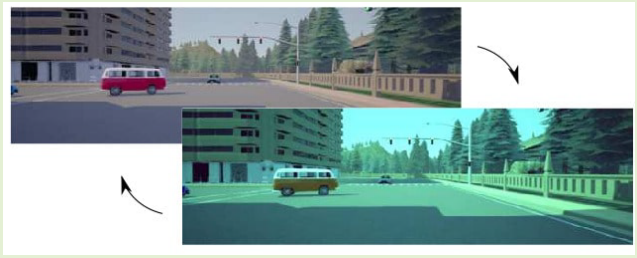
Discussion of Novel Filters and Models for Color Space Conversion

Kamil Lelowicz, *Member, IEEE*, Michał Jasiński, *Member, IEEE*, and Adam Krzysztof Piłat, *Member, IEEE*

***Abstract*—In the era of artificial intelligence perceptual algorithms used in state-of-the-art Advanced Driver Assistance Systems (ADAS), algorithm validation is not an easy task. To ensure the highest possible safety level of the solution, the performance of the algorithm must be evaluated under a variety of challenging conditions. To test the algorithms, simulators are used to emulate the virtual environment around the car taking into account road traffic, infrastructure and vehicles dynamics. Sensor models are necessary for virtual testing to provide required data to ADAS algorithms. This article introduces the issue of modeling the color filter spaces that are used in the automotive industry. The images generated by the simulator usually have RGB color. In contrast, the automotive industry uses filters such as RCCC and RYYCy. In this paper, the methods for transforming color space from RGB to RYYCy are discussed. Three novel approaches are introduced to solve this problem: analytical, polynomial, and based on a neural network. Moreover, comparative discussion of the presented solutions is shown and with the set of experiments the conversion accuracy and execution time of each algorithm are compared. In addition, introduced solution were compared with modified models that are presented in the literature.**



***Index Terms*—Cameras, virtual validation, mathematical modeling.**

I. Introduction

DIGITAL imaging devices, such as digital cameras, are used in automotive applications to record the environment around the vehicle as well as the interior of the cabin. Output data of such imaging devices is usually fed to neural networks [1] to extract information from the surrounding environment such as positions or trajectories of other vehicles [2], [3], road markings, traffic signs [4]. These image processing devices may be a part of Advanced Driver-Assistance Systems (ADAS), which usually gather and fuse data from multiple sensors, such as lidars, radars and cameras. ADAS systems are designed to monitor of car environment, detect potentially dangerous situation and warn driver about it or even take control (fully or partially) to avoid accident [5], [6]. It makes the system immensely complex. Thus, a key challenge is testing of ADAS functions. To prove the robustness of algorithms million miles of test drives should be conducted [7]. It is very costly and time-consuming to drive these miles under real-world conditions. Virtual simulations provide the ability to perform these tests, thus virtual testing is a essential topic in developing the functionality of ADAS algorithms. The flow chart in Figure 1 depicts virtual test-bench with camera sensor model and corresponding data flow. The RGB image is rendered by virtual simulation. Camera sensor model converts RGB image to raw RCCC (C - clear) or RYYCy (Y - yellow, Cy - cyan) format and adds distortion to the image to make it as similar as possible to the image from the real camera. The output from the sensor model is propagated to the ADAS module, which controls vehicle functions such as steering and acceleration. This affect the dynamics of the vehicle. The adjusted state of the vehicle dynamical model is fed back to the simulator. This article focuses on the topic of modeling camera sensors, and in particular on the topic of modeling RYYCy color filters that are used in the automotive industry.

A. Color Filters Used in an Automotive Application

An image sensor (imager) is a device that captures and transmits information being later used for image generation. More specifically, imager transforms the alternating attenuation of light waves into small bursts of current [8]. Two main types of imagers are available: charge-coupled device (CDD) and complementary metal–oxide–semiconductor (CMOS).

Manuscript received 17 March 2022; accepted 5 April 2022. Date of publication 10 June 2022; date of current version 14 July 2022. This work was supported by the Polish Ministry of Science and Higher Education carried out in cooperation of Aptiv Services Poland S.A.—Technical Center Kraków and AGH University of Science and Technology—Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering under Project 0014/DW/2018/02. The associate editor coordinating the review of this article and approving it for publication was Dr. Daniele Tosi. *(Corresponding author: Adam Krzysztof Piłat.)*

The authors are with the Department of Automatic Control and Robotics, AGH University of Science and Technology, 30-059 Kraków, Poland (e-mail: lelowicz@agh.edu.pl; jasinski@agh.edu.pl; ap@agh.edu.pl).

Digital Object Identifier 10.1109/JSEN.2022.3169805

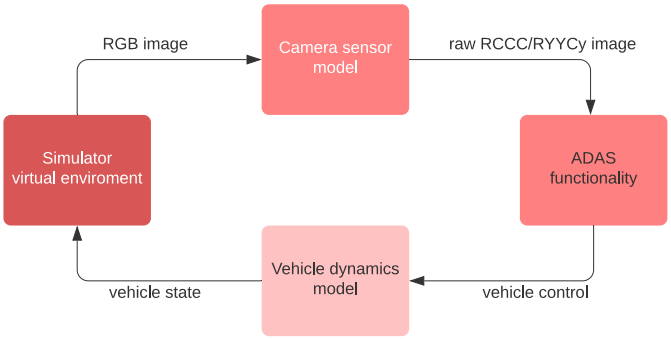


Fig. 1. Data flow in virtual test-bench with camera sensor model.

CCDs and CMOS imagers are quite common, since they similarly accumulate photo generated charge in each pixel proportionally to the local illumination intensity. However, CDD sensors allows for creation low-noise images. CMOS sensors are more susceptible to noise [9]. Neverthless, CDDs consume much more power than an equivalent CMOS sensor and are less cost efficient. On top of the image receiving surface of a given imaging device a mosaic filter, being used for sensing color information, and microlens array, which condenses the incident light on each pixel, are placed [10]. Color filters block the incident light according to the wavelength range so that the information about the light color is split into a few separately filtered intensities. The most common one is Bayern filter [11], which gives information about the intensity of light in red, green, and blue (RGB) wavelength regions (RGB channels). However, in automotive industry typical RGB cameras are rarely used [4], [12], [13].

More often, instead of RGB filtering, automotive cameras incorporate color arrays with red and triple clear channels (RCCC). The RCCC sensor is similar to a monochrome sensor. Thus, is more sensitive and offers better reproduction of details, but still provides the separate red color information [14]. In case of solutions used in the automotive industry for the environment perception, the grey scale image interpolated from clear pixels is employed for cars obstacles, pedestrian and other road users. The color information from red channel is needed for the detection of traffic lights, vehicle backlights and traffic signs [15], [16]. However, the information about the color coming from red channel only is not sufficient for the correct perception, under all environment conditions. Therefore, a new RYYCy array color matrix was invented [12], [13], [17], where RYYCy represent red, double yellow and cyan channels respectively. Newly proposed filter was introduced to precisely distinguish between white and yellow road lines and, at the same time, to maintain the good performance in low light quality by using wide-bandwidth yellow filter.

B. Related Work

Algorithms for image analysis are usually designed and validated using simulation systems that emulate the environment surrounding the vehicle and provide image data representing the field of view captured by the cameras [18]. The most popular simulators are: dSPACE ASM Traffic [19], CarMaker [20] and open source CARLA [21]. Any image processing algorithm can be explicitly tested within a given virtual environment, by providing image data as input and by analyzing the information extracted from that image. It is possible to automatically verify the algorithm output by comparing it with ground-truth data generated by the simulation system itself [22], [23].

For reliable testing, the image data provided by virtual simulations should match the output from real cameras as close as possible. Physical camera models required Physically Based Ray Tracing (PBRT) method. Geometric optics is used to compute how light propagates from a light source to camera aperture taking into account reflection of objects in the scene [24] and including effect of camera multi-element spherical lenses [25]. This allows the calculation of spectral irradiance that is used to predict an array of pixel responses using phenomenological model of the image sensor array [26], [27]. Thus, arbitrary set of color filters can be simulated.

However, in order to test the algorithms in real-time applications, the image data should be triggered with the same frequency as the real imaging device, for example a frame rate of an imaging device configured as video camera [28]. Several approaches to different fidelity level can be incorporated for simulation using a number of testing methods: hardware in the loop (HiL), software in the loop (SiL), processor in the loop (PiL). Hardware in the loop tests by their nature require real-time simulation. For SiL tests, the simulation execution time must meet higher requirements. Thousand of virtual kilometres are needed to be covered in the shortest possible time to test each new software version used in cars. Thus, execution time of the models has to be as low as possible. Accordingly, there is a need to not only accurately but also very efficiently reproduce the output of a digital imaging device based on given image data representing a scene to be captured by the imaging device.

Accurate physical models of cameras require huge computational effort, which causes limitations in using these solutions in real-time simulations. Nevertheless, currently all available simulators are able to generate RGB camera output using simplified models. Therefore, one way to solve the problem is to find out, if there is an effective and accurate method converting an RGB image into a raw RYYCy image. The conversion of an RGB image to raw RCCC requires a calculation for the clear channel (C) which can be realised as a weighted average of the channels R, G, and B. For the conversion to RYYCy no explicit method is known.

Gossett and Chen [29] proposed conversion from RGB to RYB space which is similar to RYYCy. Color mixtured in RGB is handled in additively, in our case the same should apply to RYYCy. However, in this article RYB is a subtractive color space, which reflects the behavior of colors when pigments are mixed. The proposed transformation model uses trilinear interpolation assuming that eight vertex points of the transformation are defined. The values of colors in vertex points were chosen based on Itten's suggestions [30]. In [31] Sugita and Takahashi presented a method for interconversion between RGB and RYB color spaces. In this case RYB is also modeled as a subtractive color space. The study explicitly

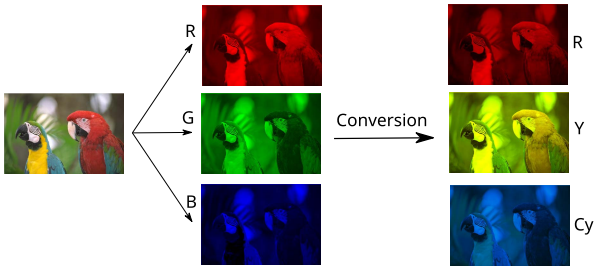


Fig. 2. RGB to RYCy conversion.

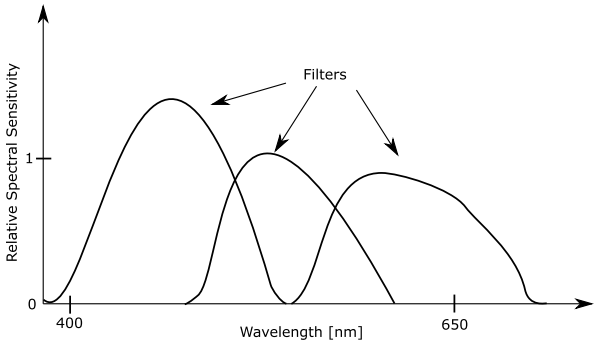


Fig. 3. Example of a mosaic filters.

presents the equations that allow the transformation of RGB space to RYB and the inverse transformation.

C. Contribution

The main goal was to develop algorithms for transforming color space between different filters and comparing them with each other. The authors introduce an analytical model [32] and proposed two novel methods to solve the conversion problem from RGB to RYYCy color space (Figure 2). Additionally, two methods from the literature [31] and [29] were modified and adapted to also solve the presented problem. An experimental analysis of the presented solutions was carried out. The accuracy of the conversions was compared, and the influence of noise on the obtained results was investigated. Additionally, the algorithms were implemented on GPU in a form allowing easy incorporation of the methods into existing simulation systems and an analysis of their execution times was presented.

II. Existence of Color Conversion Solution for Different Sets of Filters

The response of the imager with mosaic filter can be accurately modeled by a linear system, defined using spectral sensitivity function of each filter [33]. If the spectral distribution of light incident on the imager is given by $f(\lambda)$, where λ represents wavelength, the responses of the three cones can be modeled as vector components, given by:

$$c_i = \int_{\lambda_{min}}^{\lambda_{max}} s_i(\lambda) f(\lambda)\, d\lambda \quad i = 1, 2, \ldots l \tag{1}$$

where:

- s_i denotes the sensitivity of the i-th color filter.
- l denotes number of filters.
- λ_{max} and λ_{min} denote the interval of wavelengths outside of which all these sensitivities are zero.

Mathematically, the equation 1 describes the inner product, defined in Hilbert space, of square integrable functions $\mathcal{L}^2[\lambda_{min},\lambda_{max}]$. Thus, the filter response corresponds to a projection of the spectrum onto space spanned by the sensitivity functions $\{s_i(\lambda)\}_{i=1}^{l}$. In the study RGB and RYYCy color arrays utilized three separate filters. Thus, $l = 3$ (Figure 3).

Nevertheless, the nature of light is quantified. Consequently, the equation (1) may be replaced by its sampled counterparts [33]. If n uniformly spaced samples are used over the visible range and $[\lambda_{min},\lambda_{max}]$ and $n \gg 3$ the equation is as follows

$$c_i = \sum_{k=1}^{n} s_i(\lambda_i) f(\lambda_i) \Delta\lambda = s_i^T \mathbf{f} \quad i = 1, 2, 3 \tag{2}$$

where:

- $\Delta\lambda$ is the wavelength sampling interval.
- $\{\lambda\}_{i=1}^{n} - 1$ are uniformly spaced wavelengths covering the visible region of spectrum and $\lambda_i = \lambda_0 + i\,\Delta\lambda$.
- $\mathbf{f} = [f(\lambda_0), f(\lambda_1), \ldots, f(\lambda_{n-1})]^T$ is $n \times 1$ vector of samples of $f(\lambda)$.
- $s_i = \Delta\lambda[s_i(\lambda_0), s_i(\lambda_1), \ldots, s_i(\lambda_{n-1})]$ is $n \times 1$ vector of samples of $s_i(\lambda)$ scaled by the interval $\Delta\lambda$.

For simplicity of notation, the equation (2) can be rewritten as:

$$\mathbf{c} = \mathbf{S}^T \mathbf{f} \tag{3}$$

where:

- $\mathbf{c} = [c_1, c_2, c_3]^T$.
- $\mathbf{S} = [s_1, s_2, s_3]$ is the $n \times 3$ matrix with filters sensitivity vectors as its columns.

A. Proposition

For the same spectral distribution of light incident on the imager and two different sets of filters, one can get:

$$\mathbf{c}_1 = \mathbf{S}_1^T \mathbf{f} \tag{4}$$

$$\mathbf{c}_2 = \mathbf{S}_2^T \mathbf{f} \tag{5}$$

To find a function such as $\mathbf{c}_1 = \mathbf{S}_1^T \mathbf{S}_2^{+} \mathbf{c}_2$, converting color from one filter space to another, one has to find matrix S_2^{+}, for which $S_2 = S_2^T$. Such a matrix $\mathbf{S}_2^{+}$ exists only when its transposition, S_2^T, has linearly independent columns [34]. Unfortunately, this condition is not met, since matrix $\mathbf{S}_2^T$ is $3 \times n$ where $n \gg 3$, and hence matrix S_2^{+} does not exist. Instead, for a given spectral distribution $\mathbf{f}_j$, one can solve the following equation:

$$\mathbf{P}_j \mathbf{c}_2 = \mathbf{f}_j \tag{6}$$

then the color conversion equation is defined as follows:

$$\mathbf{c}_1 = \mathbf{S}_1^T \mathbf{P}_j \mathbf{c}_2 \tag{7}$$

Thus, for each spectral distribution there is an equation that is able to convert c_1 to c_2. For different spectral distribution equation (6) in the following form can be transformed into:

$$\left(\mathbf{P}_j + \Delta\mathbf{P}\right)\left(\mathbf{c}_2 + \Delta\mathbf{c}\right) = \mathbf{f}_j + \Delta\mathbf{f} \tag{8}$$

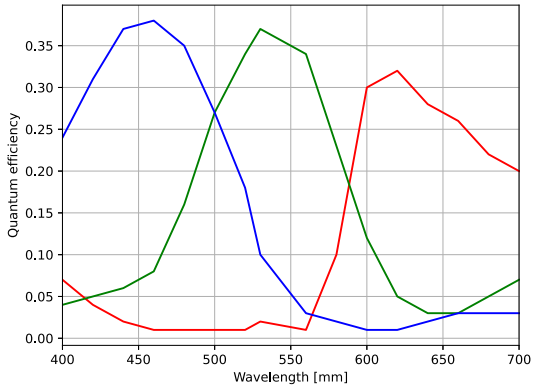


Fig. 4. The amount of red, green and blue colors as a functions of light wavelengths in the RGB color filter.

from equation (3), one can get:

$$\mathbf{c} + \Delta\mathbf{c} = \mathbf{S}^T (\mathbf{f} + \Delta\mathbf{f}) \tag{9}$$

thus:

$$\Delta\mathbf{c} = \mathbf{S}^T \Delta\mathbf{f} \tag{10}$$

after substitution (10) to (8) and simplifications one can get:

$$\Delta\mathbf{P}\mathbf{c}_2 = \left(\mathbf{I} - \left(\mathbf{P}_j + \Delta\mathbf{P}\right)\mathbf{S}_2^T\right)\Delta\mathbf{f} \tag{11}$$

It is worth noticing that for small $\Delta\mathbf{f}$ in Frobenius norm sense, $\Delta\mathbf{P}$ can be also small. Consequently, for small deviations of spectral distribution $\mathbf{f}$ changes in matrix $\mathbf{P}$ should be small as well. Hence, there is a possibility that exists a model in the form of:

$$\mathbf{c}_1 = j(\mathbf{f})\mathbf{c}_2 \tag{12}$$

which can be presented in a more general form:

$$\mathbf{c}_1 = g(\mathbf{c}_2) \tag{13}$$

where $g : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is a continuous function. However, two spectral distributions can produce the same vector $\mathbf{c}$. Consequently, the general solution for this particular problem does not exist, due to the ambiguity of the transformation.

III. Research

A. Analytical Model - RGB2RYYCyAna

The inputs to the algorithm are [32]:

- sRGB image or sRGB stream from virtual simulation.
- Camera spectral sensitivity charts, each describing the relative amount of light detected by a given color filter as a function of light wavelengths (Figure 4).

As mentioned above, the algorithm solves the problem of converting sRGB color space into some custom color space, i.e. RCCC or RYYCy. To achieve this goal, the sRGB color space is converted into CIE 1931 xyY color space [35] (Figure 5). Then, the CIE xy chromacity diagram and the dominant wavelength definition are used in order to estimate the light wavelength of the given sRGB color (Figure 6). After the wavelength recovery, the camera spectral sensitivity functions are used to get the amount of primary colors of the output color filter. Finally, in order to be able to display the

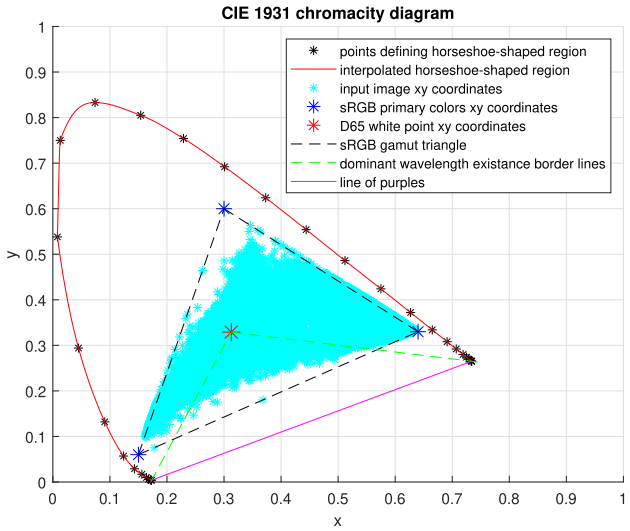


Fig. 5. Chromacity diagram with the sRGB gamut triangle. Own elaboration based on [36], [37].

output image in the simulation environment, the RGB channels used in simulation are adjusted to the amount of primary colors of the output color filter.

The algorithm consists of multiple steps, leading to the converted image, and incorporate ideas that improve the quality of the conversion.

1) Convert sRGB Color Space Into CIE 1931 xyY Color Space: In this step, first an sRGB triple into CIE 1931 XYZ color space is converted. For this purpose, a well-known matrix multiplication of the linear sRGB values is used. The numerical values below match those in the official sRGB specification [36]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R_{srgb} \\ G_{srgb} \\ B_{srgb} \end{bmatrix} \tag{14}$$

Linear sRGB values can be obtained as follows:

$$\begin{cases} C_{linear} = \dfrac{C_{srgb}}{12.92} & \text{for } C_{srgb} \leq 0.04045 \\ C_{linear} = \left(\dfrac{C_{srgb} + \alpha}{1 + \alpha}\right)^{2.4} & \text{for } C_{srgb} > 0.04045 \end{cases} \tag{15}$$

where $\alpha = 0.055$ and C is either R, G or B and it has value between 0 and 1. Next, the CIE 1931 XYZ color space is converted into CIE 1931 xyY color space, where upper case Y is the parameter representing luminance, which is constant during conversion, and subsequently lower case x and y represent chromacity coordinates easily calculated from X, Y and Z:

$$x = \frac{X}{X + Y + Z} \tag{16}$$

$$y = \frac{Y}{X + Y + Z} \tag{17}$$

2) Recover the Wavelength Value of the Input Color: In the wavelength recovery process, a key role is played by the CIE 1931 xy chromacity diagram shown in Figure 5.

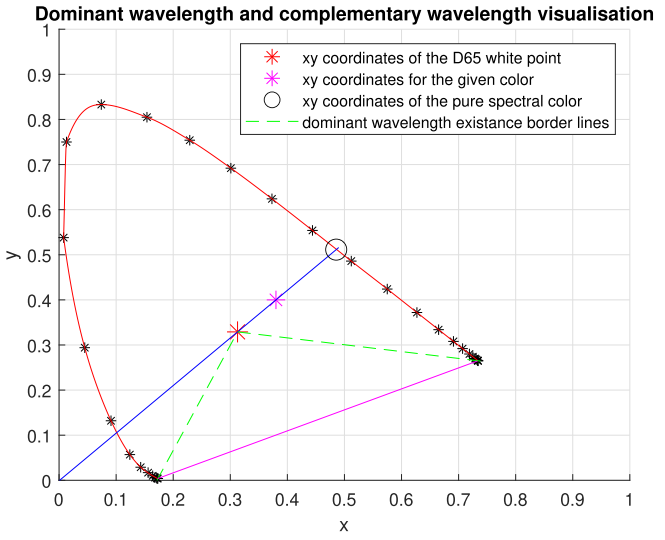


Fig. 6. Dominant wavelength and complementary wavelength visualisation.

Several important remarks have to be mentioned about this diagram:

- All colors from the sRGB color space lie inside the black dashed triangle.
- The locus of the points that define horseshoe-shaped region are the chromacity coordinates for pure spectral colors and each spectral color corresponds to a single wavelength of the visible light.
- Coordinates of the points on the horseshoe-shaped region and wavelengths corresponding to that points are tabulated with 1 nm wavelength interval.
- Colors that lie on the line of purples are non-spectral so no corresponding wavelength exist for these colors.
- Colors that lie inside the horseshoe-shaped region are the mixture of the pure spectral colors.

The consequence of the above mentioned facts is that colors from sRGB space do not have dominant wavelengths corresponding to them. However, some of the colors from sRGB can be assigned to a proper dominant wavelength value. In order to get a dominant wavelength value, the following steps should be taken (Figure 6):

- Create a line between the point corresponding to a given color and the D65 white point.
- Find the intersection point of the created line with the horseshoe-shaped region.
- Take the xy coordinates of the found point being the coordinates of the pure spectral color corresponding to the given color.
- Read the wavelength value for the pure spectral color based on its xy coordinates, which equals to the dominant wavelength value.

Unfortunately, not for all colors (sRGB triples) the dominant wavelength is explicitly estimated. In fact, there are three different cases of the wavelength recovery process discussed below:

- If the color is in grayscale range, with a given threshold and tolerance, namely, if all elements of the given sRGB triple are greater than 130 and the difference between all the sRGB triple values and its median is less than 50, the dominant wavelength is not calculated, assuming that all RGB channels take values between 0 and 255. The closer the colour is to the D65 point, the influence of the dominant wavelength is less.. In this case wavelengths of other lengths also have a large influence on the resulting colour shade. In other words, for grey tones it is difficult to find the dominant wavelenght.
- If the color is not in grayscale range then:
  - If xy coordinates of the color lie inside the triangle limited by the two green dashed lines and the magenta line, then the dominant wavelength for that color does not exist and the color is assumed to have two different wavelength values:λ_{red} = 612 nm (the dominant wavelength for sRGB red color) and λ_{blue} = 449 nm (the dominant wavelength for sRGB blue color). These are the two closest colours which have the dominant length.
  - Otherwise the dominant wavelength for that color is calculated as described above.

3) Calculate Primary Colors Values of the Output Color Filter: In this step, the camera spectral sensitivity functions is used for the purpose of primary colors values extraction. In case of the RYYCy color filter three different functions exist, one for each primary color (red, yellow and cyan). In this case, there are also three different cases of how to calculate primary colors values.

When the input color (sRGB triple) is classified as grayscale, then spectral sensitivity functions cannot be used, since no wavelength can be assigned to that color and each channel of the output color filter has the following value:

$$C = Y \tag{18}$$

where C is the value of the given primary color of the output color filter and Y is the luminancy corresponding to the input color.

If the dominant wavelength for the input color exists, then then the value of each primary color can be calculated as follows:

$$C = css(\lambda)Y \tag{19}$$

where $css(\cdot)$ is spectral sensitivity function corresponding to that color and λ is dominant wavelength value calculated for the input color.

If the input color does not have its dominant wavelength, then each primary color value is calculated as a weighted sum of the two spectral sensitivity functions values calculated from sRGB red wavelength value (λ_{red}) and from sRGB blue wavelength value (λ_{blue}).

$$C = \left(css(\lambda_{red})w_{red} + css(\lambda_{blue})w_{blue}\right)Y \tag{20}$$

where weights w_{red} and w_{blue} are calculated from the sigmoid functions. This allows to mix colours and achieve the desired shade of colour using the two closest colours that have the dominant wavelength.

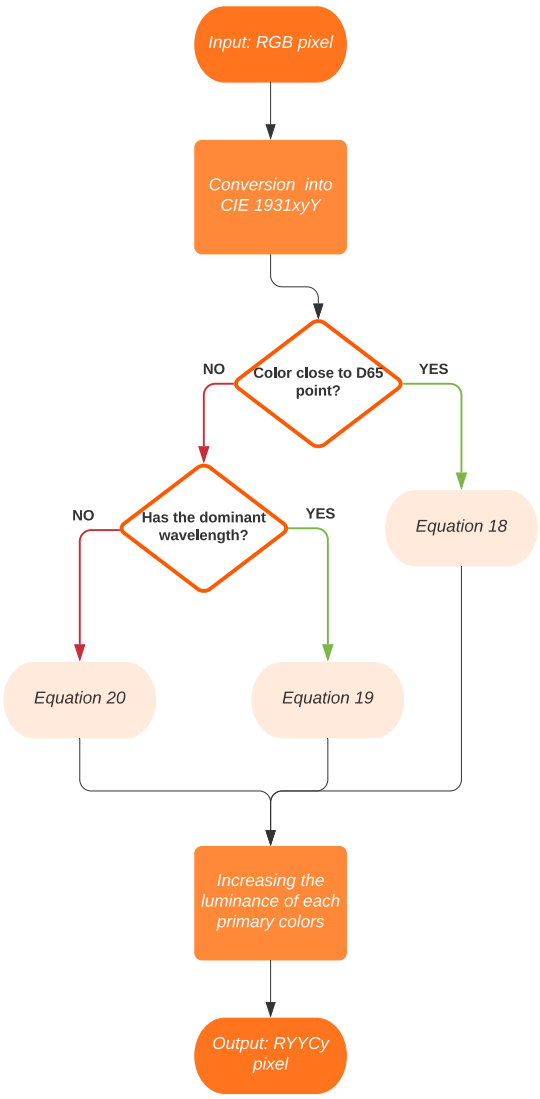


Fig. 7. Analytical model flowchart.

4) Increasing the Luminance of Each Primary Color: In this step the luminance of the primary colors that were calculated for the non-grayscale input colors is increased with the gain parameter. The values of gain parameters are chosen separately for each primary color in the process of optimization that is minimizing the error between the image in virtual simulation and the image from real camera in which the custom color filter is used. Therefore, the gain values highly depend on the type of camera that is simulated. Figure 7 present flowchart of the algorithm.

B. Data Sets

The data set consists of about 80 pictures of two color charts under different lighting conditions. Each color chart consists of 100 different colors, 10 for each row. To change lighting conditions RGBW diod lamp was used as the only light source. The number of images taken was the result of trying to get the best possible coverage for the RGB data. Both RGB and RYYCy cameras were placed next to each other, as close as possible, that the light reflected from the colour chart falls

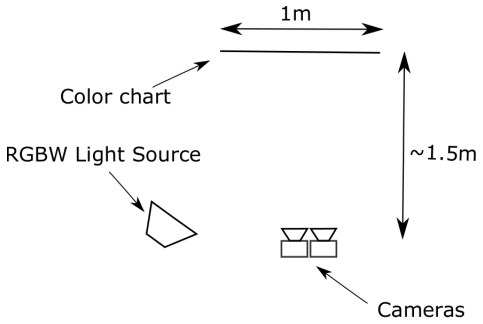


Fig. 8. Experimental setup.

at the closest possible angle to the both cameras, in order to minimise differences in the perception of light. The light source has been positioned at such a distance that uniform illumination is achieved on the colour chart. The distance from cameras to the testchart was approximately $1.5m$, to ensure that the board is fully visible to both cameras and cameras do not occlude the light source.

Figure 8 presents an experimental setup. From each color in the chart a 5 by 5 pixel square was taken. Thus, each color is represented by 25 samples. This allows for the estimation of the noise present in the data, assuming that the surroundings of a pixel on a flat colour should be homogeneous. In order to avoid any non-linearities related to the image processing, the data read directly from the 12-bit imager were used. Data were normalised to values from zero to one, no other data processing steps were performed. Gathered data are visualised in Figures 11, 12 and 13.

C. Neural Network Black-Box Approach - RGB2RYYCyNN

Based on the deliberations presented in II, it is assumed that a function which can transform one color space can exist in the form of:

$$\mathbf{c}_1 = g(\mathbf{c}_2) \tag{21}$$

Large gathered data set and the assumption of function continuity, make it possible to use neural networks to solve this problem. The effectiveness of the use of neural networks in the approximation of functions has been comprehensive justified over the last three decades [38]. Works on universal approximations theorem [39], [40] indicate that a continuous function defined on bounded domain can be approximated by a large two-layer neural network. Recently, there has been interest in understanding the approximation capabilities of deep networks [41]. The theoretical benefits of using deeper neural network for specific functions approximation were demonstrated in [42]. The development of the neural network was based on Residual neural network (ResNet). The core idea of ResNet is "identity shortcut connection" that skips one or more layers. Skipping over layers is motivated by the possibility of avoiding the problem related to vanishing gradients. The author's [43] argue that the use of more layers should not degrade network performance because the use of

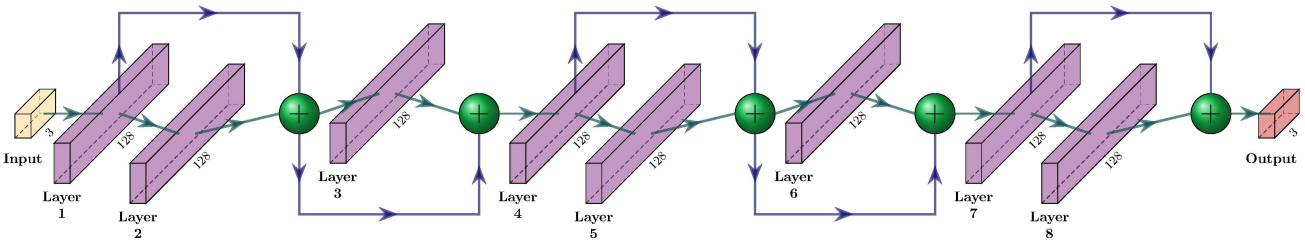


Fig. 9. Neural network architecture.

identity mapping upon the current network, and the resulting architecture would perform the similarly.

The proposed neural network architecture is a dense feed-forward neural network $f(x, \theta)$ (Figure 9). The size and dimensions of the network were selected empirically on the basis of studies carried out. The best results were obtained with 9-layer networks. A larger number of layers caused a significant overfitting of the network. A smaller number of layers did not allow to obtain good results using additional lottery ticket approach. The first 8 layers are regular densely-connected neural network layers. Additionally, each layer is performing batch normalization. The activation function is leaky rectified linear unit (Leaky ReLu):

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ 0.25x & \text{otherwise.} \end{cases} \tag{22}$$

In practice, networks with ReLu functions tend to show better convergence performance than sigmoid functions and resolves issues with vanishing gradient [44]. The last layer differs in the activation function, which is sigmoid function. The loss function was defined as L_1-norm.

To overcome overfitting problem two approaches were used. In the first one, together with samples from data set, an artificially created image was processed by the network. The input image is shown in Figure 10a. The example of image processed by neural network is shown in Figure 10a. For each batch the gradient of output image was computed. The image gradient was defined as a vector of its partials [45]:

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{23}$$

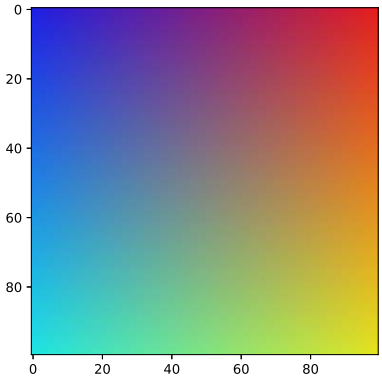
where $\frac{\partial f}{\partial x}$ is the derivative with respect to x and $\frac{\partial f}{\partial y}$ is the derivative with respect to y. These were calculated using 1-dimensional convolution operation $*$:

$$\frac{\partial f}{\partial x} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \mathbf{A} \tag{24}$$

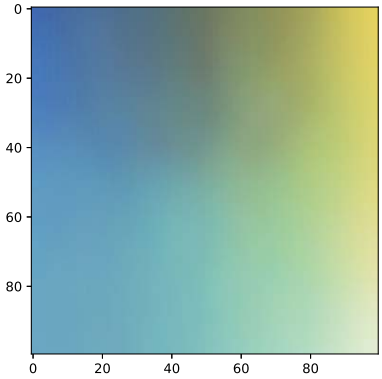
$$\frac{\partial f}{\partial y} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} * \mathbf{A} \tag{25}$$

Subsequently, for each channel the sum of gradient magnitude were calculated:

$$g_i = \sum_{k=1}^{H} \sum_{j=0}^{W} \sqrt{g_x(k, j)^2 + g_y(k, j)^2} \quad \text{for } i = 1, 2, 3 \tag{26}$$



(a) Input image



(b) Output image

Fig. 10. Images used for regularization.

where H and W are image height and width respectively. On this basis, a following component was added to the cost function in order to achieve smoothness of color transformation:

$$f = K \min\{0, \left(\sum_{i=1}^{3} g_i\right) - th\} \tag{27}$$

where th was estimated from the sum of gradient of input image and K was arbitrary parameter adjusted during training.

Also, the lottery ticket pruning procedure was used to prevent overfitting and reach higher test accuracy. The results presented in [46] show that for fully connected feed-forward networks winning lottery ticket can achieve better results on test dataset. Densely connected neural networks contain

approximately 118000 trainable parameters at the start of learning. During the lottery ticket identification only weights from biases and dense layer kernels were pruned. Weights associated with batch normalization layers were not pruned. Thus, approximately around 116500 parameters were taken into account in the lottery ticket procedure. Identification of the winning ticket was made by training the network and pruning its smallest-magnitude weights. The incorporated pruning scheme was as follows:

1) Create an empty mask m_0 and randomly initialize a neural network $f(x, \theta_0)$.
2) Train the network for k iterations, yielding parameters θ_j.
3) Prune $p\%$ of smallest-magnitude the parameters in $m_{i-1} \odot \theta_0$, creating a mask m_i.
4) Reset the remaining parameters to their values in θ_0.
5) Repeat points 2-4 i-times, creating the final mask m and the winning ticket $f(x, m \odot \theta_0)$.
6) Continue standard approach training of the winning ticket neural network $f(x, m \odot \theta_0)$.

The final network was trimmed 12 times, each time 15% of remaining weights were pruned. This process leads to only 14% of initial dense layer kernels and biases weights have magnitude different from zero. More aggressive pruning during a single iteration resulted in worse results. A less aggressive approach did not eliminate the problem of overfitting. Increasing the number of iterations caused the network to become too small, resulting in a underfitting problem.

D. Polynomial Model - RGB2RYYCyPoly

According to equation (7) for a given spectral distribution conversion from $\mathbf{c}_1$ to $\mathbf{c}_2$ is linear. Thus, each channel is represented as linear combination of color triplets, which can be presented as follows:

$$y_i = \sum_{j=1}^{3} \alpha_{ij} x_j \quad \text{for } i = 1, 2, 3 \tag{28}$$

However, for different spectral distributions and therefore also for different x the coefficients a is distinct. Nevertheless, in line with the considerations presented in section II the conversion function should be continuous. Consequently, the following modification of equation (28) was proposed:

$$y_i = \min\left\{1,\ y_{i0} + \sum_{j=1}^{3} \gamma_{ij} x_j^{\beta_{ij}}\right\} \quad \text{for } i = 1, 2, 3 \tag{29}$$

Furthermore, account has been taken of the saturation that occurs in the actual data. The purpose of these modifications is to improve model matching to gathered data during optimization.

From the relationship between RYYCy and RGB filters, one can conclude that yellow channel (Y) should depend mostly on red and green channel from RGB filters. Similarly, cyan in the real world can arise from a mixture of blue and green. Additionally, the conversion of red channel includes a possible influence of green filter. This leads to the modification of equation (29) for color triplets:

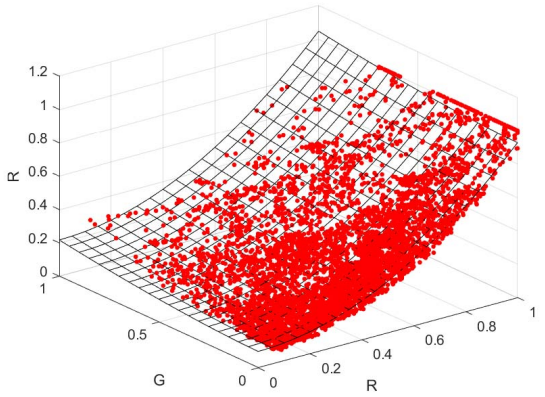


Fig. 11. Red surface.

$$y_1(x_1,\ x_2) = \min\left\{1,\ y_{10} + \gamma_{11} x_1^{\beta_{11}} + \gamma_{12} x_2^{\beta_{12}}\right\} \tag{30}$$

$$y_2(x_1,\ x_2) = \min\left\{1,\ y_{20} + \gamma_{21} x_1^{\beta_{21}} + \gamma_{22} x_2^{\beta_{22}}\right\} \tag{31}$$

$$y_3(x_2,\ x_3) = \min\left\{1,\ y_{20} + \gamma_{32} x_2^{\beta_{32}} + \gamma_{33} x_3^{\beta_{33}}\right\} \tag{32}$$

Reducing the size of the problem should allow for better generalization of the model. It also helps to mitigate the problem of overfitting to data noise. Furthermore, the idea of separating variables for individual channels allows for data visualization.

Figure 11 represents the dependence of the R-channel data in the RYYCy color filter array depending on the R-channel and G-channel data in the RGB filter. Moreover, the figure also shows the plane of the best fit for function 30. All plane functions (30-32) were determined by numerical optimization. The loss function was defined separately for each conversion as follows:

$$loss_i = \sum_{n=1}^{N} |y_i - d_{n,i}| \quad \text{for } i = 1, 2, 3 \tag{33}$$

where:

- N is number of data samples.
- $d_{n,i}$ are RYYCy data for each channel.
- y_i is one of the functions (30-32).

Analysing the graph, the polynomial model allows for good approximation for the red channel. This is mainly related to the fact that one red filter is transformed into another red filter with different sensitivity. Nevertheless, the graph shows a slight influence of the green channel.

In the case of yellow channels, it is a mixture of red and green channels what is presented in Figure 12. Similarly, for cyan channel, which is dependent on data from blue and green filters. The flattening visible in both graphs 12 and 13 is due to the fact that the RYYCy filter transmits more light than the RGB filter. This causes that for certain conditions, the images from the RYYCy camera are overexposed. For the collected data, there is no noticeable transformation ambiguity effect resulting from the theoretical considerations presented in section II. In particular, there is no visible effect of two significantly different RYYCy colors corresponding to one

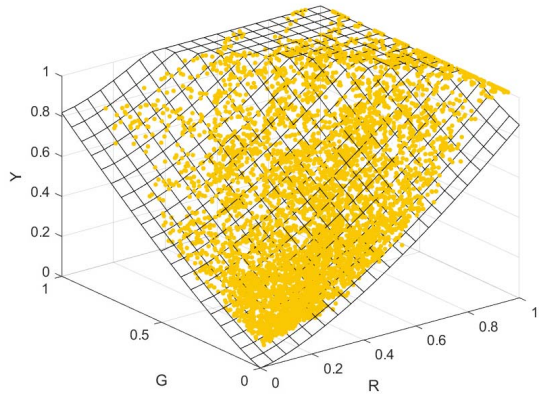


Fig. 12. Yellow surface.

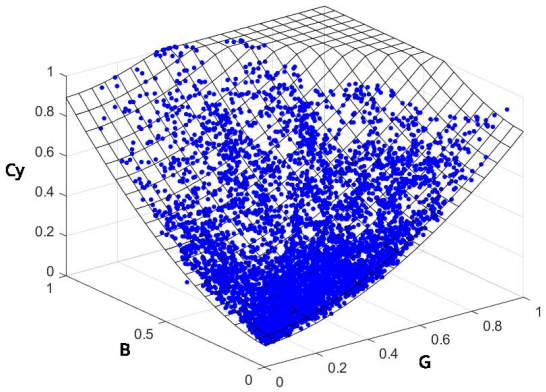


Fig. 13. Cyan surface.

RGB color. Thus, the approximation of color conversion using polynomial functions is justified.

E. RGB2RYBbyGC

Gosset and Chen [29] claimed that definition of a rigorous mathematical conversion from RGB to RYB would be difficult. Instead, they stated that a reasonable approximation my be obtained by defining a cube with each axis representing either Red, Green or Blue. By defining appropriate RYB values for each of the eight colors represented by the corners of the cube the trilinear interpolation can be used to obtain suitable RYB values for any colors defined in RGB. In our case, the conversion is from RGB to RYYCy. Thus, one need to define the RYYCy values for the corners of the cube. To achieve this, numerical optimization was used. The RYYCy values for the corners of the cube were optimized such that the trilinear interpolation minimizes the error defined by the objective function 33. Additionally, the values from trilinear approximation are saturated to be in range from 0 to 1. Obtained interpolation cube is shown in Figure 14.

F. RGB2RYBbyST

Model presented in [31] transforms RGB addtive color model to the RYB subtractive color model. The model was modified to transform to a color space that is additive. Thus, the RYB space resembles RYYCy. The resulting model

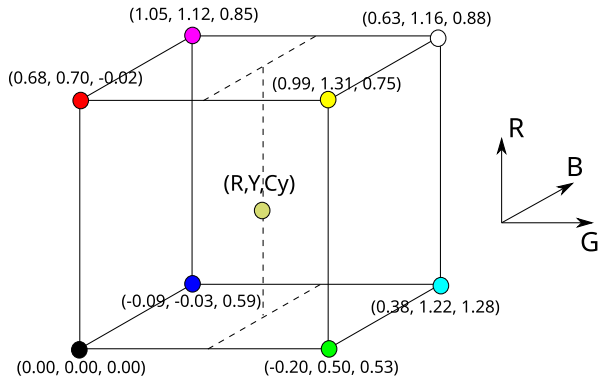


Fig. 14. RGB interpolation cube. For each corner of the RGB cube, RYYCy coordinates are established using numerical optimization. The trilinear interpolation results in suitable RYYCy values for any RGB color defined inside the cube.

equations are as follows:

$$y'_1 = x_1 - \min\{x_1, x_2\} \tag{34}$$

$$y'_2 = \frac{x_2 + \min\{x_1, x_2\}}{2} \tag{35}$$

$$y'_3 = \frac{x_2 + x_3 - \min\{x_1, x_2\}}{2} \tag{36}$$

$$\tag{37}$$

Equation normalization:

$$y_i = \frac{y'_i}{n} \quad for\ i = 1, 2, 3 \tag{38}$$

Here, n is calculated as follows:

$$n = \frac{\max\{x_1, x_2, x_3\}}{\max\{y'_1, y'_2, y'_3\}} \tag{39}$$

where:

- x_i are respectively R, G, B
- y_i are respectively R, Y, B

G. Limitations

All methods presented are pixel based. They do not use contextual information that may be present in the image. It can lead i.e. to odd edge discolouration appearing on the image. Furthermore, RGB2RYYCyAna and modified RGB2RYBbyST are not data-driven, which means that effects that are not modelled by these colour transformation algorithms are likely to cause larger errors.

IV. Discussion

The discussion focuses on comparing presented models.. The considerations take into account key quality indicators and execution times of the algorithms.

A. KPIs

The following two functions were used to calculate the quality indicators:

$$J_1 = \sum_{n=1}^{N} \sum_{i=1}^{3} |y_{n,i} - d_{n,i}| \tag{40}$$

$$J_2 = \frac{\sqrt{3}}{3} \sum_{n=1}^{N} \sqrt{\sum_{i=1}^{3} (y_{n,i} - d_{n,i})} \tag{41}$$

TABLE I
NORMALIZED L2-NORM

	mean	median	95th percentile
RGB2RYYCyAna	0.103	0.081	0.236
RGB2RYYCyNN	0.023	0.018	0.057
RGB2RYYCyPoly	0.046	0.037	0.110
Modified RGB2RYBbyGC	0.050	0.042	0.108
Modified RGB2RYBbyST	0.187	0.143	0.381

where:

- N is number of test data samples.
- $d_{n,i}$ are RYYCy data for each channel.
- $y_{n,i}$ are RYYCy data obtained from models.

The first metric is a mean absolute difference for all channels. The second one is using L2-norm to calculate distance and is also normalized to give output from range 0 to 1.

B. Results

Results for KPI J_2 are presented in Table I. The best results were obtained for the Neural Network model. The error rate was approximately 2.3%. In the case of RGB2RYYCyPoly and modified RGB2RYBbyGC the obtained results were two times worse. Nevertheless, the visual differences between captured RGB images converted to RYYCy using these methods were hardly noticeable. Comparing the presented performance indicators for the polynomial model and the neural network one can see that in both cases the error had a very similar distribution with a heavy tail. The significant difference between the median and the mean value and the relatively large 95th percentile value that leads to heavy tail distributions was most likely the result of presence of a significant amount of noise in the data. The worst results were obtained for the RGB2RYYCyAna and modified RGB2RYBbyST. There were large discrepancies between theoretical assumptions and the actual analytical model performance. The use of quantum efficiency data describing filters characteristics and assumption made about dominant length estimation did not allow for good results to be obtained. The rigidly defined equations in the case of the modified RGB2RYBbyST model do not correctly represent the transformation either.

Table II present results for J_1 KPI. As in the previous case, the best results were obtained for the neural network. The existence of large errors in the RGB2RYYCyAna and Modified RGB2RYBbyST makes it inapplicable for practical use. In the case of a RGB2RYYCyPoly, RGB2RYYCyNN the best results were obtained for the red channel. They were about 15% better in both cases compared to the best of the other channels. This was mainly due to the presence of the red filter in both color filter arrays. However, the errors were not so significantly different from the other filters. This leads to the conclusion that either the shape of the red filters was noticeably different or there was significant noise in the data. The RGB2RYYCyPoly gives better results for red and blue filters than the modified RGB2RYBbyGC model. For the yellow filter, RGB2RYBbyGC peforms slightly better than RGB2RYYCyPoly. However, it should be noted that model RGB2RYYCyPoly uses only 15 parameters while model RGB2RYBbyGC uses 24.

TABLE II
ABSOLUTE DIFFERENCE

		mean	median	95th percentile
RGB2RYYCyAna	red	0.090	0.085	0.228
	yellow	0.110	0.089	0.253
	cyan	0.094	0.073	0.211
RGB2RYYCyNN	red	0.018	0.013	0.054
	yellow	0.021	0.016	0.060
	cyan	0.021	0.014	0.060
RGB2RYYCyPoly	red	0.030	0.022	0.085
	yellow	0.052	0.040	0.151
	cyan	0.036	0.024	0.112
Mod. RGB2RYBbyGC	red	0.045	0.038	0.117
	yellow	0.047	0.039	0.114
	cyan	0.042	0.033	0.113
Mod. RGB2RYBbyST	red	0.197	0.016	0.420
	yellow	0.142	0.080	0.171
	cyan	0.151	0.117	0.368

TABLE III
NOISE

		mean	median	95th percentile
RGB	red	0.011	0.009	0.035
	green	0.010	0.008	0.028
	blue	0.009	0.007	0.025
RYYCy	red	0.007	0.006	0.018
	yellow	0.008	0.006	0.019
	cyan	0.008	0.006	0.020
RGB2RYYCyAna	red	0.011	0.007	0.034
	yellow	0.008	0.006	0.023
	cyan	0.009	0.007	0.026
RGB2RYYCyNN	red	0.090	0.007	0.028
	yellow	0.011	0.008	0.031
	cyan	0.008	0.006	0.023
RGB2RYYCyPoly	red	0.009	0.007	0.026
	yellow	0.013	0.010	0.034
	cyan	0.008	0.006	0.021
Mod. RGB2RYBbyGC	red	0.014	0.013	0.024
	yellow	0.016	0.016	0.026
	cyan	0.011	0.012	0.022
Mod. RGB2RYBbyST	red	0.015	0.007	0.061
	yellow	0.017	0.011	0.058
	cyan	0.012	0.009	0.034

Noise sources in the raw data from imager are mainly due to the existence of dark current. In order to estimate the level of noise in the images, the collected data presented in section III-B was used. Each colour is represented by 25 samples. The reference value from which the noise was calculated was the average of these samples. In order to be comparable with the quality indicators presented above, the noise estimation was performed using the previously used function J_1 (40). Table III shows noise estimate for the data used for models optimization. An estimate of noise in the data that was processed by the models is also presented. The noise levels for both RGB and RYYCy are about twice as low as the errors obtained for the best model. However, it should be noted that the noise in these data affected the error of the methods simultaneously from two sides. Together with input data the noise propagated to the output of the models where it was compared to data that also had noise. As it can be seen, all the models presented did not amplify the amount of noise in the output data. By comparing the indicators presented, it can be concluded that noise retained its distribution after passing through the models.

TABLE IV
ALGORITHMS RUNNING TIME

Lookup table source	Time [ms]		
	mean	standard deviation	95th percentile
RGB2RYYCyAna	5.028	0.174	5.215
RGB2RYYCyNN	4.965	0.154	5.143
RGB2RYYCyPoly	5.033	0.156	5.226
Mod. RGB2RYBbyGC	5.001	0.170	5.185
Mod. RGB2RYBbyST	4.983	0.161	5.210

Based on the observations, one can try to estimate that the maximum effect of noise on the error as the sum of the noise present in the RGB and RYYCy data. Thus, the contribution of noise to the error of the methods can be estimated up to a maximum of 2%. Consequently, the data obtained by the use of a neural network allows for a very accurate representation of the transformation because their error is only slightly bigger than noise power. The other source of error may be due to the existence of multiple solutions for a single colour as shown in section II. This is evidenced primarily by the large values for the 95th percentile for the neural network and polynomial model presented in Tables II and I.

Since the data set was chosen to represent the whole space, it can be concluded that the compared methods do not amplify noise in the data. On the basis of the experimental data one can find that the partial derivatives of the presented colour space transform functions are limited. In practical applications, the data to be transformed will come from a virtual simulation and will contain no noise. Consequently, the property of not amplifying noise is not so important for the colour transformation model.

C. Algorithms Running Time

In computer memory each pixel is represented by a finite number of bits. Thus, the colour conversion problem is discrete and finite which allows to present above algorithms in the form of lookup table. Assuming 24-bit color depth an implementation in CUDA leverages of lookup tables technique was prepared. For each algorithm lookup tables of the precisely same size were generated. During the test fifty images with resolution 3840×2160 were processed by algorithm implemented in CUDA on Nvidia RTX2080Ti. Algorithms conversion time are presented in Table IV.

No differences were observed between the tables created by the different algorithms due to the fact that the whole time complexity of the algorithm depends on memory access, not on calculations. Experiments show that the algorithm allows to process images with resolution 3840×2160 with frame rate at up to 200Hz.

V. SUMMARY

The publication presents a theoretical analysis of the color transformation from one color space to another. The analysis showed that in general case the task did not have a solution, but there may be methods to obtain a satisfactory approximation of the color transformation.

RGB2RYBAna and modified RGB2RYBbyST approaches obtained the worst quality indicators of all. Additionally for RGB2RYBAna model additional knowledge of camera spectral sensitivity for specific filters is required to apply these method. It is not possible to obtain results comparable to the other models without additional optimization of the model hyperparameters in order to account for external influences such as lens effects and color filter imperfections.

In the case of the RGB2RYYCyPoly, RGB2RYYCyNN and modified RGB2RYBbyGC better results were obtained. Polynomial and model RGB2RYBbyGC allowed for a significant reduction of parameters compared to the neural network and guaranteed smoothness of the solution, which was a big problem in the black box approach. In addition, for model RGB2RYYCyPoly the relationship between RGB and RYYCy filters was taken into account, which reduced the number of parameters. Model RGB2RYYCyPoly uses 42% less parameters than model RGB2RYBbyGC.

The actual implementation of the transformation is done form a finite discrete space to another finite discrete space. Thus, the look-up table approach can be used to accelerate computation speed for all presented methods.

REFERENCES

[1] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *J. Field Robot.*, vol. 37, no. 3, pp. 362–386, Apr. 2020.
[2] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3D object detection and tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 11784–11793.
[3] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12689–12697.
[4] H.-W. Huang, C.-R. Lee, and H.-P. Lin, "Nighttime vehicle detection and tracking base on spatiotemporal analysis using RCCC sensor," in *Proc. IEEE 9th Int. Conf. Humanoid, Nanotechnol., Inf. Technol., Commun. Control, Environ. Manage. (HNICEM)*, Dec. 2017, pp. 1–5.
[5] B. Kiran *et al.*, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, early access, Feb. 9, 2021, doi: 10.1109/TITS.2021.3054625.
[6] L. Liu *et al.*, "Computing systems for autonomous driving: State of the art and challenges," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6469–6486, Dec. 2020.
[7] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transp. Res. A, Policy Pract.*, vol. 94, pp. 182–193, Dec. 2016.
[8] R. F. Lyon, *The Optical Mouse: Early Biomimetic Embedded Vision*. London, U.K.: Springer, 2014, pp. 3–22.
[9] E. R. Fossum and D. B. Hondongwa, "A review of the pinned photodiode for CCD and CMOS image sensors," *IEEE J. Electron Devices Soc.*, vol. 2, no. 3, pp. 33–43, May 2014.
[10] N. Junichi, *Image Sensors and Signal Processing for Digital Still Cameras*. Boca Raton, FL, USA: CRC Press, 2005.
[11] Z. Liu, T. Lian, J. Farrell, and B. A. Wandell, "Neural network generalization: The impact of camera parameters," *IEEE Access*, vol. 8, pp. 10443–10454, 2020.
[12] K. Weikl, D. Schroeder, and W. Stechele, "Optimization of automotive color filter arrays for traffic light color separation," in *Proc. Color Imag. Conf.*, 2020, pp. 288–292.
[13] P. Pawłowski, K. Piniarski, and A. Dąbrowski, "Highly efficient lossless coding for high dynamic range red, clear, clear, clear image sensors," *Sensors*, vol. 21, no. 2, p. 653, Jan. 2021.
[14] G. Karanam. *Interfacing Red/Clear Sensors to ADSP-BF609® Blackfin Processors*. Analog Devices, Inc. Accessed: Jul. 15, 2021. [Online]. Available: https://www.analog.com/media/en/technical-documentation/application-notes/ee358.pdf
[15] R. Jenkin and P. Kanel, "Fundamental imaging system analysis for autonomous vehicles," *Electron. Imag.*, vol. 2018, pp. 1–10, Jan. 2018.
[16] H.-P. Lin, P.-H. Liao, and Y.-L. Chang, "Long-distance vehicle detection algorithm at night for driving assistance," in *Proc. 3rd IEEE Int. Conf. Intell. Transp. Eng. (ICITE)*, Sep. 2018, pp. 296–300.

[17] O. Eytan and E. Belman, "High-resolution automotive lens and sensor," U.S. Patent 2019/0 377 110 A1, Dec. 12, 2019.
[18] T. Sulkowski, P. Bugiel, and J. Izydorczyk, "In search of the ultimate autonomous driving simulator," in *Proc. Int. Conf. Signals Electron. Syst. (ICSES)*, Sep. 2018, pp. 252–256.
[19] dSPACE. *ASM Traffic*. Accessed: Jun. 15, 2021. [Online]. Available: https://www.dspace.com/en/pub/home.cfm
[20] PGAutomotive. *CarMaker*. Accessed: Jun. 15, 2021. [Online]. Available: https://ipg-automotive.com/products-services/simulation-software/carmaker/
[21] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 1–16.
[22] N. Hirsenkorn, T. Hanke, A. Rauch, B. Dehlink, R. H. Rasshofer, and E. Biebl, "Virtual sensor models for real-time applications," *Adv. Radio Sci.*, vol. 14, pp. 31–37, Sep. 2016.
[23] M. Jasinski, "A generic validation scheme for real-time capable automotive radar sensor models integrated into an autonomous driving simulator," in *Proc. 24th Int. Conf. Methods Models Autom. Robot. (MMAR)*, Aug. 2019, pp. 612–617.
[24] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann, 2016.
[25] A. L. Lin, "He computational image systems evaluation toolbox," Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Stanford, CA, USA, 2015.
[26] H. Blasinski, J. Farrell, T. Lian, Z. Liu, and B. Wandell, "Optimizing image acquisition systems for autonomous driving," *Electron. Imag.*, vol. 30, no. 5, pp. 161-1–161-7, Jan. 2018.
[27] Z. Liu *et al.*, "A system for generating complex physically accurate sensor images for automotive applications, *Electron. Imag.*, vol. 2019, no. 15, pp. 1–5, Jan. 2019.
[28] S. Schmidt, B. Schlager, S. Muckenhuber, and R. Stark, "Configurable sensor model architecture for the development of automated driving systems," *Sensors*, vol. 21, no. 14, p. 4687, 2021.
[29] N. Gossett and B. Chen, "Paint inspired color mixing and compositing for visualization," in *Proc. IEEE Symp. Inf. Vis.*, Oct. 2004, pp. 113–118.
[30] J. Itten, *The Art of Color: The Subjective Experience and Objective Rationale of Color*. New York, NY, USA: Van Nostrand Reinhold, 1973.
[31] J. Sugita and T. Takahashi, "Paint-like compositing based on RYB color model," in *Proc. ACM SIGGRAPH Posters*, New York, NY, USA, 2015, p. 1.
[32] M. Piatek and M. Jasinski, "Method for simulating a digital imaging device," Patent EP3 709 623 A1, Sep. 16, 2020.
[33] Z. Sadeghipoor, Y. M. Lu, and S. Süsstrunk, "Optimum spectral sensitivity functions for single sensor color imaging," *Proc. SPIE*, vol. 8299, pp. 26–39, Jan. 2012.
[34] R. Penrose, "A generalized inverse for matrices," *Math. Proc. Cambridge Phil. Soc.*, vol. 51, no. 3, pp. 406–413, 1955.
[35] A. D. Broadbent, "A critical review of the development of the CIE1931 RGB color-matching functions," *Color Res. Appl.*, vol. 29, no. 4, pp. 267–272, 2004.
[36] *Multimedia Systems and Equipment—Colour Measurement and Management—Part 2–1: Colour Management—Default RGB Colour Space—SRGB*, IEC Central Secretary, International Electrotechnical Commission, Geneva, CH, Standard IEC 61966-2-1:1999, 1999.
[37] E. C. Carter, Y. Ohno, M. R. Pointer, A. R. Robertson, R. Seve, J. D. Schanda, K. Witt, "Colorimetry," Int. Commission Illumination, Vienna, Austria, Tech. Rep. CIE 15:2004, 2004. [Online]. Available: https://www.cdvplus.cz/file/3-publikace-cie15-2004/
[38] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Syst.*, vol. 2, no. 4, pp. 303–314, Dec. 1989.
[39] J. Park and I. W. Sandberg, "Approximation and radial-basis-function networks," *Neural Comput.*, vol. 5, no. 2, pp. 305–316, Mar. 1993.
[40] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Trans. Inf. Theory*, vol. 39, no. 3, pp. 930–945, May 1993.
[41] S. Liang and R. Srikant, "Why deep neural networks for function approximation?" in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–17.
[42] M. Telgarsky, "Representation benefits of deep feedforward networks," 2015, *arXiv:1509.08101*.
[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[44] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Ed. Red Hook, NY, USA: Curran Associates, 2012.
[45] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2006.
[46] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," 2019, *arXiv:1803.03635*.

Kamil Lelowicz (Member, IEEE) received the master's degree in automatic control and robotics. He is pursuing the Ph.D. degree with the AGH University of Science and Technology, Cracow, Poland. He is also engaged in commercial research and development and product development activities of advanced automotive driver assistance system and cabin monitoring system. His research interests include modeling and simulation, computer vision, and machine learning. Mr. Lelowicz is a member of Polish Chapter of the IEEE Vehicular Technology Society.

Michał Jasiński (Member, IEEE) received the master's degree in automatic control and robotics. He is pursuing the Ph.D. degree with the AGH University of Science and Technology, Cracow, Poland. He works in cooperation with automotive industry being involved in research and development projects for advanced safety systems. His research is mostly focused on sensor modeling, virtual validation, and machine learning. Mr. Jasiński is a member of a Polish Chapter of the IEEE Vehicular Technology Society.

Adam Krzysztof Piłat (Member, IEEE) received the M.Sc. degree in 1996, the Ph.D. degree in 2002, and the D.Sc. degree in 2014. He is a Professor with the AGH University of Science and Technology. He is currently working with the Department of Automatic Control and Robotics as a Researcher, a Didactic, and the Head of the Photovoltaic, Robotics and Magnetic Levitation Laboratory. He is the author and coauthor of 42 articles, 85 papers, four books, 12 book chapters, and 20 patents. He is a member of IEEE Control Systems Society and Robotics and Automation Society. His scientific interest are focused on interdisciplinary modeling, simulation, controller synthesis, and real-time control. His research are mainly focused on devices design and prototyping with embedded active magnetic levitation technology.