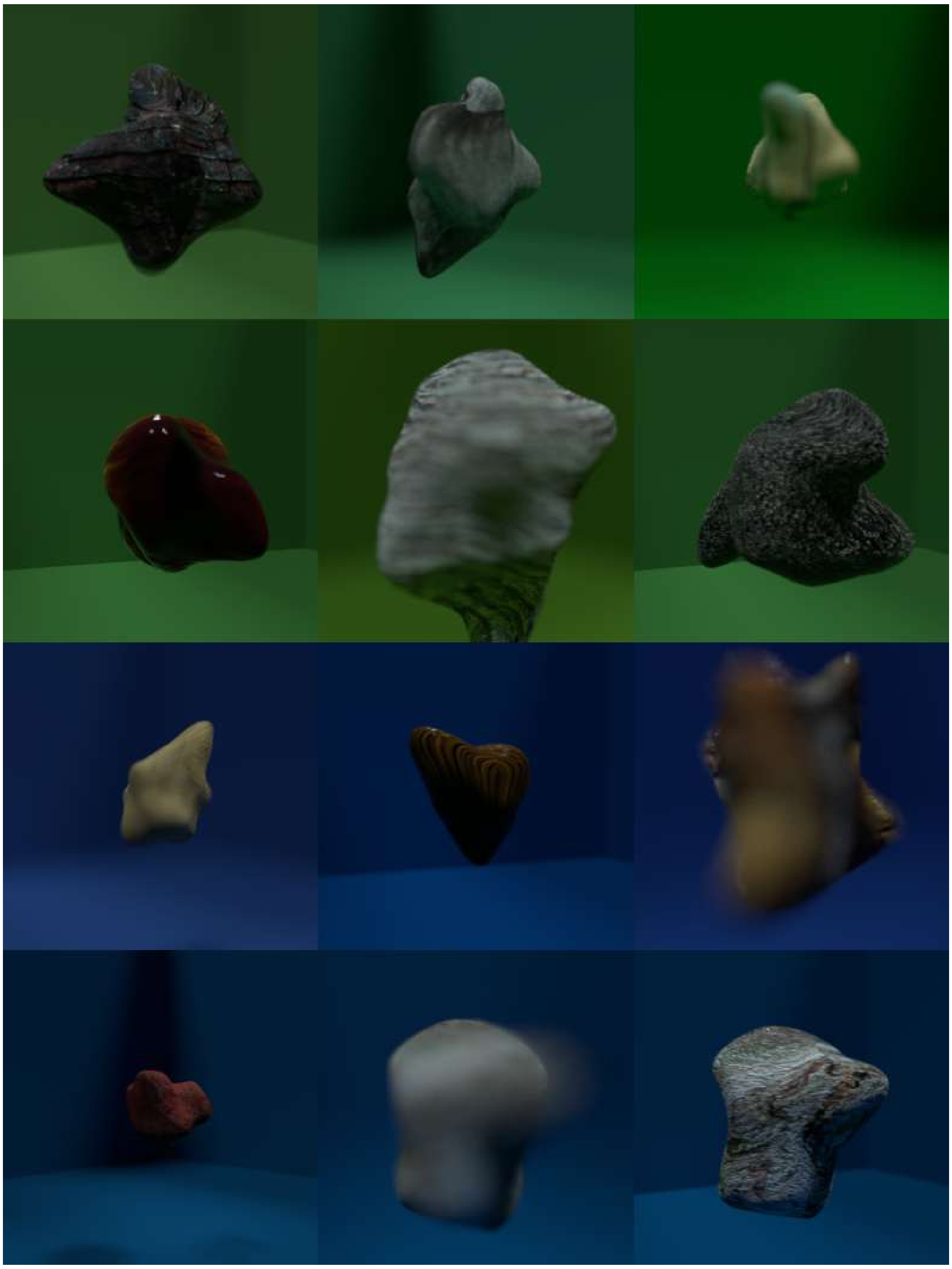


Figure 4.2: Example images of the randomly perturbed sphere data. It is noteworthy that the gradient of the background drastically varies in some images (see image (0,1) and (3,0)). Furthermore the readers attention is directed to the varying depth of field f. ex. (image (0,0) versus (3, 1)) and specularity (image (0, 1) versus (0, 2)). As can be seen the forms of the different objects vary greatly as well as the foreground - background ratio.

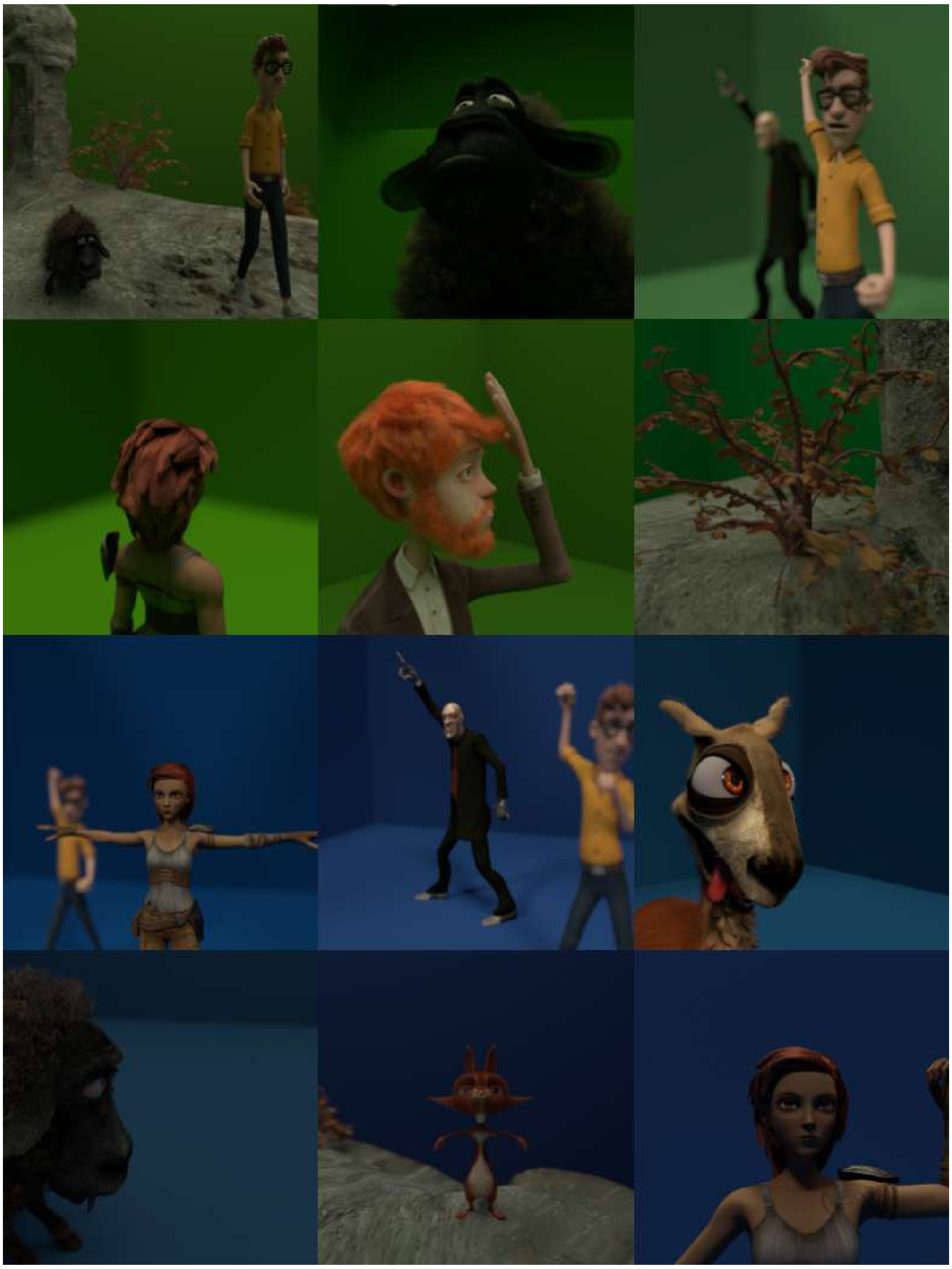


Figure 4.3: Examples from the data set. They do not reflect the frequency of model usage within the data sets. For the scene construction special care has been taken to include examples with partially overlapping objects (image (0,0)), non-overlapping objects (image (2,1)), varying degrees of background coverage (image (3,0) vs (0,0)), and disconnected background areas (image (1,2)).

5 Automated Color Spill Removal

The original research question of this work is whether synthetic data and it's computed ground truth provide enough information for a deep learning system to learn solving the color spill problem and how well the results generalize to real world data. Thus, this chapter describes the second contribution of this project, namely the production of a deep learning based spill suppression system. It first describes the conditions within which the project was realized, and then general model architecture. Subsequently the developed custom loss and metric are discussed. The last section contains the description of a small psycho-physiological experiment that was conducted to help determine what loss value indicate a good network performance.

5.1 Platform

To realize this project the Keras framework was used. It is a neural networks API written in Python. While different backends are available, this project opted for the TensorFlow one. TensorFlow is a free library for dataflow and differentiable programming. The code was programmed in a Jupyter notebook environment which was run on either a local machine or Google Colaboratory. Colaboratory is a free research tool for machine learning, which provides free hardware resources with GPU and TPU acceleration.

5.2 Model Architecture

Here, the used model architecture is a variation of the earlier presented U-Net (see Section 2.5). This choice of architecture was inspired by the work of Xu et. al (see Chapter 3) which also used an architecture containing an encoder and decoder component or, in terms of the U-Net authors, an contractive and expansive path. In contrast to the U-Net, Xu et. al only used one connection that skipped layers, allowing an immediate flow of information between source and destination.

The implemented version also consists of a contractive and an expansive pathway. Both are build from convolutional blocks consisting of one convolutional layer, followed by a batch normalization layer, another convolutional layer and a second batch normalization layer. All convolutional layers, except the output layer, use the ReLu as activation function. The input layer accepts matrices with any height and width and 3 channels. The input's scalars must be scaled to be in $[0, 1]$. The contractive path consists of a series of convolutional blocks to which a max pooling layer (with a pooling size of 2) is added. A simple convolutional block is then added and before starting the expansive path. The building blocks of the expansive path consist of an upsampling layer (of size 2), which is then followed by a concatenation layer, which concatenates the output from a down block to the output of the upsampling layer. Another convolutional block follows. There are as many contractive blocks as expansive blocks, each pair being a so called layer. The final expansive block is followed by a convolutional layer, which uses the sigmoid as activation function. The advantage of the sigmoid is, that it naturally restricts the output to be in [0, 1] All convolutional layers use kernels by the size of 3x3. The initial channel number is treated as a hyperparameter, as well as the channel factor c that determines by how many channels the next block has. In the contractive path the number of channels is halved for each level.

5.3 Loss

The loss used here is an adaptation of the root mean square error. The output of the network is a RGBA image scaled to be in [0, 1]. It is composed of two parts: one part evaluates the alpha channel while the other one evaluates the goodness of the RGB estimate. For the prediction $\hat{y}$ and the corresponding ground truth y, the alpha error for one pixel is given by

$$\alpha_{err} = (y^{\alpha} - \hat{y}^{\alpha})^2 \tag{5.1}$$

On the other hand, the RGB error for one pixel is given by:

$$rgb_{err} = y^{\alpha}((y^r - \hat{y}^r)^2 + (y^g - \hat{y}^g)^2 + (y^b - \hat{y}^b)^2) \tag{5.2}$$

The final loss, from here on referred to as weighted mean squared error (WMSE) for pixel p is then

$$wmse(p) = \alpha_{err} + rgb_{err} \tag{5.3}$$

The intend by formulating the WMSE as such is to enforce the idea that the given color spill problem entails three different aspects: the separation of fore- and background, the unmixing of border pixels, and the color correction. While the first one purely concerns the alpha channel, the problem with the latter two is that they manifest in the same way (by having a green/blue tinge) but should be handled differently. While translucent border pixels should be unmixed through the alpha channel, opaque pixels should be corrected by modifying the RGB value.

5.4 Metric

The accuracy a for image x with M pixels, is here defined as the percentage of correct pixels. A pixel i is categorized as correct, if its WMSE value doesn't deviate more than t from the ground truth value:

$$correct(i) = \begin{cases} 1 & \text{if } abs(wmse(p)) \leq t \\ 0 & \text{else} \end{cases} \tag{5.4}$$

$$a(x) = \frac{1}{M} \sum_{i=1}^{M} correct(i) \tag{5.5}$$

5.5 Training and Model Selection

To train a deep learning system one does not only need a model but also an optimizer. This work makes use of the earlier described ADAM optimizer (see Section 2.4.2), which is a standard choice for many deep learning applications. Early stopping was used to determine the length of the training. Additionally, data augmentation techniques were applied during the training phase to promote robustness of the network. These techniques included applying the following operations: shift of the image along x and y axis, shear of the image, flipping the image along its horizontal axis, and rotating the image. The set from which the rotation value etc. were selected are listed in the appendix' Table 8.2. The values were determined by the author's judgment of what values would add variety to the training data without distorting the training images. To avoid overfitting early stopping was applied as a method to determine when training should stop. The employed early stopping parameters were also selected based on the authors previous experience in training test models. The available training data was partitioned into a training (80%) and validation set (20%). This test set was used to evaluate the model's performance for early stopping. If the loss didn't improve for more than 0.001 within ten epochs then the model version with the best score during the training time was returned.

Up to this point the system's description leaves a lot of room for the final hyperparameters used to train the network. Those hyperparameters were selected in a separate process: 4-fold cross validation with "random search„. According to Goodfellow et. al [9], random search is the most effective way, given limited time and resources, to find a good model configuration. In the random search algorithm, the hyperparameters are sampled from a predefined set (for the sampling process this work used the uniform distribution) with the limitation imposed on the model to not exceed 80,000 parameters in total. Previous test models had a similar parameter count and performed satisfyingly, therefore this value was chosen. A higher parameter count would've also exceeded the capabilities of the available hardware. In this search process 20 models were first trained on $2,000$ data points, that were part of the perturbed spheres trainings set. They were then trained on all of the 3D model training set's images with 4-fold cross validation. The reason to only use a part of the perturbed spheres' data was to speed up the training process. The first training session was performed with the intention of having a „pre-trained "model available for the final cross validation training. Cross validation itself was chosen as a training method to avoid selecting a model which is by chance compatible with the selected data set. To avoid having strongly correlated images in the cross-correlation training and test set, the images for every fold were randomly selected from distinct sets of scenes. The best model was then selected by evaluating its mean performance on every fold and then retrained from scratch to obtain the final model.

5.6 Determining an Acceptable Error Value

An experiment was included into this project as an additional mean to assess the quality of the results. The results of this deep learning system are evaluated using the custom loss function 5.3 and accuracy. Formally, the accuracy is the portion of correct samples for which the model produces the correct output. But because it is very difficult for a regression system to predict the exact value in continuous space, one pixel counts as a correct sample if it deviates no more than t from the ground truth value (see equation 5.5). But what is a good value for t? In order to shed light on this question, a simple psycho-physiological experiment has been conducted. In the experiment the subjects are presented with one gray image which is gradually blended with an image of color c. The task is now to press a button as soon as the subject recognizes a change in color.

5.6.1 Experimental Setup

The question that the experiment aims to answer is how much can a color deviate before the subject notices a difference. To do so the experiment has been designed as follows:

The subject is tested in front of a computer screen and has to perform two tasks, which allow to measure the deviance and also assess the quality of the answer. The first task, which measures the deviance ,is a reaction test were a gray rectangle is displayed. The rectangle is then slowly blended with a differently colored image. The subject is now asked to press the space key whenever she/he perceives a color change. To avoid that the subjects react to certain time intervals instead of concentrating on the perceived color change, the change ratio of blend factor α is varied between 0.004 and 0.007 and an random initial offset between 500 and 1500ms is added. To make it easier for the subject to focus, a black cross is centered on each image. The second task is a forced choice task. Here, the blended image is shown at the top of the screen and four images are displayed in a row below that image. The subject is then asked which image represents best the color change. The purpose of this second task is to evaluate the subjects' confidence for doing so. Screen shots of the experiment's implementation can be found in the appendix (see Figure 8.1). The experiment consists of 50 trials, to avoid fatigue.

The reason for using this experimental setup is to investigate what change in color goes by unnoticed in human perception. While it would be possible to blend ground truth with spilled images, instead of using gray and uni-colored ones, they would not be as conclusive. Firstly, the loss would produce many different loss values. It would not be clear in what image region the subject detected a color change and thus it is not clear which loss value to use. Secondly, it would introduce many dependent variables that are addressed here collectively as image complexity. Examples for these variables are texture, contrast, color intensity, number of objects, etc. Thus, it was decided to keep the image complexity as simple as possible and use the most simple setting for the color change detection. The results acquired under these circumstances would then also hold for images with higher complexity.

5.6.2 Experimental Results

min, max blend value	$[0.03, 0.59]$
mean blend	0.2
variance of the blend value	0.01
average percentage of correct choices	24%

Table 5.1: The above table shows simple statistics regarding the blend recordings and one regarding the average percentage of correct choices.

Figure 5.1 shows the blend-value the interpolated image had when the subjects indicated that they perceived a color change. As the figure indicates, the experiment has only been performed with 4 subjects. According to [7] this is a sufficient number for any psychophysical experiment, as long

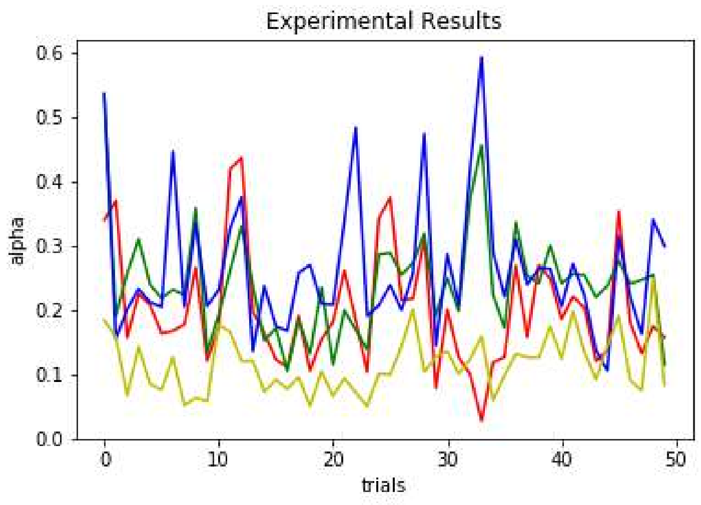


Figure 5.1: All alpha values (y-axis), ordered by trial (x-axis), shown for individual subjects (graphs).

as the number of repetitions is high enough. Table 5.1 shows simple point statistics that help to evaluate the experiment.
The mean blend value of 0.2 shows that the experiment did not measure the desired quantity. Figure 8.2 in the appendix shows clearly that the difference between the original image and its interpolated version is to large to interpret the acquired blend value as a true approximation of the targeted quantity. The computed variance of the alpha values excludes the possibility, that the mean has been influenced by outliers. It seems that the experiment measured another quantity than the targeted one. Which means that the experiment introduced an unknown dependent variable that obstructed the measurement.
The results of the forced choice task show that the subjects performance in this task was not better than randomly choosing one of the four options.

Altogether, the results show that the experiment contains a fundamental flow and needs to be redesigned. One possibility that negatively affected the results is the used screen: Although HD, its contrast might distort the result. This would set the management of the screen settings as a requirement for future experiments in this area. Another possibility would be, that the gradual change lead the subjects to compare the displayed image with a state between the current one and the initial state, instead of using only the initial state as a reference. A possible solution to that would be to only blend only one half of the gray image with a colored one, instead of the whole image. Another point, which already stood out before evaluating the data was, that the forced choice task might have encouraged the subjects to wait a little longer in the reaction task to be able to „correctly“answer the forced choice task. This would be resolved by removing this task.

To sum up, if one where to design another experiment to measure the minimum color change necessary for a human to be able to detect it, the experimenter would have to take the following points into consideration:

- screen setting management
- direct comparison of blended state versus initial state

- additional tasks might affect the experimental results.

6 Results

This chapter presents and discusses the results of the color spill correction system. As visualizations are involved, selected files have been uploaded for viewing comfort to this google drive folder. Links to individual images can be found within the captions. The reader is advised to switch to the digital version, because image differences are often subtle.

6.1 Baseline

To properly evaluate the trained system, this section aims to establish a baseline. It does so by showing statistics of the WMSE loss from Equation 5.3 and visualizing it's behavior on images from the data set.

6.1.1 Loss Behavior: Ground Truth versus Spilled Image

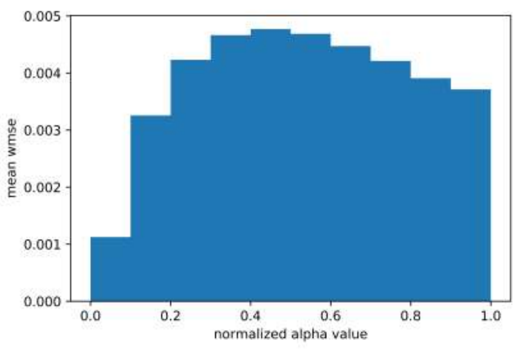


Figure 6.1: Histogram plotting the normalized alpha value against the mean WMSE using the spill with identity mapping (SIM) system to replace the prediction of a trained network. Only foreground pixels are taken into account and 50 data points from the data set have been used to compute the graph.

Figure 6.1 shows an histogram plotting the normalized alpha value against the mean WMSE. This plot has been computed taking only foreground pixels into account. The WMSE has been computed with the ground truth image and a substitute prediction. Here, the ground truth's alpha channel is concatenated with the spill image's RGB data to replace the prediction. This system will be from here on be referred to as SIM. As spill suppression is the primary concern of this project, the usage of such a system allows to emphasize the spill component in computed losses. The comparison with the SIM system equals to one where the system is capable of predicting a perfect alpha with

and reproduces the input RGB data.
Thus, to outperform the SIM system, the deep learning network's loss has to be at least within the range of $0.001, 0.004$ and preferably below 0.001. To measure the accuracy (see Equation 5.5) a fitting threshold is needed to define at what point the quality of a pixel's prediction is sufficient to be counted as a perfect prediction. Because the experiment to determine such a value was not successful, this evaluation chooses a rather non-restrictive threshold and a stricter one. The former one is determined as 0.001 (see minimal loss value to outperform the SIM system), while the latter one is determined to be a tenth of the non-restrictive's magnitude.

Figure 6.2 shows the difference between using the alpha channel as a weight in the WMSE or not weightening the loss. It shows that the weighting reduces the overall magnitude of the loss. But also the intended effect of increasing the relative contribution of opaque pixels. Note that this figure only visualizes the weighted RGB part of Equation 5.3 (see Equation 5.2). Without weighting the RGB loss, the focus of the RGB loss would be on mixed pixels (as the figure confirms) instead of spilled pixels. Mixed pixels should be corrected through unmixing, i.e. by adjusting the alpha channel, while the main target of the RGB loss are the spilled pixels, which should be corrected by changing the RGB values accordingly.

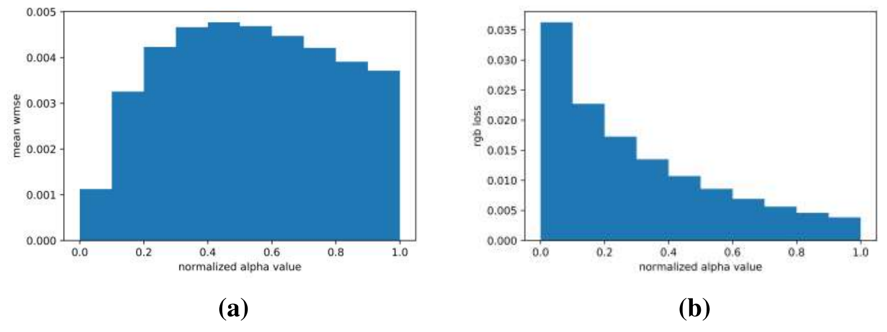
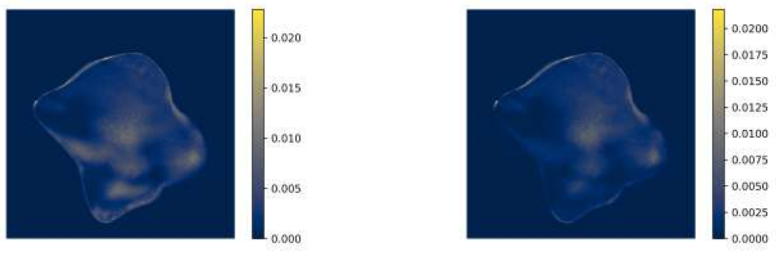


Figure 6.2: Figure (a) shows an histogram plotting the normalized alpha value against the mean WMSE. Figure (b) shows the mean loss without the alpha weighting. For both plots the predictions have been computed with the SIM system. Remarkable are the shifts in the distribution and the difference in the overall loss magnitude.

6.1.2 Loss Visualization



(a) **(b)**

Figure 6.3: Figure (a) shows the loss computed for an image with a blue background from the random forms data set. Figure (b) shows the loss for the same image, but with a green background. The loss for both images has been computed using the ground truth and the prediction of the SIM system. An online version of the visualizations can be found here.

Figure 6.3 shows the loss of SIM predictions for a blue and green screen image from the random forms set. Although the overall loss topology is for both backgrounds roughly the same, there are still visible differences in the loss' magnitude. A possible explanation for that phenomenon would be the difference in lightness between the green an blue background color.

6.2 Model Optimization

Although the purpose of the cross validation process is to find the best model configuration, it is worthwhile to not only present the final model, but examine the cross validation results. Between the trained systems, the minimal loss amounts to 0.273, while the maximal one amounts to 0.298. The mean loss equals to 0.278.
It is noted, that the minimal loss is not within the desired range of $[0.001, 0.004]$. This is an expected outcome as computation time and available data where restricted for cross validation training sessions. The GPU availability is restricted on google Colabatory. Dependent on traffic other other factors, the effective computation time ranges from 3 to 12 hours a day. The full training of one model with approximately 70.000 parameters may take 15 hours. Given these limitations, it is not feasible to train a decent amount of models with the full training set. Despite these restrictions, the cross validation process was still implemented, because it is assumed that a more suitable model version has a better performance that already shows in early learning stages. It is assumed that it does not first perform worse and, then show a sudden increase in performance.

Figure 6.4 shows the mean WMSE of the individual folds' test sets plotted against the tuned model parameters. These plots are not suited to investigate systematic dependences between individual parameters and performance. For this type of comparison all other model parameters should stay the same and not vary. But they nevertheless show that there is no individual parameter, which strongly correlates with the models performance. One explanation could be that the searched spaces

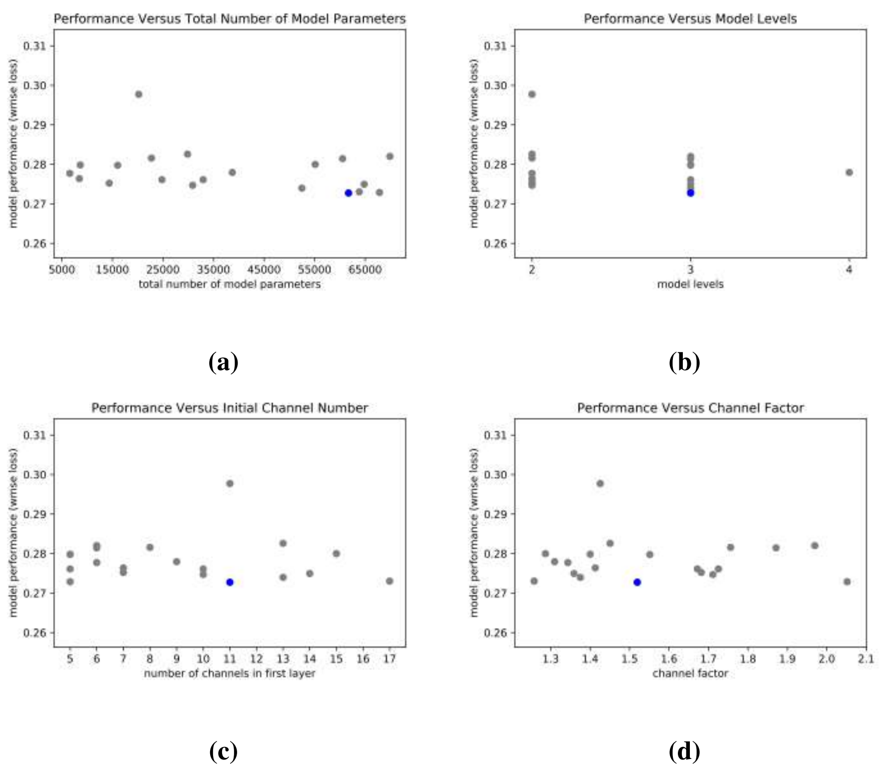


Figure 6.4: The above figures show the performance of the individual in dependence of a selection of parameters, that were tuned in the cross validation process. The performance is here the mean WMSE of the individual folds' test sets. The best model is highlighted in blue.

are not large enough to uncover such correlations. Another one could be that the earlier named assumption is invalid and that the models' performance at that stage is roughly the same for all models and only longer training sessions can serve as a basis for a performance evaluation and comparison. Despite these results, the model with the lowest loss was used. Figure 6.5 presents the architecture of the chosen model.

6.3 System Evaluation

Figure 6.5 shows the final architecture chosen through the cross validation process. This model has been trained in two variations:

1. blue spill CNN
2. green spill CNN

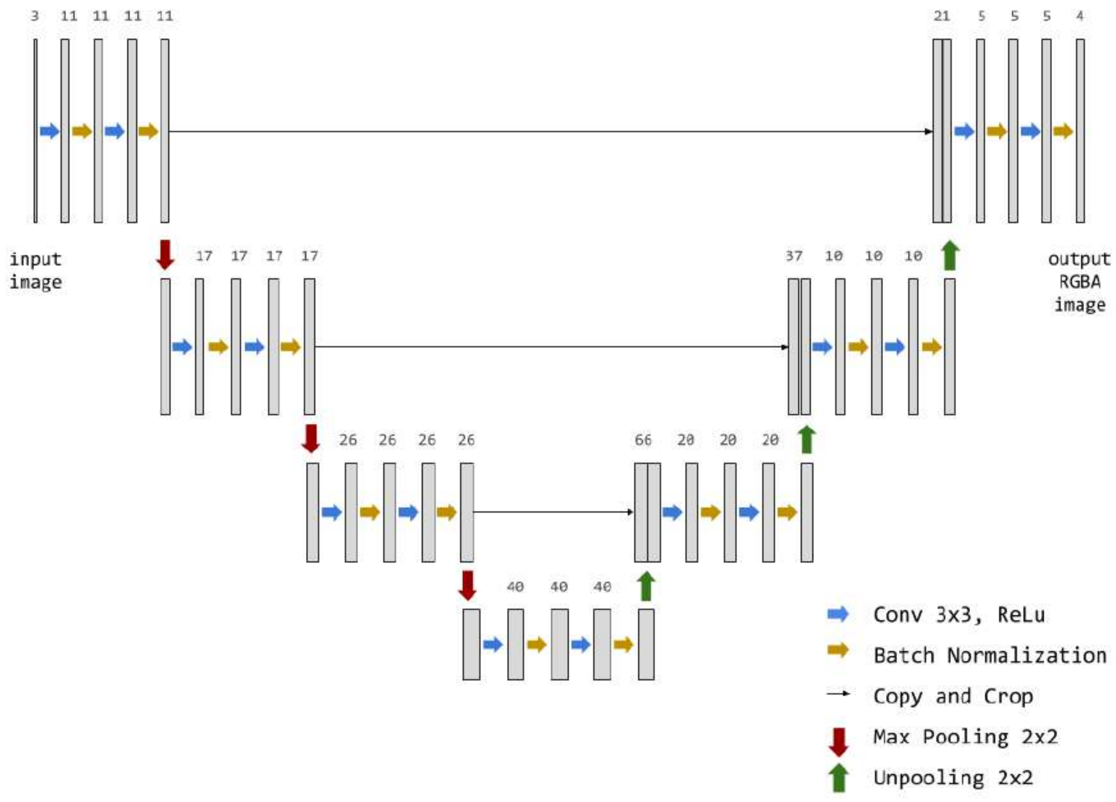


Figure 6.5: The final architecture chosen though the cross validation process. It has 11 channels in the first convolutional layer, 3 levels, and the channel factor was 1.5. These parameters lead to a model with a receptive field size of 113x113. A detailed visualization of the model's architecture can be found online.

This section evaluates their performances with respect to the achieved loss and accuracy on the validation set and furthermore shows visual examples to put the results into perspective. Because the experiment intended to find an appropriate threshold for wmse based accuracy a (see Equation 5.4) failed, the accuracy has been computed with two thresholds: 0.001 and 0.0001. The thresholds have been selected with regard to the mean WMSE per alpha bin analysis shown in Figure 6.1.

6.3.1 Numeric Evaluation

The blue spill model is the only version that has been trained from scratch. It was first trained on the random forms and then on the 3D model trainings set and only on blue spill data. The random forms training lasted for 1.5k epochs and the subsequent 3D model training lasted for 2k epochs. The epoch numbers where manually chosen and dependent on the stagnation of the training. The partitioning of the training has the advantage, that it allows to study the influence of different data types on the model's performance. The original idea behind this partition was that the simple data set would first allow the network to learn the basic principles of color suppression and image matting, before adapting to more difficult situations. It was expected that this procedure leads to a

	RF BSP	BSP Model	RF GSP	GSP Model
training set loss	0.00104	0.00048	0.00149	0.00052
evaluation set loss	0.00776	0.00211	0.01021	0.00092
alpha loss	0.00696	0.00158	0.00984	0.00080
accuracy with t=0.001	68.86%	83.35%	45.68%	80.42%
accuracy with t=0.0001	42.86%	72.49%	20.08%	65.86%

Table 6.1: Table summarizing the performance of the trained networks. Here, RF refers to models only trained on the random forms set, while BSP/GSP model refers to the fully trained model (random forms and 3d model set) which where trained on blue or green spill data respectively.

lower overall error. This hypothesis could not be fully tested but a small test indicated that it does not matter whether the network receives a sequential or all-at-once type of training. But because of the named advantages the training structure has been kept.
From here on, models trained on blue/green spill data are referred to as **BSP/GSP** models respectively. Additionally, models trained on the random forms data are referred to as **RF** models, while models that have been additionally trained on 3D models are described as **fully trained**.
The GSP model on the other hand has been trained using the RF BSP model. With this model as a base, it was then trained on the RF GSP data for 0.5k epochs, before being trained on the 3D model data for another 2k epochs
Table 6.1 summarizes different performance measured of the trained models. Overall, the fully trained BSP model attains the lowest evaluation loss. And only the fully trained models' evaluation loss is within/below the desired range of [0.001, 0.004]. An expected phenomenon, is the evaluation loss decrease for the fully trained models in comparison to the RF models.
For most models the gap between evaluation loss and the training loss is within bounds, except for the one of the fully BSP model. Here, the evaluation loss is roughly four times as big as the training loss. This suggests that the model has poor generalization abilities. The evaluation data set contains two 3D models that have not been used in the trainings set: a caterpillar and a human. The caterpillar is of abstract form and has various body colors that are not strongly correlated to distinct surface structures (example image). Computing the evaluation loss without the caterpillar data decreases the loss from 0.00211 to 0.00035. This indicates, that the fully trained BSP model suffers from overfitting. The human 3D model might be similar enough to models used in the trainings set for the model to perform well, while the caterpillar differs too much from the training data. The fully trained GSP model is probably not overfitted, because it has been first trained on a blue data (a similar but not exactly the same problem) before being readjusted to its true task.
Remarkable is also the high alpha loss value for all model variations. For the RF models, the alpha loss makes up more than 90% of the overall evaluation loss, while accounts for approximately 75%(BSP)/86%(GSP) of the fully trained models. Ideally, the alpha loss and weighted RGB loss would be balanced.
The accuracy of all versions and thresholds is limited and leaves room for improvement.

In conclusion, the all the fully trained models' performance is above the established baseline. In contrast to the fully trained GSP model, the BSP one shows symptoms of overfitting. The most important finding regarding the custom loss is the imbalance of its alpha loss and spill suppression components.

6.3.2 Visual Evaluation

This section shows and analyses a selected set of visual examples of the trained model's prediction on synthetic and natural data. Besides the shown predictions, many more have been computed and the reader is invited to explore them in this online directory.

The following pages show first a series of figures depicting the prediction behavior of the fully trained networks with the main focus on the GSP model. Their captions describe the networks behavior in detail. The subsequent passage then summarizes these findings.

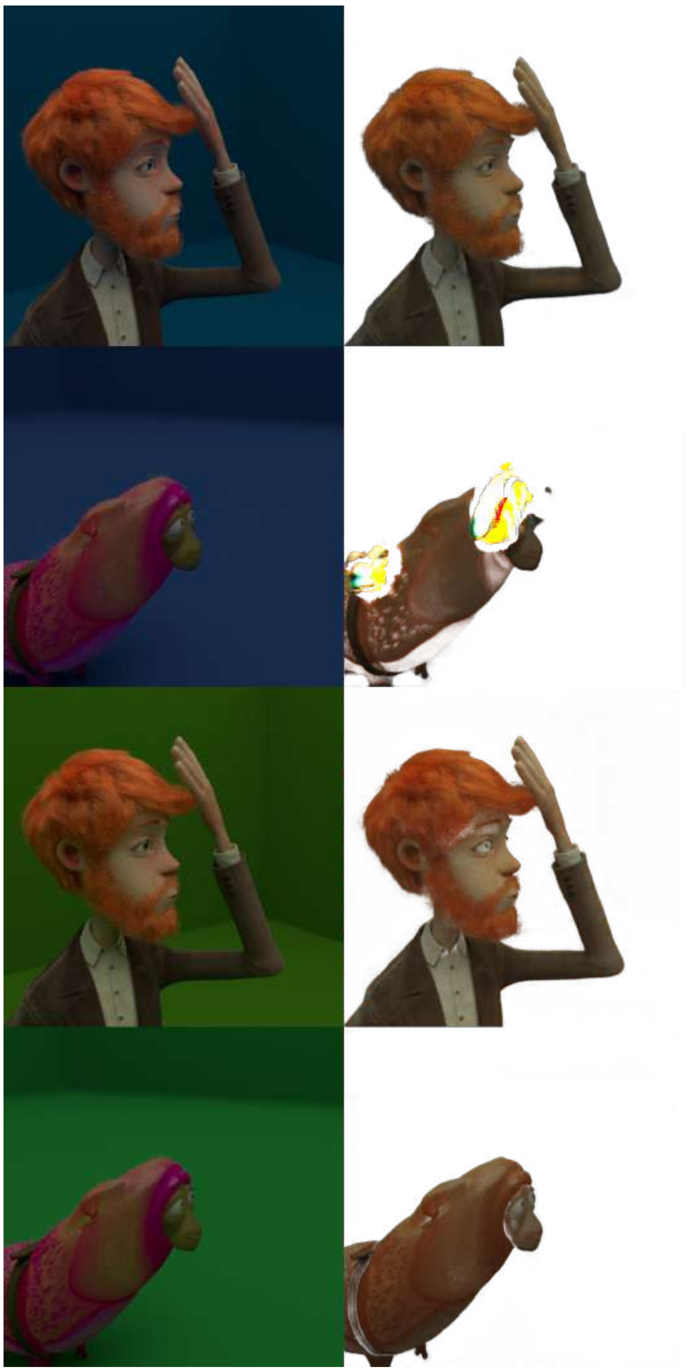
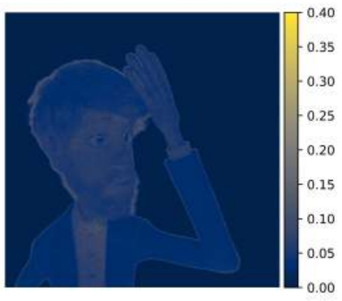
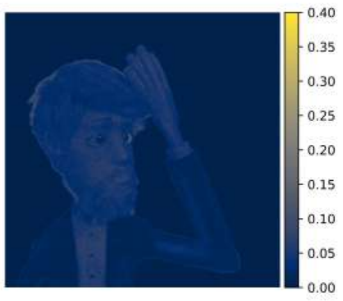


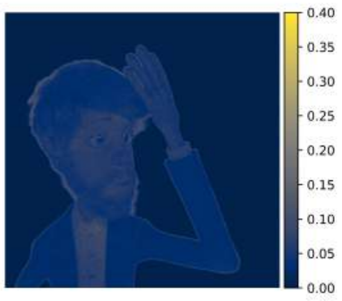
Figure 6.6: This figure shows images from the evaluation set (left) and their predictions from the respective fully trained model (right). While the overall color is slightly off, fine structures are retained. The first row shows a good prediction of the overfitted BSP model. The second row shows the more abstract caterpillar model, where the prediction diverges. The third and fourth row show the corresponding green screen images and their predictions. The visual quality of these predictions is roughly on the same level, while the ones of the BSP model fluctuate between very good and divergent. However, even the GSP model cannot correctly reproduce the caterpillar's colors and only retains its overall structure and base color. The individual images can be found in their original resolution online.



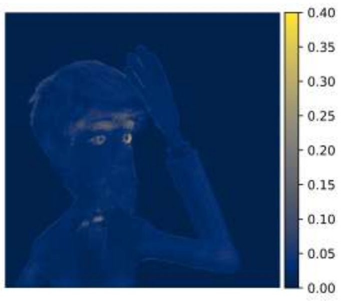
(a) RGB distance between SIM prediction (BSP) and ground truth



(b) RGB distance between fully trained BSP model and ground truth



(c) RGB distance between SIM prediction (GSP) and ground truth



(d) RGB distance between fully trained GSP model and ground truth

Figure 6.7: The above figures show a WMSE comparison between the predictions of fully trained BSP/GSP models and the SIM system output. The images illustrate the overall decreased loss of the model predictions in comparison to the SIM output. In Figure (d) the loss reflects the GSP model's difficulty in predicting an accurate matte in the the forehead area.

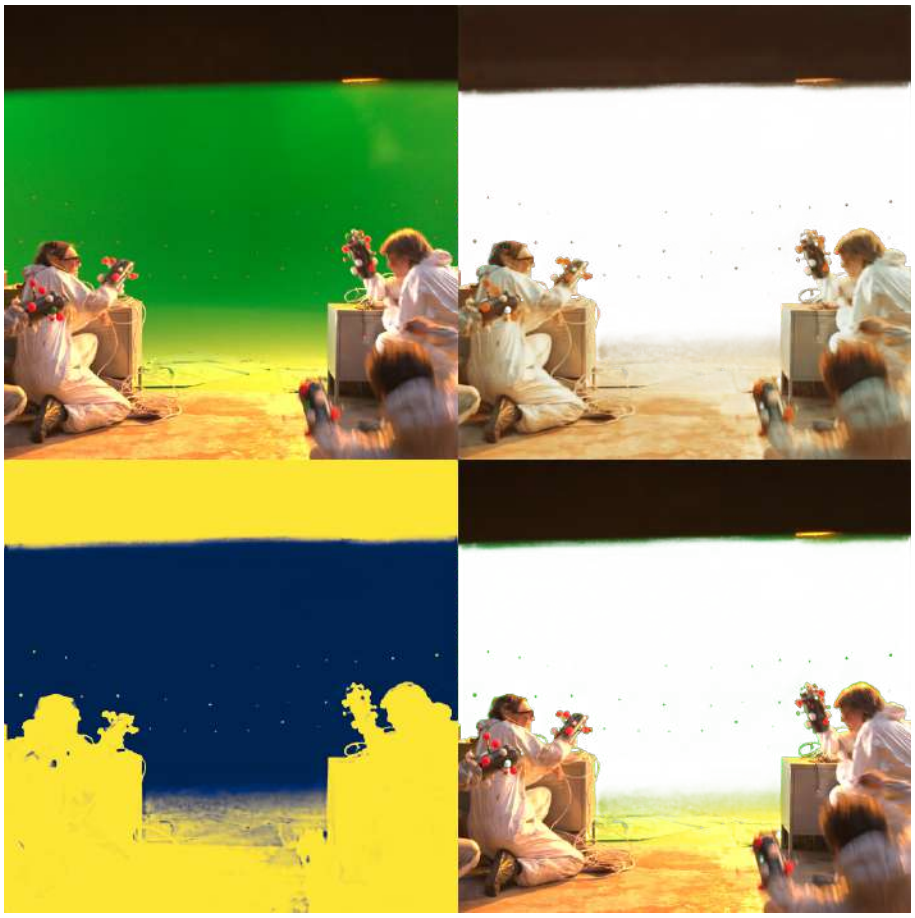


Figure 6.8: This image shows (in clockwise direction) the original green screen image, the prediction, the prediction's matte applied to the original image, and a visualization of the matte. The overall matte quality is very good and this example shows that the system differentiates between mixed and spilled pixels. The matte gradually fades out in the middle of the image, where the floor gradually merges with the background. Almost no green is visible in the original RGB with predicted matte picture, entailing that the network relied here primarily on unmixing via the alpha channel. Another example for mixed pixels can be found on the edge of the dark image section. In contrast to the slow alpha fade in the image's center, the network makes use of both spill suppression (compare the edge in lower right image and upper right one) and unmixing (see alpha map). The right box is an area where only spill suppression has been applied: it has a green tinge in the lower right picture which is not visible in the prediction. The tiny dots on the green screen, which have been segmented by the alpha, are tracker markers. They are used in post-processing to compute the correct orientation of the new background. The original images can be found online. The image originates from the Tears of Steal movie [13].

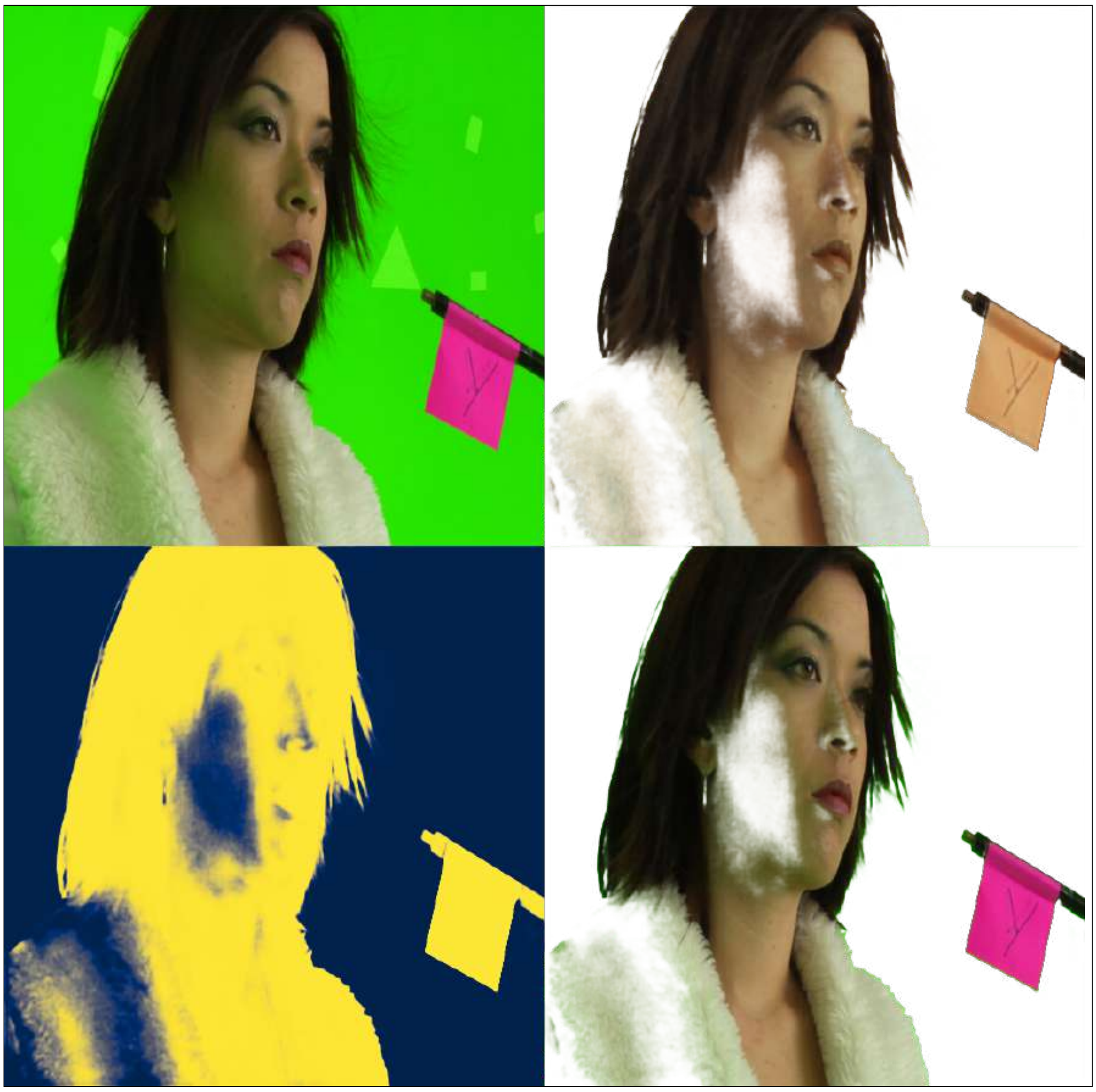


Figure 6.9: This image shows (in clockwise direction) the original green screen image, the prediction, the prediction's matte applied to the original image, and a visualization of the matte. The overall quality of the matte is not as convincing as the one from Figure 6.8. In contrast the color reproduction quality of the face, hair and clothing excels. This can be observed in the corner of the right eye, nose and chin region. This prediction is a good example on how the network relies on unmixing in weakly textured areas that are dyed in the background color (cheek), and on spill suppression in strongly structured regions (hair). Ideally, the hair strands' color would have been corrected through unmixing instead of spill suppression. In conjunction with the caterpillar prediction from Figure 6.6, this image also shows a peculiarity in the network's predictions: The reproduction of pink colors is especially weak. The images can be found online. The image originates from Hollywood Camera Work that provides green screen plates for free [11].

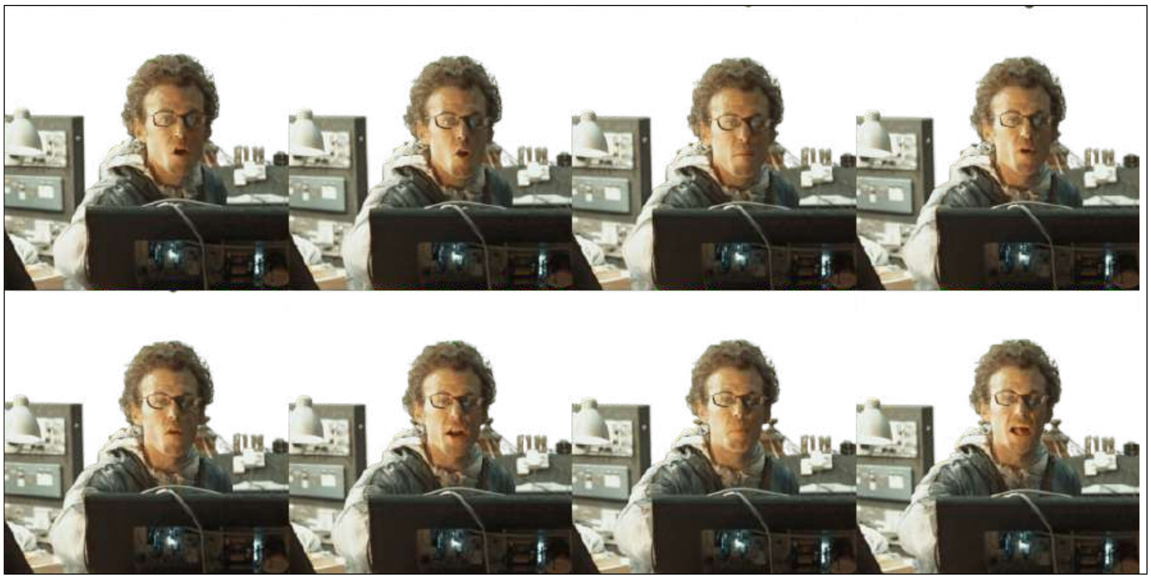
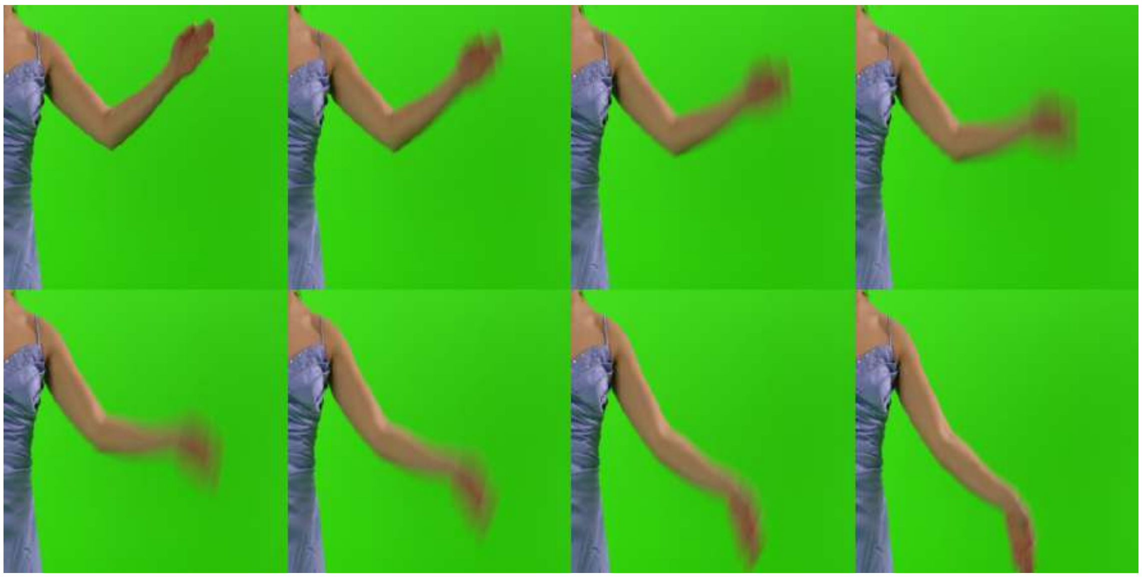
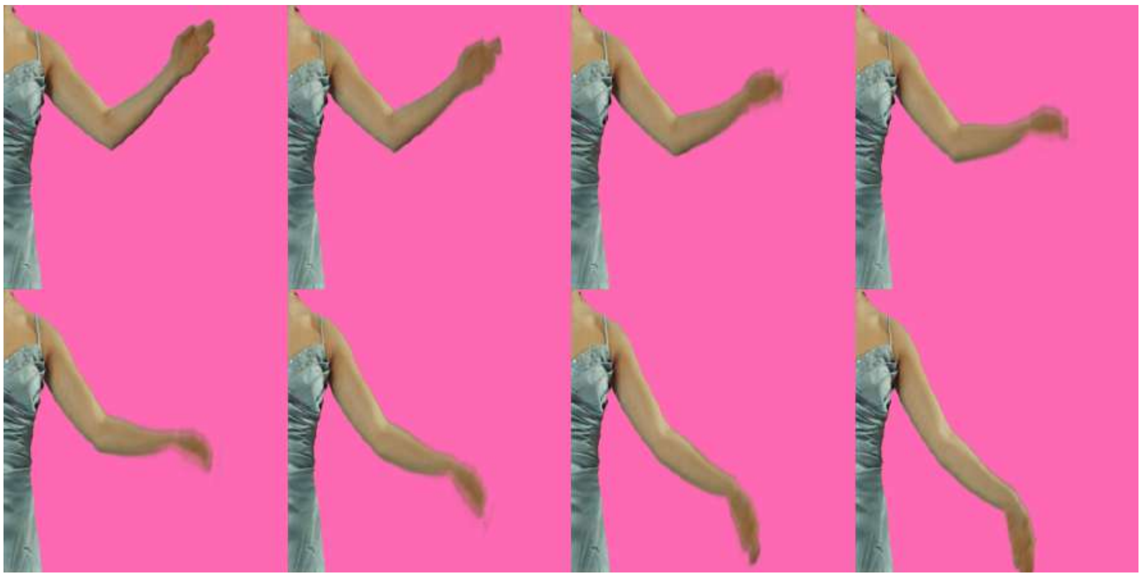


Figure 6.10: To investigate the output's consistency between images, a small video sequence was predicted. This video has only small movements, which are favorable conditions for the system (stable lighting, no substantial input variations). At the same time it makes it easier for the observer to judge the predictions. The system seems to produce consistent results, which aligns with the fact that the training data set has been generated by rendering individual frames of video sequences. The prediction's individual frames and the video can be found online. The image originates from the Tears of Steal movie [13].



(a) original frames



(b) composed prediction

Figure 6.11: Here, the starting frames from a video demonstrating motion blur are shown. (a) shows the original frames, while (b) the corresponding predictions composed onto a new background. The network correctly retained the foreground-background interaction, as established in (b). Although the only blur in the training data set was simulated using the depths of field, it seems enough to correctly resolve motion blur. The corresponding material is available online. The image originates from Hollywood Camera Work that provides green screen plates for free [11].

Figure 6.6 presents predictions of images from the evaluation set. While they correctly reproduce the depicted foreground, the images reveal an overall reduction in the color quality: the shade diverges a little from the original and its reproduction fails in extreme cases. But the images also show that fine structures such as hair strands can be preserved. These results demonstrate that the fully trained BSP model has difficulties in accurately predicting the caterpillar model. But it also shows a very good prediction of the system with the human 3D model. This performance gap, between the more familiar human 3D model and the caterpillar model of unfamiliar form, is due to the earlier mentioned overfitting effect.
In comparison to the BSP prediction of the human 3D model, the overall performance of the GSP model is weaker. But the prediction quality does not depend as much on the familiarity of the used 3D model.

Figure 6.7 shows a comparison between the prediction of the fully trained BSP and GSP model and the respective SIM output. The individual plots show how the, compared to the SIM loss, overall better performance of both models manifests in an overall diminished loss.

The next two figures reveal the fully trained GSP model's performance on natural green screen images. The model predicts the two images with varying quality. In the first image, Figure 6.8, demonstrates that the system differentiates between mixed and spilled pixels. Although this image lacks true spill pixels, the system depends on the matte to correct spill in large, weakly structured areas. It relies on both, spill suppression and unmixing, to predict areas with stronger textures and gradients. This hypothesis is supported by Figure 6.9: Areas with a heavy spill and smooth gradient are faded out and finer structures (hair strands) are corrected using spill suppression and unmixing. The later behavior is welcomed, as it corrects the color while retaining information about foreground background interactions. The former one in contrast is problematic, as the network doesn't seem to take into account the objects structure and apparently just relies on lower-level features such as gradient and hue. Taking the network's receptive field into account, it seems that this may be the root of that behavior. A receptive field size of 113x113 is not large enough for the network to „see " the background. Thus it mistakes large areas with heavy spill for the background. Figure 8.3 from the appendix visualizes the receptive field for representative points. Predictions on natural blue spill scenes have been computed, but are left out due to the systems poor generalization capabilities. Predictions and input scenes are available online.

The last two figures illustrate selected frames from a predicted video sequences. Figure 6.10 displays selected frames from a green screen video. The color fidelity between corresponding pixels is very high and no flickering is visible. Considering that the network has been trained on video frames, this result is anticipated. This is even clearer upon viewing the video online. The second Figure 6.11 shows two images. The first one contains frames from a green screen video that demonstrates motion blur. The second image shows the network's output for it composed onto a new background. Although the training data did not contain explicit motion blur, it seems to be sufficient to simulate blur by adjusting the depth of field accordingly. Showing the composition instead of the plain prediction, demonstrates the importance of being able to retain information about the foreground/background interaction through the alpha matte.

In summary, the visualizations of the synthetic data show the overfitting effect of the fully trained BSP model, as well as the better generalization capabilities of the GSP model. The GSP model is capable of differentiating between mixed and spilled pixels but the matte quality diminishes large spill affected areas with weak structures. Possibly, due to the data set not containing enough examples of large spill affected areas with a weak texture.

7 Conclusion and Future Prospects in Color Spill

This project successfully produced a large data set for spill suppression and trained a deep learning system on it. The fully trained models performance exceeds the minimal requirements established through the statistical analysis of the data. Furthermore, its capabilities extend to natural images, while the model has only been trained on synthetic data.
The statistical analysis revealed a weakness of the custom loss function, namely the imbalance between its alpha and spill components. A solution to this problem would be to use of a weighted sum.
The model trained on blue spill data shows symptoms of overfitting. The large gap between training and evaluation loss are a first indication for this hypothesis. It is further supported by the visual examples provided in Figure 6.6.
On the other hand, the fully trained GSP model has decent generalization capabilities that extend to natural images. The overall color quality is good but not sufficient for the discerning human eye. In most cases the network successfully differentiates between spilled and mixed pixels. For the latter one, it does not exclusively rely on the matte, but may use it in conjunction with spill suppression. Performance issues of the fully trained GSP model arise in large, spill affected areas with weak textures and small gradients.
The solution to this problem is not straightforward. One possibility would be to split the deep learning system into two architectures: one for the matting/unmixing and one for the spill suppression. In this case, the matting network would be trained on RGB data and its output would serve as additional input for the spill suppression network, which now generates unspilled RGB images from RGBA data. The image matting network could be trained using mattes from existing data sets compiled with the ones from this project. Similar to the data acquisition method of Xu et. al [23], existing mattes could be used to construct chroma keying scenes. The advantage of using existing matting data sets would be the usage of natural images which hopefully increase the matte quality of natural images. No trimap would be necessary, maintaining the system's status as being fully automatic. The usage of spilled data is required, to avoid that the network renders heavily spilled pixels transparent. The spill suppression network would then be trained on RGBA data from this project's data set.
Advantages of this structure would be an (hopefully) increased matte quality. The natural image matting algorithm from Xu et. al [23] can discern very fine structures, which should also be the goal for any spill suppression system with matte computation. Another advantage of this partition is the resolution of the imbalance between the alpha and spill suppression component in the presented loss function.

Beyond improving this project's results, future prospects in color spill would be to extend it to natural scenes. Here, the goal would still be to remove the foregrounds lighting interactions with the background. The difficulty is increased manifold as distinct background components contribute to varying degrees to the the spill effect. The acquisition of ground truth data would also be more difficult than the one of this project.

8 Appendix

model parameters	
levels	[2, 5]
channel number of first layer	[5, 20]
channel multiplication factor	[1.25, 2.1]
adam parameters	
lr	(0.00001, 0.1]
beta 1	(0.8, 1]
beta 2	(0.8, 1]
decay	not tuned during model selection

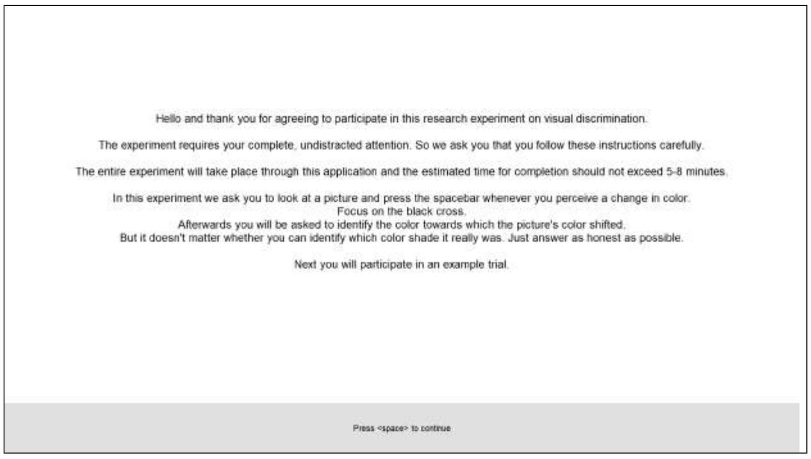
Table 8.1: Complete list of hyper-parameters tuned during model selection.

model parameters	
maximal number of parameters	80,000
data augmentation parameters	
degree of random rotations	15°
shift along x axis	0.2%
shift along y axis	0.2%
shear intensity in degrees	2°
zoom range in %	[.7, 1]
fill mode	reflection
horizontal flip	{Yes, No}
early stopping parameters	
validation split	20%
monitored quantity	WMSE
number of epochs without improvement after which training will be stopped	10
minimum change in monitored quantity to qualify as improvement	0.001

Table 8.2: Complete list of fix hyper-parameters in the model selection process.

3D Model	**License Type**	**License Link**
Vincent	CC Attribution 4.0	https://cloud.blender.org/terms-and-conditions
Frank Sheep	CC-BY 3.0	https://gooseberry.blender.org/about/
Frank Caterpillar	CC-BY 3.0	https://gooseberry.blender.org/about/
Victor	CC-BY 3.0	https://gooseberry.blender.org/about/
Prog	CC-BY 2.5	https://orange.blender.org/blog/creative-commons-license-2/
Sintel	CC-BY 3.0	https://durian.blender.org/sharing/
Rinky	CC-BY 3.0	https://peach.blender.org/about/
Koro	CC-BY 3.0	http://www.caminandes.com/sharing/

Table 8.3: Licensing conditions for all 3D models.


Hello and thank you for agreeing to participate in this research experiment on visual discrimination.

The experiment requires your complete, undistracted attention. So we ask you that you follow these instructions carefully.

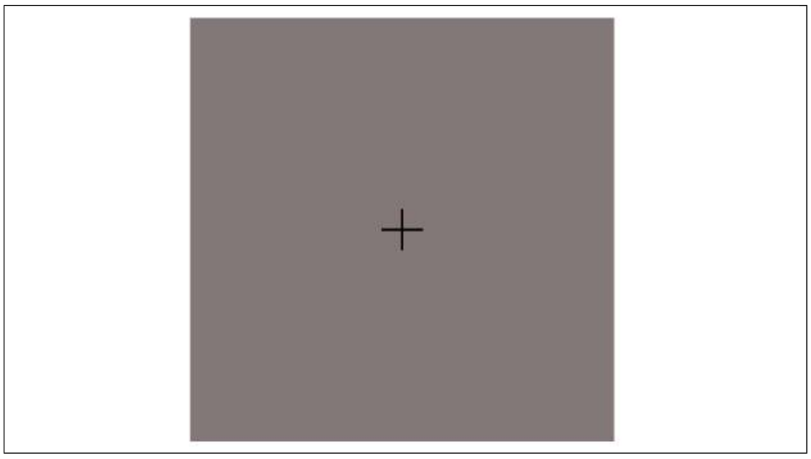
The entire experiment will take place through this application and the estimated time for completion should not exceed 5-8 minutes.

In this experiment we ask you to look at a picture and press the spacebar whenever you perceive a change in color.
Focus on the black cross.
Afterwards you will be asked to identify the color towards which the picture's color shifted.
But it doesn't matter whether you can identify which color shade it really was. Just answer as honest as possible.

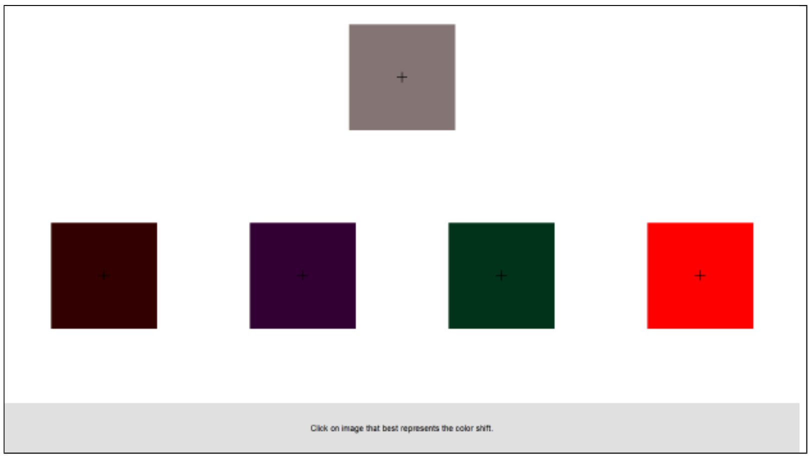
Next you will participate in an example trial.

Press <space> to continue

(a) Instruction screen



(b) Reaction screen


Click on image that best represents the color shift.

(c) Forced choice screen. The middle top square shows the blended image at the time the subject indicated that they perceived an image.

Figure 8.1: The above screen shots show the experiments layout. Not included where the screens indicating the end of the demo and indicating the end of the experiment.

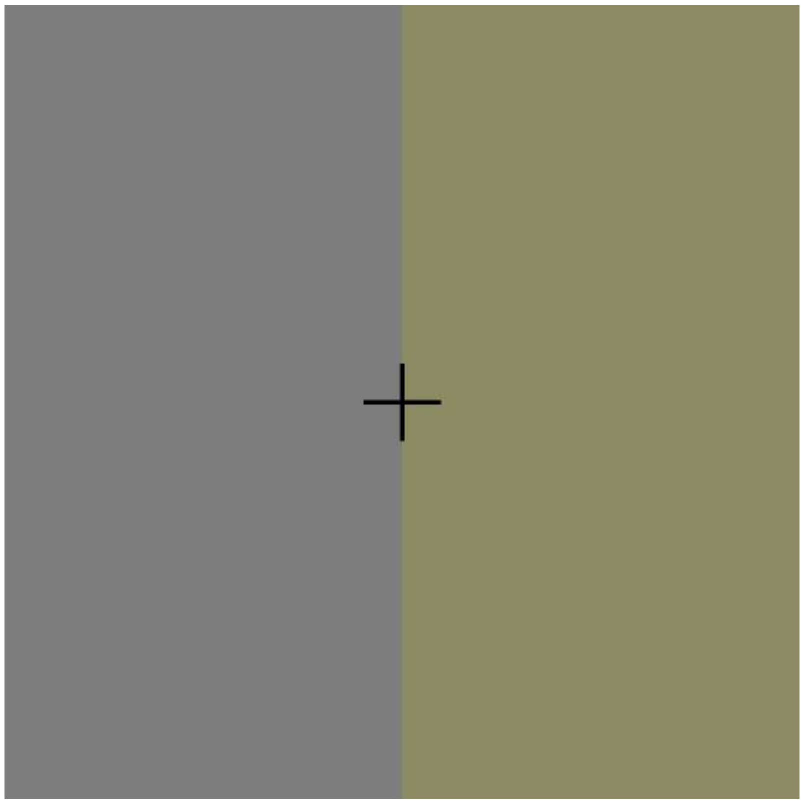


Figure 8.2: Visualization of the contrast between the gray image used in the experiment (left side) and the gray image blended with a yellow image with $\alpha = 0.2$. It is recommended to view the image on a screen.

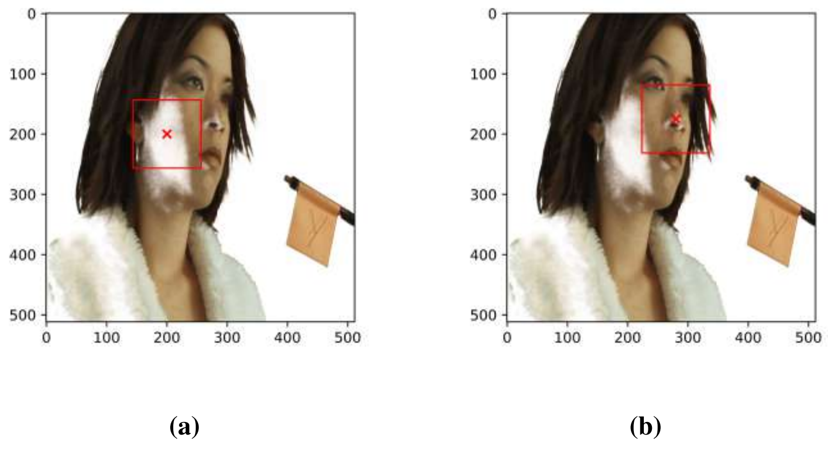


Figure 8.3: Network predictions with selected image locations and corresponding receptive fields. Graphic (a) shows the receptive field for one point in the cheek region. The network receives at that point no input from the image's background region. Graphic (b) depicts a similar situation for the nose region. Here, the background area is included but is scarce. The image originates from Hollywood Camera Work that provides green screen plates for free [11].

Bibliography

[1] Y. Aksoy, T. O. Aydin, M. Pollefeys, A. Smolić. "Interactive High-Quality Green-Screen Keying via Color Unmixing." en. In: *ACM Transactions on Graphics* 35.5 (Aug. 2016), pp. 1–12. ISSN: 07300301. DOI: 10.1145/2907940. URL: http://dl.acm.org/citation.cfm?doid=2965650.2907940 (visited on 01/24/2019) (cit. on p. 25).

[2] *Amazon Mechanical Turk*. URL: https://www.mturk.com/ (visited on 08/19/2019) (cit. on p. 25).

[3] A. Bousseau, S. Paris, F. Durand. "User-Assisted Intrinsic Images." en. In: *ACM Transactions on Graphics (TOG)* 28.5 (2009), p. 130 (cit. on p. 27).

[4] D. J. Butler, J. Wulff, G. B. Stanley, M. J. Black. "A Naturalistic Open Source Movie for Optical Flow Evaluation." en. In: *Computer Vision – ECCV 2012*. Ed. by D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, C. Schmid. Vol. 7577. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 611–625. ISBN: 978-3-642-33782-6 978-3-642-33783-3. DOI: 10.1007/978-3-642-33783-3_44. URL: http://link.springer.com/10.1007/978-3-642-33783-3_44 (visited on 03/01/2019) (cit. on p. 29).

[5] R. Carroll, R. Ramamoorthi, M. Agrawala. "Illumination Decomposition for Material Recoloring with Consistent Interreflections." en. In: *ACM Transactions on Graphics (TOG)* 30.4 (2011), p. 43 (cit. on p. 27).

[6] L. Colin. *Sintel*. 2010. URL: https://durian.blender.org (cit. on p. 29).

[7] D. Cunningham, C. Wallraven. *Experimental Design: From User Studies to Psychophysics*. Vol. 2011. Nov. 2011. ISBN: 978-1-56881-468-1 (cit. on p. 39).

[8] X. Glorot, A. Bordes, Y. Bengio. "Deep Sparse Rectifier Neural Networks." en. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (June 2011), pp. 315–323 (cit. on pp. 15, 16).

[9] I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org/ (visited on 01/20/2018) (cit. on pp. 13, 14, 19, 20, 38).

[10] C. M. Goral, K. E. Torrance, D. P. Greenberg, B. Battaile. "Modeling the Interaction of Light Between Diffuse Surfaces." In: *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '84. New York, NY, USA: ACM, 1984, pp. 213–222. ISBN: 978-0-89791-138-2. DOI: 10.1145/800031.808601. URL: http://doi.acm.org/10.1145/800031.808601 (visited on 08/31/2019) (cit. on pp. 10, 28).

[11] *Green Screen Plates*. URL: https://www.hollywoodcamerawork.com/green-screen-plates.html (visited on 10/11/2019) (cit. on pp. 52, 54, 61).

[12] G. Hinton. *Neural Networks for Machine Learning - Overview of mini-batch gradient descent*. Coursera. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (visited on 09/21/2019) (cit. on p. 19).

[13] I. Hubert. *Tears of Steel*. En. 2012. URL: https://mango.blender.org/ (visited on 10/11/2019) (cit. on pp. 8, 51, 53).

[14] D. P. Kingma, J. Ba. "Adam: A Method for Stochastic Optimization." In: *arXiv:1412.6980 [cs]* (Dec. 2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980 (visited on 08/23/2019) (cit. on p. 20).

[15] T. M. Mitchell. *Machine Learning*. en. McGraw-Hill series in computer science. New York: McGraw-Hill, 1997. ISBN: 978-0-07-042807-2 (cit. on pp. 13, 14).

[16] *Optical blur and the circle of "non-sharpness"*. en. Oct. 2017. URL: https://craftofcoding.wordpress.com/2017/10/23/optical-blur-and-the-circle-of-non-sharpness/ (visited on 10/12/2019) (cit. on p. 9).

[17] L. Prechelt. "Early Stopping | but when?" en. In: (1988), p. 15 (cit. on p. 20).

[18] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, P. Rott. "A Perceptually Motivated Online Benchmark for Image Matting." en. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2009), p. 8 (cit. on pp. 25, 30).

[19] O. Ronneberger, P. Fischer, T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." en. In: *arXiv:1505.04597 [cs]* (May 2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597 (visited on 07/23/2018) (cit. on p. 21).

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge." en. In: *International Journal of Computer Vision* 115.3 (Dec. 2015), pp. 211–252. ISSN: 1573-1405. DOI: 10.1007/s11263-015-0816-y. URL: https://doi.org/10.1007/s11263-015-0816-y (visited on 07/22/2019) (cit. on p. 29).

[21] E. Shahrian, B. Price, S. Cohen, D. Rajan. "Temporally coherent and spatially accurate video matting." en. In: *Computer Graphics Forum* 33.2 (2014), pp. 381–390. ISSN: 1467-8659. DOI: 10.1111/cgf.12297. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12297 (visited on 02/18/2019) (cit. on p. 31).

[22] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, S. Birchfield. "Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization." en. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Salt Lake City, UT: IEEE, June 2018, pp. 1082–10828. ISBN: 978-1-5386-6100-0. DOI: 10.1109/CVPRW.2018.00143. URL: https://ieeexplore.ieee.org/document/8575297/ (visited on 04/30/2019) (cit. on pp. 11, 29).

[23] N. Xu, B. Price, S. Cohen, T. Huang. "Deep Image Matting." In: *arXiv:1703.03872 [cs]* (Mar. 2017). arXiv: 1703.03872. URL: http://arxiv.org/abs/1703.03872 (visited on 01/16/2019) (cit. on pp. 11, 23, 24, 30, 56).

Universität Bremen

Nachname	Maul	Matrikelnr.	3024878
Vorname/n	Pascale		

A) Eigenständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.
Alle Teile meiner Arbeit, die wortwörtlich oder dem Sinn nach anderen Werken entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht. Gleiches gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.
Die Arbeit wurde in gleicher oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht.
Die elektronische Fassung der Arbeit stimmt mit der gedruckten Version überein.
Mir ist bewusst, dass wahrheitswidrige Angaben als Täuschung behandelt werden.

B) Erklärung zur Veröffentlichung von Bachelor- oder Masterarbeiten

Die Abschlussarbeit wird zwei Jahre nach Studienabschluss dem Archiv der Universität Bremen zur dauerhaften Archivierung angeboten. Archiviert werden:

1) Masterarbeiten mit lokalem oder regionalem Bezug sowie pro Studienfach und Studienjahr 10 % aller Abschlussarbeiten
2) Bachelorarbeiten des jeweils ersten und letzten Bachelorabschlusses pro Studienfach u. Jahr.

[✓] Ich bin damit einverstanden, dass meine Abschlussarbeit im Universitätsarchiv für wissenschaftliche Zwecke von Dritten eingesehen werden darf.

[✓] Ich bin damit einverstanden, dass meine Abschlussarbeit nach 30 Jahren (gem. §7 Abs. 2 BremArchivG) im Universitätsarchiv für wissenschaftliche Zwecke von Dritten eingesehen werden darf.

[ ] Ich bin nicht damit einverstanden, dass meine Abschlussarbeit im Universitätsarchiv für wissenschaftliche Zwecke von Dritten eingesehen werden darf.

C) Einverständniserklärung über die Bereitstellung und Nutzung der Bachelorarbeit / Masterarbeit / Hausarbeit in elektronischer Form zur Überprüfung durch Plagiatssoftware

Eingereichte Arbeiten können mit der Software *Plagscan* auf einen hauseigenen Server auf Übereinstimmung mit externen Quellen und der institutionseigenen Datenbank untersucht werden.
Zum Zweck des Abgleichs mit zukünftig zu überprüfenden Studien- und Prüfungsarbeiten kann die Arbeit dauerhaft in der institutionseigenen Datenbank der Universität Bremen gespeichert werden.

[✓] Ich bin damit einverstanden, dass die von mir vorgelegte und verfasste Arbeit zum Zweck der Überprüfung auf Plagiate auf den *Plagscan*-Server der Universität Bremen hochgeladen wird.

[✓] Ich bin ebenfalls damit einverstanden, dass die von mir vorgelegte und verfasste Arbeit zum o.g. Zweck auf dem *Plagscan*-Server der Universität Bremen hochgeladen u. dauerhaft auf dem *Plagscan*-Server gespeichert wird.

[ ] Ich bin nicht damit einverstanden, dass die von mir vorgelegte u. verfasste Arbeit zum o.g. Zweck auf dem *Plagscan*-Server der Universität Bremen hochgeladen u. dauerhaft gespeichert wird.

Mit meiner Unterschrift versichere ich, dass ich die oben stehenden Erklärungen gelesen und verstanden habe.
Mit meiner Unterschrift bestätige ich die Richtigkeit der oben gemachten Angaben.

Datum Unterschrift