JPMTR-2403
DOI 10.14622/JPMTR-2403
UDC 535.6:004.9

Original scientific paper | 190
Received: 2023-01-19
Accepted: 2024-06-12

A neural network to predict the spectral reflectance of prints

Sven Ritzmann, Peter Urban

University of Wuppertal,
School of Electrical, Information and Media Engineering,
42119 Wuppertal, Germany

ritzmann@uni-wuppertal.de
purban@uni-wuppertal.de

Abstract

The reconstruction of spectral reflectance from RGB triplets created by digital cameras is a topic of great interest. Different approaches dealing with this topic have been published. In the recent years, most approaches utilize neural networks. These approaches mainly differ in the chosen network architecture, the way of obtaining the dataset and the used hardware. While the most approaches aim for generalized applicability on a wide range of spectra, this paper aims for applicability on a limited set of spectra given by a typical use case of the printing industry. In this paper a neural network was trained to predict the spectral reflectance of prints. Therefore, 10 800 color patches were printed, measured by a spectrophotometer and captured by an RGB camera under different light sources generated with a DLP projector. The performance of the trained network was tested by determining the CIEDE2000 color difference as well as the mean squared error between the predicted and the measured spectral reflectance. The dataset was systematically reduced to examine how the number of color patches and light sources used for training influences the performance of a network. This paper shows that a network performed best when confronted with prints printed on the same substrate using the same color management settings as the dataset used for training. Training a network with multiple datasets on different substrates increased the generalization of a network, but decreased the performance compared to a network trained with a single combination of substrate and color management settings. Reducing the number of color patches as well as reducing the number of light sources influenced the performance of a network negatively, but still leads to decent results.

Keywords: spectral reflectance reconstruction, neural networks, RGB to multispectral, light source, color patches

1. Introduction and context

This article is part of a project that aims to digitize cultural heritage items. The digitization system used implements the 3D scanning method of structured light. During a structured light scan, a projector projects a series of light patterns onto the scene while a camera captures these patterns. This series of patterns results in a unique code for each pixel of the projector. As the camera captures the patterns, each of its pixels corresponds to a pixel of the projector. A 3D point of the scene is triangulated based on these correspondents and the known position and orientation of the camera and the projector (Drouin and Beraldin, 2020). To obtain a colorized 3D scan each 3D point is assigned to an RGB triplet captured by the used camera.

The main idea of the mentioned project is to use a neural network and specific hardware (a camera and a projector) to reconstruct the spectral reflectance of a scene point based on RGB triplets captured under different light sources. As a colorized 3D scan processes the same RGB triplets, the reconstruction of the spectral reflectance leads to a multispectral 3D scan. This article represents the first step to realize this concept. As a 3D scene adds complexity to the problem, e.g. interreflections, this research reduces the problem to a 2D space and uses a typical scenario from the printing industry. It centers on the reconstruction of the spectral reflectance of color patches printed by a UV-curing digital printer. The neuronal network will be able to predict the spectral reflectance for each pixel of an RGB image of a print. This opens new ways to proof the accuracy of a print or a specific part of a print, e.g. during production.

The idea to use neural networks to reconstruct the spectral reflectance of a scene is not new. In the past years, three competitions on spectral reconstruc-

tion from RGB images have been conducted (Arad, et al., 2018; 2020; 2022). The datasets used for training consist of hyper spectral images showing a diverse set of scenes captured with a mobile hyperspectral camera in several (not specified) environments. A simulated camera pipeline transforms the multispectral images to RGB images. The pipeline implements the spectral response of a real camera, automated exposure determination, noise, image signal processing and image compression for the JPEG format. Arad, et al. (2018; 2020; 2022) briefly describe the best performing approaches for each year. Zhang, et al. (2022) offer a detailed insight and categorization of several approaches based on the same or similar datasets. Simplified, due to the used datasets, most of the approaches presented are based on convolutional neural networks (CNN) or generative adversarial networks (GAN), that process the image as a whole. Therefore, the resulting networks perform well reconstructing the spectral reflectance of a whole scene, including interreflections. Due to interreflections and the diverse dataset, a network trained based on printed color patches with no interreflections should perform better in the specific scenario of this research.

Lazar, Javoršek and Hladnik (2020) follow a contrary approach. They use 1 031 Munsell matte patches as the foundation of their dataset. Each is measured by a spectrophotometer at five different points. The average of the five measurements equals the spectral reflectance of the patch. To obtain the corresponding RGB triplets a digital camera captures each patch under constant lighting conditions (3 019 K). The median RGB triplet of the inner 50 % of the pixels represent the RGB triplet of the corresponding patch. A densely connected feed forward network with one hidden layer is trained using the dataset. As such networks tend to find a local minimum of the cost function instead of the global minimum, each training is repeated 41 times. Lazar, Javoršek, and Hladnik (2020) suggest that the performance of the best network (with 96.2 % of the predicted reflectance resulting in a $\Delta E_{00} < 3.0$) could be enhanced by increasing the number of hidden layers and the number of RGB triplets per patch, using multiple cameras with different spectral responses. Lazar and Hladnik (2023) proof this assumption to be correct.

The described approach is close to the approach of this research. However, the first difference is the use of printed color patches instead of Munsell patches. While Munsell patches are predefined, printing gives freedom to define a dataset regarding the used halftone values and the number of patches. The disadvantage is, that the range of spectra is limited by the gamut of the used printer. The second difference is the use of multiple light sources instead of multiple cameras to enhance the number of RGB triplets. Both approaches follow the same principal. In this article the light source is changed due to the given hardware. Furthermore, using multiple cameras will lead to registering issues when predicting the spectral reflectance for a whole image.

2. Methods

2.1 Mathematical background and main research workflow

A camera captures the light reflected from a point in a scene and converts it into a triplet of red, green, and blue color values. The resulting triplets depend on the spectral distribution of the light source, the spectral reflectance of the material of the scene point, and the characteristics of the RGB filters used in the camera (Park, et al., 2007) (Equation 1).

$$\mathrm{px}_K = \int_{380nm}^{730nm} \mathrm{LS}(\lambda)\beta(\lambda)C_K(\lambda)\mathrm{d}\lambda \qquad [1]$$

where $\mathrm{LS}(\lambda)$ is the spectral power distribution of the light source, px_K is the k-th color channel of a pixel, $\beta(\lambda)$ is the spectral reflectance of a scene point, and C_K is the spectral response of the camera for k-th color channel.

Assuming the spectral distribution of the light source $\mathrm{LS}(\lambda)$ and the spectral response of the camera C are measured in discrete steps and the unknown spectral reflectance $\beta(\lambda)$ is discrete as well, the integral in Equation 1 reduces to a summation as shown in Equation 2. The use of multiple light sources leads to a linear equation system as shown in Equation 3. Equation 4 shows Equation 3 in matrix notation. To compute the spectral reflectance, the inverse of the matrix LSC is multiplied to both sides of Equation 4 leading to Equation 5. Solving Equation 5 leads to the unknown spectral reflectance.

In general, the use of multiple light sources ensures that the LSC is a square matrix and therefore invertible. In the optimal case, a monochromatic grayscale camera K = [gray] captures the patches, and the number of light sources n equals the number of measurement points in the discrete spectra *m*. The discrete spectral distribution of each light source is 1 at one measurement point (or wavelength) and 0 at the others and has no overlap with the other light sources. Consequently, the spectral distribution of $\mathrm{LS}_{\mathrm{j}=1}$ is [1, 0, …, 0] and the spectral distribution of $\mathrm{LS}_{\mathrm{j}=\mathrm{m}}$ is [0, 0, …, 1]. This means that LSC is a diagonal matrix and easy to invert. Using a camera with multiple color channels (e.g. K = [R, G, B)] would reduce the number of necessary light sources n according to Equation 6.

$$\mathrm{px}_{jK} = \sum_{i=1}^{m} \mathrm{LS}_{ij}\, \beta_i\, C_{Ki} \quad [2]$$

$$\begin{bmatrix} \mathrm{px}_{1K} \\ \vdots \\ \mathrm{px}_{nK} \end{bmatrix} = \begin{bmatrix} LS_{11}\, C_{K1} & \cdots & LS_{m1}\, C_{Km} \\ \vdots & \ddots & \vdots \\ LS_{1n}\, C_{K1} & \cdots & LS_{mn}\, C_{Km} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_m \end{bmatrix} \quad [3]$$

$$\overrightarrow{\mathrm{px}_K} = \mathrm{LSC}_K\, \vec{\beta} \quad [4]$$

$$\vec{\beta} = \mathrm{LSC}_K^{-1}\, \overrightarrow{\mathrm{px}_K} \quad [5]$$

$$n = \frac{m}{n_{\mathrm{channels}}} \quad [6]$$

Where i is the i-th point in the discrete spectrum [1 ; m], j is the j-th light source [1 ; n], K is the color channel [R,G,B] or monochromatic [gray], n_{channels} represents the number of color channels, β is the discrete spectral reflectance of the patch, C is the discrete spectral response of the camera, LS is the discrete spectral distribution of the light source, and px is a pixel vector.

In practice, using these equations has disadvantages. All variables, except the spectral reflectance, has to be known. While measuring the spectral distribution of each light source would be relatively easy, measuring the spectral response of a camera would be more complex. Furthermore, Equation 1 neglects the camera pipeline processing of the captured data to an RGB image. For most consumer cameras these pipelines are black boxes, but have to be known to reproduce the spectral reflectance.

A neural network can be trained to approximate these equations including the spectral distributions of the light sources and the spectral response of the camera. Furthermore, a neural network can approximate those equations with less light sources than determined by Equation 6. A set of RGB triplets captured under different light sources and the corresponding spectral reflectance would be sufficient. Therefore, obtaining these datasets is the main task alongside training and evaluating the neural network. Figure 1 shows the main research workflow and the performed processes.

Each process is assigned to one of the main fields namely the datasets, the capturing device and the neural network. The datasets field deals with building the sets of RGB triplets and corresponding spectral reflectance used to train, validate and test the neural network. The first processes are to define the halftone values of the color patches (Section 2.4.1) and to print these patches on charts using different parameters (Section 2.4.2). The results are four printed datasets. For each printed dataset each color patch is measured by a spectrophotometer to obtain its spectral reflectance (Section 2.4.2). The obtained data is reorganized in an array for further processing (Section 2.4.3). Parallel, the color charts are placed into the capturing device (Section 2.4.4) for being captured (Section

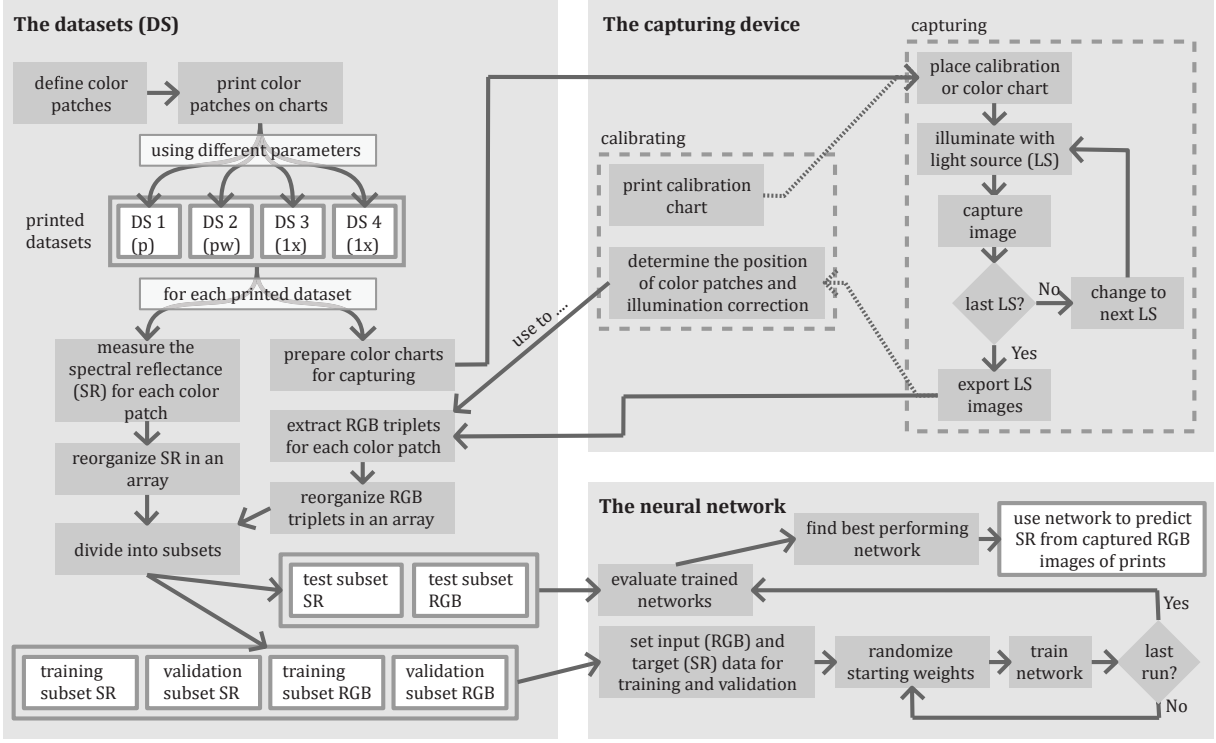

Figure 1: The main research workflow

2.4.5) The capturing device field has two main processes that interacts with each other. The calibration process starts with printing a calibration chart (Section 2.4.6). The calibration chart is given to the capturing process. During capturing the calibration chart is illuminated by a light source emitted by a projector and an RGB camera captures an image of the illuminated chart. As long as the current light source is not the last one it is changed to the next one. At the end, an RGB image of the calibration chart is captured for each light source. The capturing process, returns these images to the calibration process which determines the position and dimension of each patch in the image (Section 2.4.6) and computes a factor for each patch to correct uneven illumination (Section 2.4.8). The calibration result is stored for later use.

The prepared color charts run through the same capturing process as the calibration chart. The resulting images are returned to the extract RGB triplets process in the dataset field. This process extracts an RGB triplet for each color patch and light source (Section 2.4.7) using the information gathered during calibration. The results are reorganized into an array following the same order as the corresponding spectral reflectance array.

The two arrays represent the RGB triplets and the corresponding spectral reflectance of one dataset. In the next process each dataset is divided into a training, validation and test subset (Section 2.5). The training and validation subsets are set as input (RGB) and target (spectral reflectance) data for the first process concerning the neural network. The starting weights of the network are randomized followed by the actual training of the network. As the starting weights highly influence the training outcome, the randomization and training are repeated several times (Section 2.6). When the last training run is finished, the test subset is used to evaluate the trained networks (Section 2.7). Based on the results, the best performing network is chosen. This network can be used to predict the spectral reflectance for each pixel of an RGB image of a sample printed and captured with the same parameters as the dataset used for training.

By varying parameters of this workflow, this article examines the performance of a neural network depending on the use of different datasets (Section 3.1), regarding its generalization capabilities (Section 3.2), depending on the size of the dataset (Section 3.3) and the number and combination of the light sources used (Section 3.4).

2.2 Software and frameworks

Adobe InDesign (Adobe Inc., 2023) version 18.1 was used to create the printing files for the datasets. The raster image processor software VersaWorks (Roland Digital Group Corporation, 2023) version 6.18.1.1 was used to prepare the datasets for printing especially in terms of color management. The framework gPhoto2 version 2.5.28 (gPhoto, 2022) was used on a Raspberry PI 4b to programmatically trigger the camera to capture the color charts and to control camera parameters like ISO, shutter speed and aperture. OpenCV for Python version 4.51.48 (Bradski, 2000) was used to open, save and convert images. It was also used to detect the position and dimension of color patches in images. The framework will be referred as cv2 in the following. Numpy version 1.21.4 (Harris, et al., 2020) was used to prepare, store, and process the created data. The framework will be referred as np in the following. Keras version 2.10 as part of TensorFlow version 2.10.1 (TensorFlow, 2023) was used to build, train and evaluate the neural networks in this paper. The framework colour-science version 0.4.1 (Mansencal, et al., 2022) was used to transform spectra to CIEXYZ (ISO, 2020b), RGB and CIELAB (ISO, 2020c). Each color transformation in this paper used the color matching functions of the CIE 1931 2° Standard Observer (ISO, 2020a) and D50 (ISO, 2022a) as illuminant. In addition, transformations from spectra to RGB use Aces2065 as color space. Furthermore, colour-science was used to visualize the CIE 1976 UCS (ISO, 2023) chromaticity diagrams and to compute the CIE ΔE_{00} (ISO, 2022b) color difference between predicted and measured spectral reflectance.

2.3 Hardware

The flatbed printer RolandDG VersaUV LEF-300 (firmware 2.80) was used to print the datasets. The printer was equipped with ECO-UV-4 UV-curing inks. Besides cyan, magenta, yellow and black the printer is able to print white. It will be referred as LEF300 printer in the following. A Konica Minolta Auto Scan Spectrophotometer FD9 was used to obtain the spectral reflectance of the printed color charts. It will be referred as FD9 spectrophotometer in the following. A Canon EOS M50 MK2 (firmware 1.0.3) digital camera equipped with a Canon EF-M 28MM F/3.5 MACRO IS STM lens and a polarizing filter was used to capture the color charts. It will be referred as M50 camera in the following. During capturing the Texas Instruments DLP LightCrafter Display 230NP EVM projector served as adjustable light source. Its relevant components are an Osram KR CSLNM1.23_EN red LED, an Osram KP CSLNM1.F1_EN green LED and an Osram KB CSLNM1.14_EN blue LED, that define the spectral distribution of the emitted light. A polarizing filter was installed in front of its lens. The projector will be referred as DLP projector in the following. A RaspberryPI 4b was used to programmatically control the M50 camera and the DLP projector.

2.4 Obtaining the datasets

2.4.1 Defining the color patches

A total of 10 800 color patches were defined, whereby 400 were combined into one color chart. To reach a sufficient variance of colors chart 27 represents primary colors (cyan, magenta, yellow) as well as black with opacities from 1 % to 100 % ascending 1 % per step. Charts 1 to 3 represent secondary colors (red, green, and blue) where opacity of one primary color ascends in 5 % steps from 5 % to 100 % per row while opacity of the other primary color ascends per column. Charts 24 to 26 each combine one primary color with black following the same principle. The remaining charts (4 to 23) represent tertiary colors. Therefore, opacity of magenta ascends per row and opacity of cyan ascends per column while opacity of yellow ascends per chart in 5 % steps from 5 % to 100 % (Figure 2). The background of each chart was set to 4C black (cyan, magenta, yellow, black) with an opacity of 100 %. The export function in InDesign creates the printable pdf. During creation any color transformation processes were deactivated and no color profile was embedded.

2.4.2 Printing the color patches and obtaining their spectral reflectance

The datasets were printed by the LEF300 printer on Clairfontaine DCP 1849C paper. To obtain four different datasets each chart was printed four times with different parameters. In the RIP software VersaWorks LEF300EcoUV4_Generic_v720x720.icc delivered by Roland DG was set as device color profile for all datasets. The CMYK simulation target profile and the matching method was set to PSOCoated_v3.icc and colorimetric for the first three datasets. The first dataset was directly printed on the substrate. From here on this dataset will be referred as 'p' (for paper). As the substrate includes optical brighteners the second dataset aims to eliminate their influences. Therefore, one layer of 100% 4c black (cyan, magenta, yellow and black) is printed to block these influences. Three layers of white were printed on top followed by the actual color charts. This dataset will be referred as '1x' (1 = one layer of color, x = blocked substrate). Printing the third dataset followed the same procedure except that the color charts were printed with 200% instead of 100% opacity (setting) by setting the overprint parameter in VersaWorks to 2. This dataset will be referred as '2x'. The fourth dataset was also directly printed on DCP 1849C paper but with the CMYK simulation target profile set to RolandWideGamut2_CMYK.icc. This dataset will be referred as 'pw' (for paper wide gamut). Figure 3 shows the mentioned profiles in comparison. While the LEF300 profile represents the printable gamut of the printer, the simulation target profile PSOcoated_v3 limits its gamut. RolandWideGamut2_CMYK includes the complete gamut of the LEF300-printer and, therefore, does not limit the printable gamut.

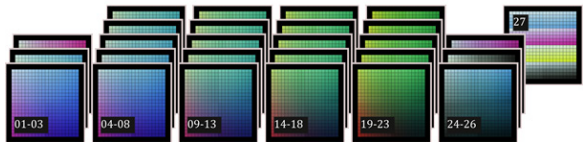

Figure 2: The 10 800 color patches on 27 charts

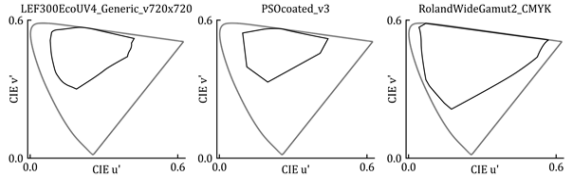

Figure 3: CIE 1976 UCS chromaticity diagrams of the ICC profiles used for printing

The spectral reflectance of each patch was measured by the FD9 spectrophotometer using the measurement modes M0, M1 and M2. The spectral reflectance was measured from 380 nm to 730 nm in 10 nm steps. Figure 4 visualizes the datasets in the CIE 1976 UCS chromaticity diagram. For an analytical comparison Figure 5 shows the ΔE_{00} differences between every pair of datasets.

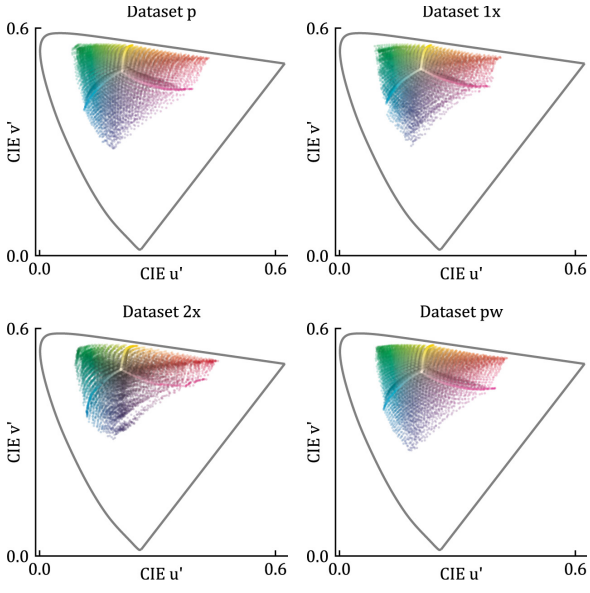

Figure 4: CIE 1976 UCS chromaticity diagrams of the datasets

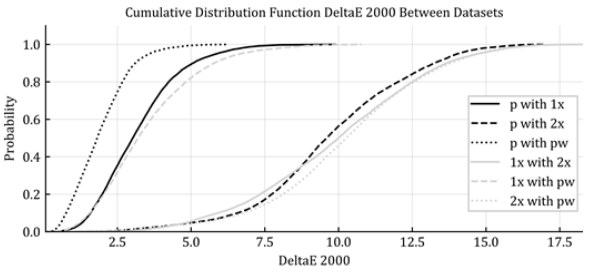

Figure 5: Probability that the color difference between corresponding color patches of two different datasets is less than a certain ΔE_{00} value

The color patches of 'p' and 'pw' are the closest to each other. Accordingly, the gamut of the LEF300 is the closest to the one of the PCOcoated_v3 profile. The difference between 'p' and '1x' is explainable as '1x' is printed on top of one layer of black and three layers of white and therefore on another substrate. The dataset '2x' has the most difference from the other as each color patch is printed with 200% opacity. Therefore, 2x is more saturated, which widens the gamut especially in red, but also limits the gamut in blue.

2.4.3 Organizing the spectral reflectance in arrays

After measuring, the spectral reflectance values of the color patches of a dataset were stored in several csv files. Before this data can be used for training they will be reorganized in np.arrays. Therefore, for each chart the corresponding csv files were opened by calling np.genfromtxt() with the delimeter set to ',' and unpack set to false. As the csv files contains more information than needed, the resulting array was reduced to the spectral reflectance. This information was stored into one array for each chart. Each array has a shape of 400 by 36. Each row represents one color patch while each column represents its spectral reflectance at the corresponding wavelength.

To gather the spectral reflectance of a complete dataset the spectral reflectance arrays of the corresponding charts were combined to a single array. The combined spectral reflectance array has a shape of 10 800 by 36. The patches order in this array is ascending by chart and per chart ascending by row and then ascending by column. So, the first row in the array represents the patch at the first row and first column of the first chart while row 431 in the array represents the patch at the second row and eleventh column of the second chart.

2.4.4 The capturing device

As mentioned in the introduction, the capturing device is a prototype of a 3D scanning system. The housing of the system has the dimensions of 520 x 300 x 300 mm. It is enclosed to lock out extraneous light.

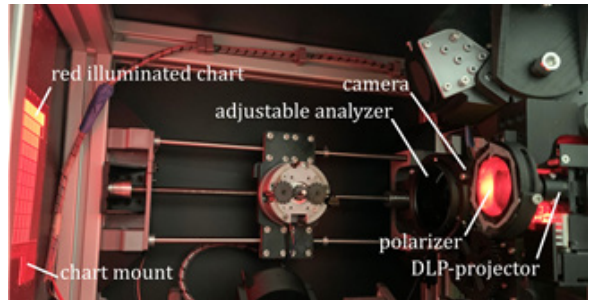

Figure 6: The capturing device with a mounted and red illuminated chart on the left and the DLP projector and M50 camera on the right

On the inside the walls are matte black to reduce internal light scattering. The M50 camera and the DLP projector have a fixed position next to each other on the right side of the box, facing in the direction of the wall on the left site. In the center of this wall is a mount to place a color chart (Figure 6).

Ideally the DLP projector would illuminate the chart at an angle of 45 degree to the camera to avoid reflections. Unfortunately, the angle is limited by the 3D scanning method and the space. To compensate this issue, a polarizing filter in front of the projector serves as polarizer while a polarizing filter in front of the camera serves as analyzer. As polarizer and analyzer were aligned perpendicular to each other, reflections were minimized during capturing (Figure 7).

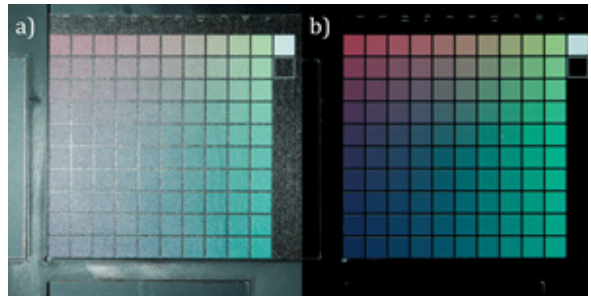

Figure 7: A chart illuminated by white light captured with a) the analyzer aligned parallel to the polarizer and b) the analyzer aligned perpendicular to the polarizer

2.4.5 Capturing the charts

To capture a chart, it has to be placed into the mount. As the substrate printed on is relatively flexible, the chart had to be stiffened. Therefore, each color chart was applied onto white cardboard. Furthermore, each chart had to be divided into four parts (chart parts) to completely fit into the illuminated area (Figure 8).

When the first chart part was placed into the mount, the DLP projector illuminated it subsequently with-

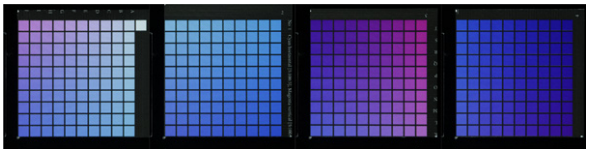

Figure 8: The images of the four chart parts of chart 1 (dataset 'pw') illuminated by white light

white, red, green, blue, cyan, magenta and yellow lights, which can be seen as seven different light sources. With each light source the M50 camera captured an image of the illuminated chart part. When finished, part two to part four of the same chart were captured by following the same procedure. So, each chart part is present in seven images corresponding to the different light sources (Figure 9).

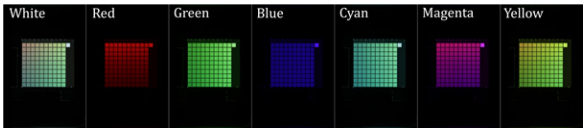

Figure 9: Example of a series of images showing color chart 12 part 1 (dataset 'pw')

This procedure was repeated for each color chart part of each dataset. The resulting images were stored for further processing. Regarding the settings the M50 camera is set to manual control to avoid changing settings due to automated processes. The white balance was set to 5 000 K. The file format for the images was JPEG and the embedded color space was Adobe RGB 1998. Aperture was set to 14, while shutter speed was set to 1 second and ISO was set to 100. Regarding the DLP projector the brightness was set to maximum, while the LEDs were directly controlled using the API provided by Texas Instruments.

2.4.6 Determining the positions and dimensions of the patches in an image of a chart part

To complete the datasets RGB triplets for each color patch had to be extracted from the captured images. Therefore, the positions and dimensions of the colored patches in the images had to be determined. To easily determine position and dimension of each patch despite their changing colors and contrasts to the background, a chart part consisting of only white colored patches was used. The white patches have a high contrast to the black background which eases contour detection. In the following this chart part will be referred as calibration chart. As each colored chart part was captured at the same position as the calibration chart, the determined positions and dimensions of the white patches could be used to extract RGB triplets for each patch on every chart part.

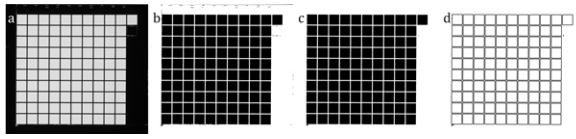

Figure 10: The process of detecting the contour of the color patches: (a) the image of the calibration chart as grayscale image: (b) the image of the calibration chart transformed to binary image: (c) the binary image with less noise: (d) the detected contours

The calibration itself started with capturing the calibration chart resulting in seven images (Section 2.4.5). The calibration chart image illuminated by white light were loaded as grayscale image by calling cv2.imread() with the option cv2.IMREAD_GRAYSCALE (Figure 10a). Calling cv2.threshold() with a threshold of 100, a maximum value of 255 and the type-variable set to cv2.THRESH_BINARY_INV converted the grayscale image into an inverted binary image (Figure 10 b). To remove smaller black parts in the background and smaller white parts in the black patches cv2.morphologyEx() was called with morphological operation set to cv2.MORPH_CLOSE and a 5x5 matrix filled with ones used as kernel (Figure 10c). These preparations optimized the image for calling cv2.findContours() with mode set to cv2.RETR_TREE and method set to cv2.CHAIN_APPROX_SIMPLE. Figure 10d shows the detected contours. To remove false detections, contours with an area (cv2.contourArea()) smaller than 20 000 pixels and greater than 1 000 000 pixels were filtered out. Finally, each contour was simplified to a rectangle (cv2.boundingRect()) defined by its position in the image (x, y) and its width and height. This information can be used to find the color patches in each image of a chart part.

2.4.7 Extracting RGB triplets from chart part captures

With the gathered positions and dimensions of each patch in the calibration chart each pixel of each patch in the colored chart parts is addressable. The RGB triplets of all pixels of a patch was reduced to a single RGB triplet. Therefore, each rectangle describing a patch (Section 2.4.6) was shrunk to half its size. Its position was corrected so that the rectangle is centered on the corresponding patch. The average of the RGB triplets of the pixels framed by a rectangle represented the RGB triplet of the corresponding color patch. Figure 11 visualizes the extracted information for the calibration chart.

As each color patch was present in seven images corresponding to the seven light sources, seven RGB triplets had to be extracted for each color patch. As the extraction was done per chart part, the results were gathered and sorted according to the sorting of the

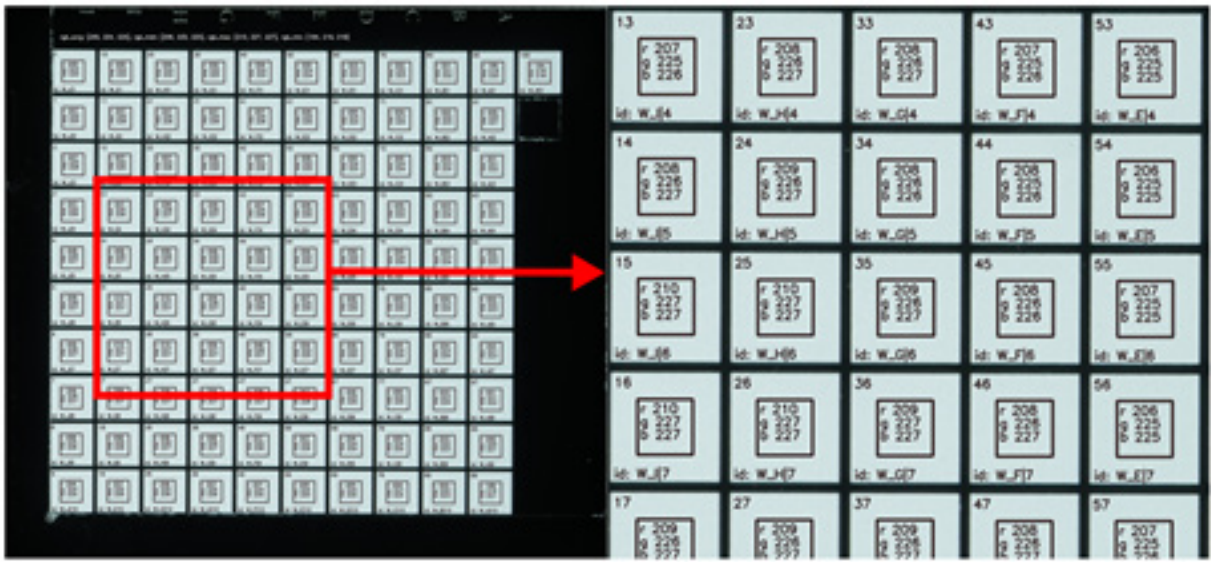

Figure 11: Exemplary RGB triplet extraction for color patches of the calibration chart with white illumination: in the top left of each patch is its number; the rectangle marks the extraction area while r, g, and b are the extracted values and ID describes the patch identity defined by the chart number (here w for white), row and column on the undivided chart

spectral reflectance array (Section 2.4.3). The resulting np.array had a shape of 10 800 by 21, where each row represented a color patch and each column represented a channel of an extracted RGB triplet (R_{white_ls}, G_{white_ls}, B_{white_ls}, …, R_{yellow_ls}, G_{yellow_ls}, B_{yellow_ls}).

2.4.8 Correcting unevenly distributed illumination

The extracted RGB triplets in Figure 11 also showed that the illumination was uneven. Although, all patches had the same color with the same spectral reflectance the corresponding RGB triplets differ. The minimum values are min(R) = 194, min(G) = 216, min(B) = 218 while the maximum values are max(R) = 210, max(G) = 227, max(B) = 227. To compensate this error for each patch (i) of the calibration chart and each channel of its extracted seven RGB triplets ($K\mathrm{cc}_{ij}$) corresponding to the seven light sources (j) a correction factor f_{ij}K was computed according to Equation 7. These factors were stored in an np.array. During extraction (Section 2.4.7) these factors were applied to the corresponding extracted values according to Equation 8.

$$f_{ijK} = \frac{\overline{\mathrm{K}cc}}{\mathrm{K}cc_{ij}} \qquad [7]$$

$$K\mathrm{crr}_{ij} = f_{ij\mathrm{K}}\, K_{ij} \qquad [8]$$

where $K\mathrm{crr}_{ij}$ is the corrected value for K, f_{ijK} is the correction factor, i is the number of patch in a chart part [0 ; 100], j is the illumination (white, red, green, blue, cyan, magenta, yellow), K is the extracted K value (RGB) of a patch of a color chart, Kcc is the extracted K value of a patch of the calibration chart and $\overline{Kcc}$ is the average of extracted K values of the patches of the calibration chart.

2.5 Splitting the datasets for training, validation and evaluation

Each dataset had to fulfill three purposes. The first purpose is to train the neural network while the second one is to validate the network during training. The third purpose is to test and evaluate the trained network. It is crucial that data used for training is not used for either validation or evaluation, which leads to three subsets. Due to the tests in this article further requirements had to be considered. Firstly, the patches of the subsets had to be equally distributed to represent a wide range of color combinations and to prevent a network to be biased to specific colors (for example reddish colors). Secondly, the subsets had to contain the same patches for each dataset. This enables to compare the performance of networks based on different datasets. It also enabled to test the performance of networks trained with a training subset of one dataset when evaluated with a test subset of another dataset. Thirdly, the training subset had to be reduced systematically. This enabled to test the performance of a network in dependence on the size of the training subset.

To select equally distributed color patches to form the subsets, three cases had to be considered. While the primary colors were distributed linearly, the secondary

Colors were distributed squarely and the tertiary colors were distributed cubically (Section 2.4.1). For each of these three cases, a step value was calculated according to Equation 9, where the splitting factor was the portion of patches of the dataset. The step value defined the step between color patches in each dimension. In the case of linear distribution, every step-th color patch was used for a subset as the color itself only changed in one dimension. In the case of squared distribution, every step-th color patch (respectively column) from every step-th row of the corresponding chart was part of the subset as one primary color changed per row and the other per column. In the case of cubic distribution, one primary color changed per row, the second per column, and the third per chart. Therefore, from every stepth chart, every step-th color patch from every step-th row was selected.

$$step = 20\left(20^{\,d}\ f_{\text{split}}\right)^{-\frac{1}{d}} = f_{\text{split}}^{-\frac{1}{d}} \qquad [9]$$

where step determines the frequency of selected patches (every step-th patch is included in the subset), f_{split} represents the splitting factor [0.0 ; 1.0] and d indicates the number of dimensions [1 ; 3]

In the best-case step is a natural number that directly points to the color patches to select. But it is more likely that step is a rational number. In that case the decimal part of step equaled an error variable and step was rounded off by calling np.floor(). The rounded off step was used to select the first color patch (or row, column or chart depending on the number of dimensions). For the distance between the last selected and the next color patch the current error was added to step. After that the decimal part again equaled the current error and step was rounded off. This process visualized for the linear, squared and cubic case in Figures 12 to 14. As a result, the error was minimized during selection.

For the test subset, the splitting factor was set to 0.2, and the color patches were selected as described. Depending on the test performed, the splitting factor for the training subset varied. For the main tests, it was set to 1.0. To ensure that each color patch appeared in only one subset, all color patches present in both the test and training subsets were deleted from the training subset. Afterwards, the subsets were shuffled by calling np.random.shuffle() with the seed fixed at 19840924. The splitting factor for the validation subset was set to 0.1. Unlike before, for the validation subset, the first *n* color patches in the shuffled training subset were extracted, where *n* was calculated according to Equation 10. The resulting subsets for the case described above are visualized in Figure 15. Table 1 shows the dataset shares of the subsets corresponding to the previously mentioned splitting factors.

$$n = ceil\left(N_{\text{ts}}\ \mathrm{f}_{split}\right) \qquad [10]$$

where *n* is the number of patches extracted from training

linear: splitting factor = 0.2, step = 5

step:					5.00					5.00					5.00				5.00
error:					0.00					0.00					0.00				0.00

Figure 12: Subset color patch selection in case of linear distribution (primary colors)

squared: splitting factor = 0.2, step = 2,24

step:		2.24		2.48		2.72		2.96			3.20		2.44		2.68		2.92		3.16
error:		0.24		0.48		0.72		0.96			0.20		0.44		0.68		0.92		0.16

Figure 13: One dimension of the subset color patch selection in case of squared distribution (secondary colors)

cubic: splitting factor = 0.2, step = 1,71

step:	1.71		2.42		2.13	1.84		2.55		2.26	1.97		2.68		2.39		2.10	1.81	2.52
error:	0.71		0.42		0.13	0.84		0.55		0.26	0.97		0.68		0.39		0.10	0.81	0.52

Figure 14: One dimension of the subset color patch selection in case of cubic distribution (tertiary colors)

subset, N_{ts} is the number of patches in the training subset and f_{split} is the splitting factor for validation [0.0 ; 1.0]

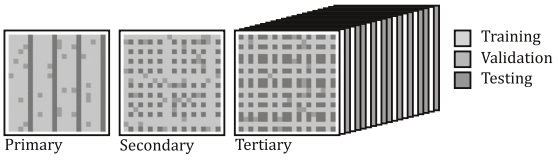

Figure 15: Exemplary RGB-value

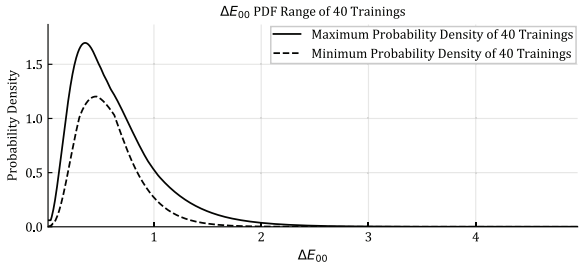

Figure 16: Range of the performance of 40 trainings using the same parameters and dataset

Table 1: Splitting factors and resulting dataset shares

	Training	Validation	Testing
Splitting factor	1.00	0.10	0.20
Dataset share [%]	70.88	7.88	21.24
Number of patches	7 655	851	2 294

2.6 Parameters of the neural network

The neural network was a densely connected feed forward network. It had one input layer (keras.Input()), three hidden layers (keras.layers.Dense()) and one output layer (keras.layers.Dense()). Its input layer had 21 Neurons corresponding to the seven RGB triplets of a color patch captured under seven different light sources. Its output layer had 36 neurons representing the spectral reflectance of the same color patch at wavelengths from 380 nm to 730 nm in 10 nm steps. Its first hidden layer had 180 neurons, its second hidden layer had 80 neurons and its third hidden layer had 40 neurons. The rectified linear unit function was the activation function of all neurons in the hidden layers, while the sigmoid function was the one of the neurons in the output layer. During training the loss between the measured and the predicted spectral reflectance was was determined by the mean squared error (MSE) using keras.losses.MeanSquaredError() with reduction set to auto and the other parameters set to default. The ADAM (Adaptive Moment Estimation)-algorithm (Kingma and Ba, 2017) (tf.keras.optimizers.Adam()) minimized the loss during training with its learning rate set to 0.001 and its other parameters set to default. A training ran for a maximum of 2 000 epochs with a batch size of 32. Training was stopped if the validation loss was not decreasing for 50 epochs. In the case of an early stop the best weights were restored (tf.keras.callbacks.EarlyStopping() with monitor set to 'val_loss', patience set to 50, mode set to 'min' and restore_best_weights set to true).

Before training the RGB triplets of the subsets (Section 2.5) were normalized by dividing by 255. The spectral reflectance values of the subsets were reduced to those obtained with measurement mode M2. The training was started by calling keras.model.fit() with the RGB triplets and the spectral reflectance of the training subset set as x and y, the validation subset used as validation data and the other values set as mentioned.

At the start of training, the weights of the neurons were set randomly. These starting weights mainly influenced the training outcome and therefore the performance of a trained network. Figure 16 showed the range of probability density functions (Section 2.7) of 40 trainings using the same parameters and dataset. Due to the randomly chosen starting weights, the performance of the trained networks varied. During pre-testing, 40 trainings were found to be sufficient to cover the entire performance range (which was close to the 41 repetitions used by Lazar, Javoršek, and Hladnik (2020)).

2.7 Evaluation and visualization of the training results

To evaluate a trained network or model, it predicted spectral reflectance from the RGB triplets of the test subset. The predictions were compared to the measured spectral reflectance using two metrics. The first one was the MSE between the predicted and the measured spectral reflectance, which also is the loss function during training. Therefore, the model should be optimized to this metric. The second and mainly used metric is the CIEDE2000 (ΔE_{00}) color difference. To compute the difference, the predicted and corresponding measured spectral reflectance were transformed to the CIELAB color space (Section 2.2).

These metrics were computed for each color patch of the test subset. To analyze the performance of the model as a whole, mean, median, standard deviation, minimum and maximum of these metrics were computed (Tables 2 and 3).

For further visualization the ΔE_{00} metric is used exclusively. To visualize the model performance as a whole a generalized gamma distribution (Stacy, 1962) was fitted to the ΔE_{00} values. Therefore, the ΔE_{00} values of the color patches were sorted ascending by calling

np.sort(). Calling scipy.stats.gengamma.fit() returned the shape parameters *a* and *c* as well as the scale and the location defining the fitted distribution. Calling scipy.stats.gengamma.pdf() with these parameters returns the probability density function (PDF) of the ΔE_{00}-values (e.g. Figure 17). Integrating the PDF on the intervals [0.0; 1.0[, [1.0; 2.0[and [2.0; 3.0[leads to the probability that the color difference lies within these intervals (see Table 2).

The PDF visualizes the model performance regarding the ΔE_{00} values for the whole test subset of a dataset. Furthermore, PDFs are used to visualize the performance of different models in comparison (see Figure 17).

A model with a higher and narrower density peak positioned further left performs better than a network with a lower and wider density peak located further right. The legend in Figure 17 shows the PDFs were sorted descending by their probability to predict a spectral reflectance that leads to a ΔE_{00} less than 1.0.

Regarding the naming each model is assigned to an id. The id consists of the year, month, day, hour, minute and second the training was started. For the PDFs the id is followed by abbreviation of the dataset used for training. In Tables 2–5, 9 and 10 the same dataset can be find in the 'DS' (dataset) column.

3. Results and discussion

3.1 Model performance depending on dataset and measurement mode

Therefore, out of the 40 training runs for each dataset the model with the highest probability to predict a spectral reflectance leading to a ΔE_{00} between 0.0 and 1.0 was chosen as the best performing model. Training with the 'pw' dataset resulted in the best performing model regarding ΔE_{00} (Table 2, Figure 17) and MSE (Table 3) followed by training with the 'p' dataset. Regarding ΔE_{00} the model trained with the '1x' dataset performed better than the one trained with the '2x' dataset (Table 2, Figure 17). Regarding MSE the model trained with '2x' performed better than the one trained with '1x' (Table 3). Due to blocking the optical brighteners during printing (Section 2.3.2) the M2 (UV-cut filter) measurement mode used for training (Section 2.6) should have less influence on the '1x' and '2x' datasets than on the 'p' and 'pw' datasets. To evaluate the influence of the measurement mode on the model performance, the test was repeated with the '1x' and 'pw' dataset. This time the spectral reflectance was obtained using measurement modes M0 and M1. Notation wise the dataset name was complemented by the measurement mode used; e.g. for 'pw-M0' measurement mode M0 was used to obtain the spectral reflectance of the 'pw' dataset. The resulting models were compared to the best performing model so far (20231027075227).

Changing the measurement mode changes the outcome of the training as well. While for 'pw' measurement mode M2 continued to lead to the best performing model, for '1x' M1 leads to the best model (20231122000640) regarding both ΔE_{00} and MSE (Tables 4 and 5, Figure 18). The probability to predict a spectral reflectance leading to a ΔE_{00} less than 1.0 increased from 0.897 (Table 2) with measurement mode M2 to 0.922 (Table 4).

In conclusion, the color management settings, the substrate and the measurement mode of the spectrophotometer used influence the performance of a model. The 'pw' trained model seems to perform slightly

better than the 'p' trained one because of using a wider CMYK simulation target profile (Section 2.4.2). The same color management settings in combination

Table 2: The ΔE_{00} statistics of the best performing models sorted descending by prob.[0;1[

Model id	DS	Mean	Median	StD	Min.	Max.	Prob.[0;1[	Prob.[1;2[	Prob.[2;3[
20231027075227	pw	0.515	0.461	0.280	0.035	2.246	0.938	0.061	0.001
20231006161556	p	0.528	0.479	0.286	0.048	2.177	0.930	0.068	0.001
20231025122410	1x	0.582	0.527	0.318	0.012	2.153	0.897	0.100	0.004
20231026133808	2x	0.737	0.629	0.474	0.030	4.120	0.778	0.201	0.018

Table 3: The MSE statistics of the best performing models sorted ascending by mean

Model id	DS	Mean	Median	StD	Min.	Max.
20231027075227	pw	1.35	6.81×10^{-6}	2.23×10^{-5}	3.50×10^{-7}	3.81×10^{-4}
20231006161556	p	1.38×10^{-5}	7.41×10^{-6}	2.12×10^{-5}	5.79×10^{-7}	2.91×10^{-4}
20231026133808	2x	1.41×10^{-5}	8.12×10^{-6}	1.81×10^{-5}	4.43×10^{-7}	2.45×10^{-4}
20231025122410	1x	1.53×10^{-5}	9.95×10^{-6}	1.95×10^{-5}	4.99×10^{-7}	3.84×10^{-4}

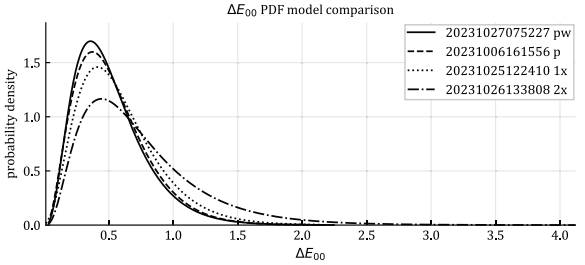

Figure 17: The ΔE_{00} PDFs of the best performing model from each dataset in comparison (ascending sorted by Prob.[0;1[in legend)

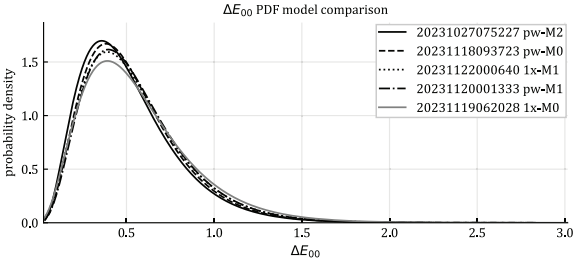

Figure 18: The ΔE_{00} PDFs of the model performance depending on the measurement mode

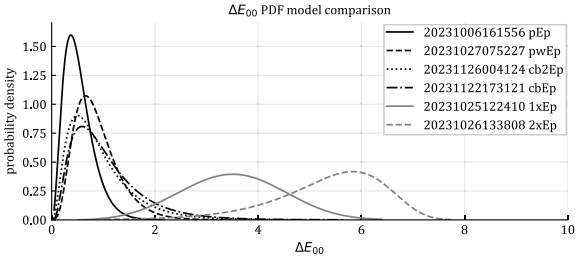

Figure 19: The ΔE_{00} PDFs of all models evaluated with the 'p' test subset

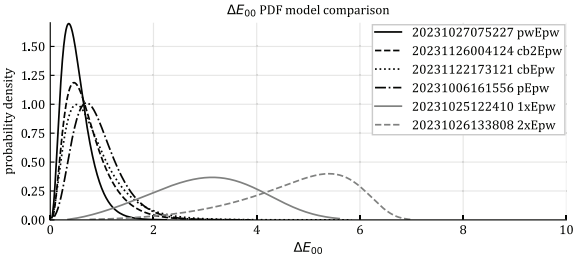

Figure 20: The ΔE_{00} PDFs of all models evaluated with the 'pw' test subset

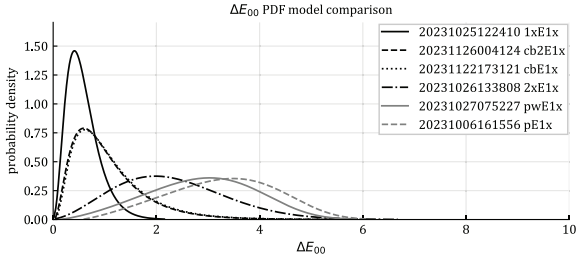

Figure 21: The ΔE_{00} PDFs of all models evaluated with the '1x' test subset

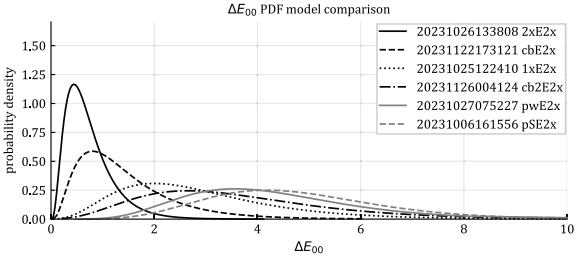

Figure 22: The ΔE_{00} PDFs of all models evaluated with the '2x' test subset

Table 4: The ΔE_{00} statistics for testing the influence of the measurement modes

Model id	DS	Mean	Median	StD	Min.	Max.	Prob.[0;1[	Prob.[1;2[	Prob.[2;3[
20231027075227	pw-M2	0.515	0.461	0.280	0.035	2.246	0.938	0.061	0.001
20231118093723	pw-M0	0.535	0.486	0.285	0.054	2.833	0.931	0.067	0.001
20231122000640	1x-M1	0.546	0.489	0.297	0.027	1.918	0.922	0.076	0.006
2023120001333	pw-M1	0.551	0.502	0.296	0.034	2.845	0.922	0.076	0.001
2023119062028	1x-M0	0.566	0.508	0.318	0.030	3.016	0.905	0.092	0.002

Table 5: The MSE statistics for testing the influence of the measurement modes

Model id	DS	Mean	Median	StD	Min.	Max.
20231027075227	pw-M2	1.35×10^{-5}	6.81×10^{-6}	2.23×10^{-5}	3.50×10^{-7}	3.81×10^{-4}
20231122000640	1x-M1	1.45×10^{-5}	9.84×10^{-6}	1.46×10^{-5}	4.50×10^{-7}	1.75×10^{-4}
20231119062028	1x-M0	1.56×10^{-5}	1.05×10^{-5}	1.67×10^{-5}	6.04×10^{-7}	2.10×10^{-4}
20231118093723	pw-M0	1.60×10^{-5}	8.67×10^{-6}	2.80×10^{-5}	6.26×10^{-7}	5.75×10^{-4}
20231120001333	pw-M1	1.99×10^{-5}	9.51×10^{-6}	5.73×10^{-5}	8.32×10^{-7}	1.66×10^{-3}

with the different substrates 'p' and '1x' are printed on cause the performance of the '1x' trained model to decrease in comparison. The performance of a '2x' trained model seems to decrease because of the 200% opacity (Section 2.4.2) compared to a '1x' trained one. The performance of a '1x' trained model increases when using measurement mode M1 instead of M2, seemingly, because these measurement modes deal differently with the fluorescence of ink and substrate. In theory, each of these assumptions should not influence the training of a neural network as long as the RGB triplets correlates well with the corresponding spectral reflectance. So, the final assumption is that the different datasets influence this correlation. Within the scope of this work the exact reasons are not clear and should be topic of further investigation.

3.2 Model generalization

This test examineed if a model is specialized on the dataset used for training or if it is generalized. Therefore, each model determined in 3.1 was evaluated with the test subsets of the datasets that are not related with its training. Furthermore, the datasets weree combined to train more generalized models. So, the training subsets of all datasets were combined to one training subset ('cb' for combined). The second combined training subset ('cb2') did not include the '2x' training subset. Regarding the notation for the PDF, the first subset is the subset the model was trained with followed by 'E' (for evaluate) as divider and then by the subset used for evaluation. So, 'pwE1x' shows the performance of a model trained with 'pw' and evaluated with the test subset of '1x'. Due to the number of models (26) the statistics were reduced to two values for each test and training subset combination. The two values are the mean ΔE_{00} with its added standard deviation (Table 6) and the probability that that the ΔE_{00} of a sample is less than 3.0 (Table 7).

In general, the performance of a model went down if the test subset was from another dataset than the training subset. Figures 19 to 22 each show the PDFs of the models evaluated with one test subset. For both test subsets 'p' (Figure 19) and 'pw' (Figure 20) the models trained with '1x' and '2x' performed the worst with a mean ΔE_{00} with its added standard deviation between 4.042 and 6.450 (Table 6) and a probability between 0.032 and 0.482 that a prediction results in a ΔE_{00} less than 3.0 (Table 7). The corresponding PDFs have a noticeably lower peak located far right from the other PDFs (Figures 19 and 20). The test subsets '1x' (Figure 21) and '2x' (Figure 22) each have two PDFs with similar characteristic corresponding to the models trained with 'pw' and 'p'. While the mean ΔE_{00} with its added standard deviation range between 3.926 and 6.741 (Table 6) is close to before, the probability that a prediction leads to a ΔE_{00} less than 3.0 between 0.137 and 0.528 (Table 7) is slightly better. The common property between '1x' and '2x' which differs from the common property of 'p' and 'pw' is the substrate. In contrast, the datasets 'p' and '1x' as well as 'pw' and '1x' are closer to each other than '1x' and '2x' (Figure 5). This observation lead to the assumption that accuracy of the prediction of a model increases if the sample is printed on the same substrate.

Overall, the non-combined datasets ('p', 'pw', '1x', '2x') have an average mean ΔE_{00} with its added standard deviation between 3.103 and 4.228 (Table 6). The two models trained with a combination of all datasets on the other hand perform noticeable better with an average mean ΔE_{00} with its added standard deviation of 1.868 for 'cb' and 2.629 for 'cb2'. The probability to predict a spectral reflectance resulting in ΔE_{00} less than 3.0 is between 0.928 and 0.994 for 'cb'. As 'cb2' does not include the '2x' training subset the probability decreases from 0.928 to 0.425 for the '2x' test subset compared to 'cb' (Table 7). The probability for the other test subsets increases slightly from 0.974 to 0.977 for '1x', 0.978 to 0.987 for 'p' and 0.994 to 0.997 for 'pw'.

In conclusion, training with a combination of multiple training subsets leads to a generalized model. If a single subset is neglected during training, the performance of the corresponding test subset decreases significantly more than the performance increases for the other test subsets. Furthermore, the performance of the generalized models decreases compared to the

Table 6: The mean ΔE_{00} added by the standard deviation of training and test subset combinations

Test	1x	2x	p	pw	cb	cb2
1x	0.900	3.365	4.337	3.926	1.840	1.795
2x	5.062	1.211	6.741	6.365	2.450	5.984
p	4.491	6.450	0.814	1.325	1.764	1.563
pw	4.042	5.885	1.362	0.795	1.419	1.175
avg.	3.624	4.228	3.314	3.103	1.868	2.629

Table 7: Probability that the ΔE_{00} of a sample is less than 3.0

Test	1x	2x	p	pw	cb	cb2
1x	1.000	0.755	0.385	0.528	0.974	0.977
2x	0.608	0.997	0.137	0.228	0.928	0.425
p	0.307	0.032	1.000	0.997	0.978	0.987
pw	0.482	0,078	0.997	1.000	0.994	0.997
avg.	3.624	4.228	3.314	3.103	1.868	2.629

Table 8: Splitting factors of training subsets and resulting dataset share

	0.15	**0.25**	**0.39**	**0.52**	**0.71**	**0.80**	**0.89**
Dataset share [%]	9.91	19.86	29.76	39.38	50.62	60.21	69.51
n-patches train	1 070	2 145	3214	4 253	5 467	6 503	7 507
n-patches validation	119	239	358	473	608	723	835

Table 9: The ΔE_{00} statistics of the models with reduced size of the training subset

Model id	**DS**	**Mean**	**Median**	**StD**	**Min.**	**Max.**	**Prob.[0;1[**	**Prob.[1;2[**	**Prob.[2;3[**
2023111145455	pwR89	0.524	0.469	0.293	0.018	2.931	0.933	0.066	0.001
2023111011123	pwR80	0.551	0.504	0.298	0.011	2.693	0.921	0.077	0.001
20231109211733	pwR71	0.579	0.524	0.302	0.050	2.857	0.907	0.091	0.001
20231109024923	pwR52	0.601	0.534	0.343	0.021	3.433	0.882	0.114	0.003
20231108163826	pwR39	0.621	0.562	0.339	0.024	3.322	0.872	0.124	0.003
20231106043622	pwR25	0.729	0.639	0.440	0.015	4.815	0.789	0.196	0.013
20231107221515	pwR15	0.776	0.679	0.475	0.052	5.504	0.754	0.225	0.018

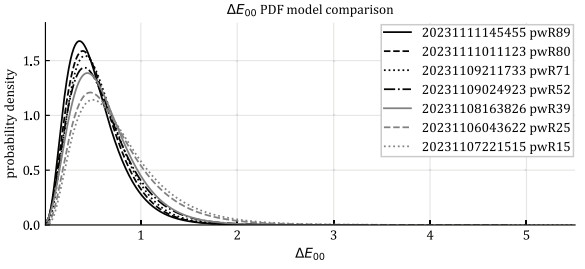

Figure 23: The ΔE_{00} PDFs of the models trained with reduced training subsets

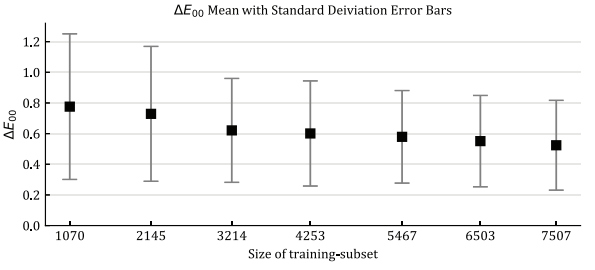

Figure 24: The ΔE_{00} means depending on the size of the training subset

performance of a model trained and evaluated with the subsets of the same dataset.

3.3 Decreasing size of the training subset

This test examines the performance of a model depending on the size of the training subset. Therefore, the 'pw' dataset that results in the best performing model (Section 3.1) is used for training. The size of the training subset is decreased from 70 % of the size of the dataset down to 10 % in 10 %-steps by reducing the splitting factor of the training subset (Section 2.5). Due to the reduction logic a splitting factor of 0.7 does not equal a share of 70% of the dataset. The used splitting factors and the resulting shares of the dataset are shown in Table 8. The dataset name is complemented by 'R' (for reduction test) followed by the corresponding splitting factor in percentage.

With the decreasing size of the training subset the peaks of the corresponding PDFs flatten out and shift to the right indicating a decreasing performance (Figure 23). If the number of color patches is reduced from 7507 (pwR89) to 1070 (pwR15) the probability to predict a spectral reflectance that leads to an error less than 3.0 ΔE_{00} decreases slightly from 1.0 to 0.997. If a ΔE_{00} less than 1.0 is needed the probability drops from 0.933 to 0.754 (Table 9). The expected error range lies between 0.231 and 0.817 ΔE_{00} for a training subset size of 7507 color patches. This range extends to a range from 0.301 to 1.251 ΔE_{00} if the training subset size is reduced to 1 070 color patches (Figure 24). In conclusion, the performance of a model decreases with the size of the training subset.

3.4 Number and combination of light sources

This test examined the performance of a model depending on the number and different combinations of light sources. Therefore, the RGB triplets (Section 2.4.7) were reduced and recombined. The tested light sources are white (W), a combination of red, green and blue (RGB) and a combination of cyan, magenta and yellow (CMY). Using two complementary colors should, in theory, cover the whole spectrum with no overlaps. That leads to the combination of red and cyan (RC), green and magenta (GM) and blue and yellow (BY). To test if more overlapping colors influence the performance a combination of cyan and yellow (CY) was added. For comparison the best performing

Table 10: The ΔE_{00} statistics of the models trained with different light sources

Model id	DS	Mean	Median	StD	Min.	Max.	Prob.[0;1[	Prob.[1;2[	Prob.[2;3[
20231027075227	WRGBCMY	0.515	0.461	0.280	0.035	2.246	0.938	0.061	0.001
20231207114655	CMY	0.569	0.502	0.321	0.028	2.213	0.902	0.096	0.004
20231213171513	CY	0.640	0.565	0.367	0.030	3.312	0.853	0.140	0.001
20231212090246	BY	0.660	0.584	0.375	0.044	2.744	0.839	0.154	0.003
20231205190528	RGB	0.669	0.598	0.387	0.025	3.104	0.832	0.159	0.007
20231210141326	GM	0.681	0.612	0.388	0.037	3.887	0.827	0.164	0.007
20231208225150	RC	0.697	0.611	0.401	0.038	3.668	0.814	0.176	0.009
20231203213010	W	0.878	0.749	0.580	0.048	5.265	0.680	0.278	0.035

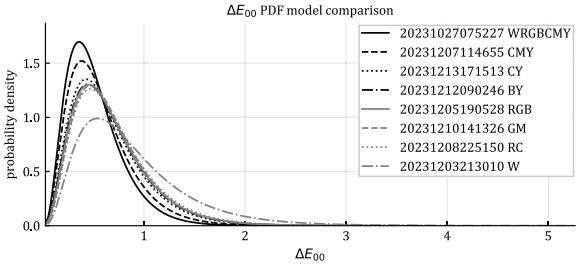

Figure 25: The ΔE_{00} PDFs of the models trained with different light sources

model from Section 3.1 (20231027075227) was used as it combines all seven light sources (WRGBCMY). Therefore, the other models were trained and evaluated using the 'pw' dataset. In Figure 25, the PDF with the highest peak located the furthest left corresponds to the combination of all light sources followed by the PDF of the CMY trained model. It is noticeable, that the PDFs of the models trained with the light source combinations CY, BY, RGB, GM and RC are close to each other whereby the PDF of CY has a slightly higher peak located more left (Figure 25). The PDF with the lowest peak located the furthest right corresponds to the W trained model. The results indicate that the model performance increases with the spectral overlap of the light sources used.

As long as two light sources were used for training the probability that the prediction leads to an error less than 3.0 ΔE_{00} was at least 0.999 (Table 10). If a ΔE_{00} less than 1.0 is needed the probability was at least 0.814 (Table 10). In the case of a single white light source the 3.0 ΔE_{00} probability was 0.993 while the 1.0 ΔE_{00} probability was 0.68 (Table 10). Finally, Figure 26 shows a multispectral image of a print processed by a model of this paper. It has to be mentioned, that the accuracy of this prediction was not tested. This is something to be done in the future.

4. Conclusion

In summary, the results show that a neural network can be used to predict the spectral reflectance of prints with at least acceptable tolerance. In practice it will be useful to have a more generalized network. Therefore, one should gather similar substrates with similar color management settings in groups and train a network for each group. Capturing thousands of color patches takes time (in this paper approximately 3 hours per dataset). If necessary this time can be reduced by reducing the number of light sources and the number of color patches. Reducing the number of color patches and light sources will not

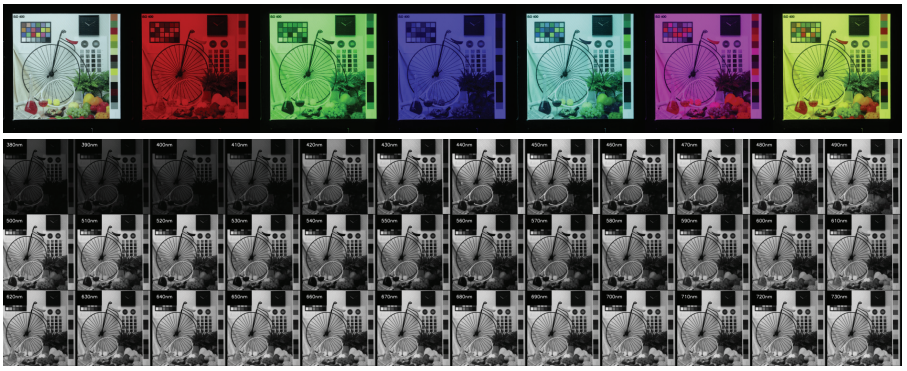

Figure 26: The print illuminated by the seven light sources (top) and the spectral reflectance predicted by the network of this paper (bottom). The printed image is part of the Image Quality Testform Package (Fogra, 2024).

only reduce the time needed for capturing but also the time needed for training. Although the reduction influences the performance negatively, the prediction results still are decent even in the worst scenarios. In fact, one has to find the perfect compromise regarding time and prediction accuracy for each use case. Finally, it should be mentioned that especially the light source has an influence on the research results. In this paper the light source was optimized for 3D scanning. Therefore, optimizing the light source for the purpose to reconstruct spectral reflectance should further improve the performance. Zhang, et al. (2021) identify such optimal projector spectra in a simulated environment.

References

Adobe Inc., 2023. Adobe InDesign (18.1). [computer program] Adobe Inc.. Available at: <https://www.adobe.com/de/products/indesign.html> [Accessed 07 August 2023].

Arad, B., Ben-Shahar, O., Timofte, R., Van Gool, L., Zhang, L., Yang, M. -H., Xiong, Z., Chen, C., Shi, Z., Liu, D., Wu, F., Lanaras, C., Galliani, S., Schindler, K., Stiebel, T., Koppers, S., Seltsam, P., Zhou, R., El Helou, M., Lahoud, F., Shahpaski, M., Zheng, K., Gao, L., Zhang, B., Cui, X., Yu, H., Can, Y. B., Alvarez-Gila, A., van de Weijer, J., Garrote, E., Galdran, A., Sharma, M., Koundinya, S., Upadhyay, A., Manekar, R., Mukhopadhyay, R., Sharma, H., Chaudhury, S., Nagasubramanian, K., Ghosal, S., Singh, A. K., Singh, A., Ganapathysubramanian, B. and Sarkar, S., 2018. NTIRE 2018 Challenge on spectral reconstruction from RGB images. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (CVPRW). Salt Lake City USA, 18–22 June 2018. https://doi.org/10.1109/CVPRW.2018.00138.

Arad, B., Timofte, R., Ben-Shahar, O., Lin, Y. -T., Finlayson, G., Givati, S., Li, J., Wu, C., Song, R., Li, Y., Liu, F., Lang, Z., Wei, W., Zhang, L., Nie, J., Zhao, Y., Po, L. -M., Yan, Q., Liu, W., Lin, T., Kim, Y., Shin, C., Rho, K., Kim, S., Zhu, Z., Hou, J., Sun, H., Ren, J., Fang, Z., Yan, Y., Peng, H., Chen, X., Zhao, J., Stiebel, T., Koppers, S., Merhof D., Gupta, H., Mitra, K., Fubaram, B.J., Sedky, M., Dyke, D., Banerjee, A., Palrecha, A., Sabarinathan, S., Uma, K., Vinothini, D.S., Sathya Bama, B. and Roomi, S.M.M., 2020. NTIRE 2020 Challenge on Spectral Reconstruction from an RGB Image. In: IEEE, IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Virtual, 14–19 June 2020. https://doi.org/10.1109/CVPRW50498.2020.0023.

Arad, B., Timofte, R., Yahel, R., Morag, N., Bernat, A., Cai, Y., Lin, J., Lin, Z., Wang H., Zhang, Y., Pfister, H., Van Gool, L., Liu, S., Li, Y., Feng, C., Lei, L., Li, J., Du, S., Wu, C., Leng, Y., Song, R., Zhang, M., Song, C., Zhao, S., Lang, Z., Wei, W., Zhang, L., Dian, R., Shan, T., Guo, A., Liu, J., Agarla, M., Bianco, S., Buzzelli, M., Celona, L., Schettini, R., He, J., Xiao, Y., Xiao, J., Yuan, Q., Kwon, T., Ryu, D., Bae, H., Yang, H.-H., Chang, H.-E., Huang, Z.-K., Chen, W.-T., Kuo, S. -Y., Chen, J., Li, H., Sabarinathan, S., Uma, K., Bama, B. S. and Roomi, S.M.M., 2022. NTIRE 2022 Spectral Recovery Challenge and Data Set. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. New Orleans USA, 19–24 June 2022. https://doi.org/10.1109/CVPRW56347.2022.00102.

Bradski, G., 2000. The OpenCV Library. Dr. Dobb's Journal of Software Tools. vol. 25, no. 11, pp. 120–125.

Drouin, M. and Beraldin, J., 2020. Active Triangulation 3D Imaging Systems for Industrial Inspection. In: Liu, Y., Pears, N., Rosin, P.L., Huber, P.ed. *3D Imaging, Analysis and Applications*. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-44070-1.

International Organization for Standardization. 2020a. *ISO/CIE 11664-1:2019, Colorimetry – Part 1: CIE standard colorimetric observers*. Berlin: Beuth Verlag GmbH.

International Organization for Standardization. 2022a. *ISO/CIE 11664-2:2022, Colorimetry – Part 2: CIE standard illuminants.* Berlin: Beuth Verlag GmbH.

International Organization for Standardization. *2020b. ISO/CIE 11664-3:2019, Colorimetry – Part 3: CIE tristimulus values*. Berlin: Beuth Verlag GmbH.

International Organization for Standardization. 2020c. *ISO/CIE 11664-4:2019, Colorimetry – Part 4: CIE 1976 L*a*b* colour space*. Berlin: Beuth Verlag GmbH.

International Organization for Standardization. 2023. *ISO/CIE DIS 11664-5:2023, Colorimetry – Part 5: CIE 1976 L*u*v* colour space and u', v' uniform chromaticity scale diagram*. Berlin: Beuth Verlag GmbH.

International Organization for Standardization. 2022b. *ISO/CIE 11664-6:2022, Colorimetry – Part 6: CIEDE2000 colour-difference formula.* Berlin: Beuth Verlag GmbH.

Fogra Forschungsinstitut für Medientechnologien e.V., 2024. Image Quality Testform Package [pdf] Fogra Forschungsinstitut für Medientechnologien e.V.. Available at: < https://fogra.org/fileadmin/files/7_downloads/Testformen/ImageQuality_Testforms_2024_Jan.pdf > [Accessed 11 March 2024]

gPhoto - opensource digital camera access and remote control, 2022. gphoto2 (2.5.28). [computer program] Available at: < http://www.gphoto.org/ > [Accessed 08 December 2023].

Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., Del Río, J.F., Wiebe, M., Peterson,

P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. and Oliphant, T.E. 2020, Array programming with NumPy, Nature. [e-journal] vol. 585, no. 7825, pp. 357–362. https://www.doi.org/10.1038/s41586-020-2649-2.

Hunter, J.D. 2007, Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering. [e-journal] vol. 9, no. 3, pp. 90–95. https://www.doi.org/10.1109/MCSE.2007.55.

Kingma, D. P. and Ba, J. 2017, Adam: A Method for Stochastic Optimization [PDF]. https://doi.org/10.48550/arXiv.1412.6980.

Lazar, M., Javoršek, D., i Hladnik, A. 2020, Study of Camera Spectral Reflectance Reconstruction Performance using CPU and GPU Artificial Neural Network Modelling. Tehnički vjesnik. [e-journal] vol.27, no. 4, pp. 1204-1212. https://doi.org/10.17559/TV-20190526202030.

Lazar, M. and Hladnik, A., 2023. Comparison of Artificial Neural Network and Polynomial Approximation Models for Reflectance Spectra Reconstruction. Sensors. [e-joural] vol. 23, no. 2. https://doi.org/10.3390/s23021000.

Mansencal, T., Mauderer, M., Parsons, M., Shaw, N., Wheatley, K., Cooper, S., Vandenberg, J.D., Canavan, L., Crowson, K., Lev, O., Leinweber, K., Sharma, S., Sobotka, T. J., Moritz, Do., Pppp, M., Rane, C., Eswaramoorthy, P., Mertic, J., Pearlstine, B., Leonhardt, M., Niemitalo, O., Szymanski, M., Schambach, M., Huang, S., Wei, M., Joywardhan, N., Wagih, O., Redman, P., Goldstone, J., Hill, S., Smith, J., Savoir, F., Saxena, G., Chopra, S., Sibiryakov, I., Gates, T., Pal, G., Tessore, N., Pierre, A., Thomas, F.-X., Srinivasan, S. and Downs, T., 2022. Colour 0.4.2. https://doi.org/10.5281/zenodo.7367239.

Park, J. -I., Lee, M. -H., Grossberg, M. D. and Nayar, S. K., 2007. Multispectral Imaging Using Multiplexed Illumination. In: IEEE, IEEE 11th International Conference on Computer Vision. Rio de Janeiro, Brazil, 2007, pp. 1-8. https://www.doi.org/10.1109/ICCV.2007.4409090.

Roland Digital Group Corporation 2023, VersaWorks 6 RIP Software (6.17.0). [computer program] Roland Digital Group Corporation Available at: < https://www.rolanddg.eu/de/produkte/software/versaworks-6-rip-software> [Accessed 11 March 2024].

Stacy, E. W. 1962. A Generalization of the Gamma Distribution. The Annals of Mathematical Statistics, vol. 33, no. 3, pp. 1187–1192 [Online]. Available at: <http://www.jstor.org/stable/2237889> [Accessed 11 March 2024]

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J.., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F. and van Mulbregt, P. 2020, SciPy 1.0: fundamental algorithms for scientific computing in Python, Nature methods. [e-journal] vol. 17, no. 3, pp. 261–272. https://www.doi.org/10.1038/s41592-019-0686-2.

TensorFlow Developers, 2023. TensorFlow. https://doi.org/10.5281/zenodo.8118033

Zhang, J., Meuret, Y., Wang, X. and Smet, K. A. G. 2021, Improved and Robust Spectral Reflectance Estimation. LEUKOS. [e-joural] vol. 17, no. 4, pp. 359–379. https://doi.org/10.1080/15502724.2020.1798246

Zhang, J., Su, R., Fu, Q., Ren, W., Heide, F. and Nie, Y. 2022, A survey on computational spectral reconstruction methods from RGB to hyperspectral imaging. Scientific Reports. vol. 12, no. 1, p. 11905. https://doi.org/10.1038/s41598-022-16223-1