

# Fast and Robust Multiple ColorChecker Detection using Deep Convolutional Neural Networks

Pedro D. Marrero Fernandez, Fidel A. Guerrero-Peña, Tsang Ing Ren, *Member, IEEE*, and Jorge J. G. Leandro

**Abstract**—ColorCheckers are reference standards that professional photographers and filmmakers use to ensure predictable results under every lighting condition. The objective of this work is to propose a new fast and robust method for automatic ColorChecker detection. The process is divided into two steps: (1) ColorCheckers localization and (2) ColorChecker patches recognition. For the ColorChecker localization, we trained a detection convolutional neural network using synthetic images. The synthetic images are created with the 3D models of the ColorChecker and different background images. The output of the neural networks are the bounding box of each possible ColorChecker candidates in the input image. Each bounding box defines a cropped image which is evaluated by a recognition system, and each image is canonized with regards to color and dimensions. Subsequently, all possible color patches are extracted and grouped with respect to the center's distance. Each group is evaluated as a candidate for a ColorChecker part, and its position in the scene is estimated. Finally, a cost function is applied to evaluate the accuracy of the estimation. The method is tested using real and synthetic images. The proposed method is fast, robust to overlaps and invariant to affine projections. The algorithm also performs well in case of multiple ColorCheckers detection.

**Index Terms**—ColorChecker Detection, Photograph, Image Quality, Color Science, Color Balance, Segmentation, Convolutional Neural Network.

## I. INTRODUCTION

The illumination of a scene highly influences the reproduction of the colors in images captured with digital cameras. Given a camera sensor with a Spectral Sensitivity, color renderings can deviate significantly from colors perceived by human eyes depending on the Color Stimulus, that is, the product between the Spectral Power Distribution of the incoming light and the Spectral Reflectance of the object. However, accurate color reproduction still requires colorimetric camera calibration for different illuminations that is usually done using a ColorChecker (CC) that shows predefined regions with specified colors. Manufacturers offer various checker models for specific applications [1].

ColorChecker Targets are reference standards that professional photographers and filmmakers use to ensure predictable results under every lighting condition. The use of the CC speeds up the color adjustment in the process of obtaining accurate colors. Therefore, minimizing tedious work of trial and error color adjustments while editing or color grading. In



Fig. 1. Examples of the X-Rite ColorCheckers: a) ColorChecker® Classic; b) ColorChecker® Digital SG.

order to scale the assessment of the color accuracy and auto white balance for a large number of images, it is necessary to automate this process, thus automating CC detection is also required.

Currently, there are several published papers in the literature on geometric camera calibration. However, only a few publications on the detection of ColorChecker in images. Many types of ColorCheckers exist, each one specifically designed for a different class of device and for the property to assess. Fig. 1 shows two of the most used ColorChecker's: X-Rite ColorChecker® Classic (CCC) and X-Rite ColorChecker® Digital SG (CSG). The CCC is an  $8 \times 11$  inch chart which consists of 24 patches with 18 familiar colors and six grayscale levels having optical densities from 0.05 to 1.50 and a range of 4.8 f-stops. The colors are not highly saturated, and the ColorChecker quality is very high. Each patch is printed separately using controlled pigments, and the patches have a smooth matte surface. The CCC is a standard color target. The exact description of these ColorCheckers can be found on the X-Rite website<sup>1</sup>.

Tajbakhsh and Grigat proposed an algorithm for semi-automatic CCC detection in distorted images [2]. Initially, the user selects four corners in the ColorChecker image, and the system estimates the position of all color patches using projective geometry. The image is processed with a Sobel kernel, a morphological operator, and a threshold is applied converting it into a binary image and the connected patches are found.

Kapusi et al. proposed a method combining geometric and colorimetric camera calibration with a unified calibration checker [3]. A black and white chessboard pattern is used for the geometric calibration, and 24 circular reference color patches are placed in squares center for the color calibration. The chessboard pattern detection is obtained using the OpenCV library [4]. A subsequent region growing algorithm

Pedro D. Marrero Fernandez, Fidel A. Guerrero-Peña and Tsang Ing Ren are with Centro de Informática, Universidade Federal de Pernambuco, Recife, Brazil (e-mail: pdmf@cin.ufpe.br; fagp@cin.ufpe.br; tir@cin.ufpe.br).

Jorge J. G. Leandro is with Motorola Mobility, LLC, Brazil (e-mail: jleandro@motorola.com)

<sup>1</sup><http://xritephoto.com/colorchecker-targets>

segments the circular color regions [3].

Bianco et al. [5] presented a method for single CCC detection. The method locates the ColorChecker by finding the coordinates on the local color descriptors spaces using the SIFT descriptors and recognizes the ColorChecker using an optimization function.

Ernst et al. proposed a robust algorithm to detect and track color calibration checkers in images [1]. The automatic CCC detection procedure uses a cost function to find the checker in an image. The cost function compares the colors of the patches with the reference colors and the standard deviations of the colors within the patch regions. Four corners of the checker model are projected with the use of the Direct Linear Transformation to find the coordinates in an image. The coordinates of the CCC are obtained by minimizing the cost function using the Levenberg-Marquardt algorithm. The procedure can detect ColorChecker if it is in front of the camera within an allowed operating range.

Andrzej et al. present an algorithm for automatic ColorChecker detection and color patch value extraction. The algorithm can detect different types of CC in various images. This method performs a k-means kind of clustering over the RGB color space, using 25 colors as centroids (24 color chart + 1 background), which generates a segmentation of the regions corresponding to the patches. In the sequel, it eliminates some of the regions according to a criterion of shape and area. It then groups the regions by area and by the distance between centers of the patches. Finally, it estimates the bounding parallelogram (hypothesis) on the convex hull of the obtained groups [6]. This method assumes that the final hypothesis is a parallelogram, which is not necessarily true [7].

Software tools to detect CCC, such as CCFind<sup>2</sup> and MacDuff<sup>3</sup>, are freely available. CCFind is implemented in Matlab (Mathworks Inc.) and returns the coordinates of the center points of the color patches. By not using colors as a cue, it can be used with unconventional lighting and multispectral sensors. On the other hand, MacDuff is implemented in C++ and uses OpenCV library. By performing geometrical operations (rotations, scaling, etc.) and by computing a distance metric between the colors, the ColorChecker is finally found.

Some commercial software is also capable of doing a semi-automatic ColorChecker target detection. Examples are the X-Rite ColorChecker Passport Camera Calibration Software<sup>4</sup>, Imatest<sup>5</sup> and BabelColor PatchTool<sup>6</sup>, which tries to perform an initial automatic detection. These softwares however usually rely on human intervention to manually mark or correct the detected reference target (by dragging the cursor), after which color correction is performed. Since manual intervention is not practical in mass digitalization processes for obvious reasons of cost and speed, it is interesting to develop a fully automatic tool for the detection of one or several ColorChecker targets in digital images.

The objective here is to propose a fast and robust method for automatic multiple ColorChecker's detections in the image. We applied a convolutional neural network for the localization of the CCC trained over a synthetic dataset (the generation process of this dataset is shown in Fig. 2) and a new ColorChecker patch recognition method.

## II. PROPOSED METHOD

We present a novel methodology for the detection and recognition of multiple ColorChecker's Classic. This algorithm is easily adaptable to any ColorChecker that has a set of uniformly colored patches and a parallelogram shape. Fig. 3 shows an overview of our system pipeline. The method is comprised of two primary stages: (1) CCC candidates location in the image and (2) CCC color patches recognition and pose estimation. The code for the ColorChecker-detection<sup>7</sup> was made available in a repository.

Stage (1) is responsible for the localization of all the CCC in the image. Once the CCCs are localized, the stage (2), focuses on the image regions analysis with a high probability of containing a CCC. The initial localization in stage (1) increases the speed of the system and its accuracy as will be shown in the experiment section. Each component of the pipeline is described in this section.

### A. Deep Learning ColorChecker Localization

The first step of the proposed method is the localization of all possible ColorChecker candidates. In [5] the SIFT descriptor is employed for this task. However, this class of methods is not robust to changes in the illumination (over or underexposed) and has problems with scalability for multiple ColorChecker's in the image. Moreover, SIFT is a patented invention. A Convolutional Neural Network model provides an end-to-end solution suited for this problem.

In recent years, convolutional neural networks (CNN) with deep architectures have outperformed traditional machine learning approaches for several computer vision tasks, where a large amount of labeled training data is available. Generally, the harder the task, the deeper the needed neural network, and more training data is also required [8].

When labeled training data is unavailable, synthetic data can be generated to compensate for this lack of data. Unsupervised generative model is a recent promising approach but generally, has a slow test-time inference, because it needs new data. Remarkable results have been obtained using synthetic data solutions, including the limited form of data augmentation [9], [10].

An interesting work on text recognition in the wild [11]–[13] is an example, which was achieved by training a neural network to recognize text using synthetically generated realistic renders. Goodfellow et al. [14] addressed the problem of the recognition of house numbers in images from the Google Street View in a supervised fashion, also solving reCaptcha [15] images using synthetic data to train a recognition neural network from image to latent text. They were able to access

<sup>2</sup><http://issl.udayton.edu/index.php/research/ccfind/>

<sup>3</sup><https://github.com/ryanfb/macduff>

<sup>4</sup><http://www.xrite.com/>

<sup>5</sup><http://www.imatest.com/>

<sup>6</sup><http://www.babelcolor.com/>

<sup>7</sup><https://github.com/pedrodiamel/colorchacker-detection>

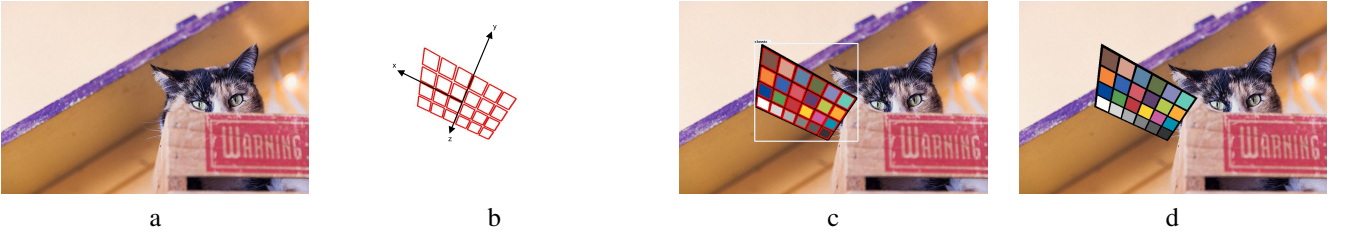


Fig. 2. The synthetic dataset generation process: from a) Original initial image, b) generated 3d model, c) random placement of the ColorChecker and d) final synthetic image.

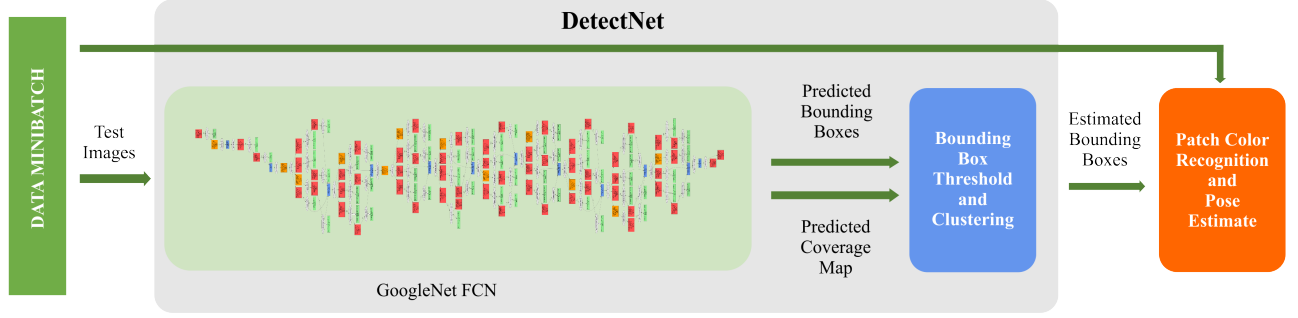


Fig. 3. The overview of the complete proposed system showing its different components.

the actual reCaptcha generative model. Therefore, they could generate millions of labeled instances to use in a standard supervised learning pipeline. More recently, Stark et al. [16] also used synthetic data for captcha solving and Wang et al. [17] for font identification.

Le et al. [8] demonstrated that the use of synthetic data to train a neural network is equivalent to train an artifact to do amortized approximate inference [18]. In this work, we created a new layer to generate synthetic data. Fig. 4 shows the training scheme for the DetectNet<sup>8</sup> using synthetic data. We describe the generation model similar to that by Le et al. [8].

The proposed synthetic data generation model for ColorChecker localization specifies the joint densities  $p(x, y)$ , that defines the latent random variable  $x$  and the corresponding ColorChecker image  $y$ . The latent structured random variable  $x = \{C, \epsilon_{1:C}, i_{1:C}\}$  includes  $C$ , the number of ColorCheckers in the image,  $\epsilon_{1:C}$ , a multidimensional structured parameter set that controls the CCC-rendering such as the various deformations types (affine deformations, color deformations, etc), and  $i_{1:C}$ , the ColorChecker identities. We use a custom stochastic CC renderer  $R$  to generate each image  $y$ . The synthetic data generator corresponds to the model:

$$y|x \sim R(x) \quad (1)$$

In particular, the model places uniform distributions over different intervals for  $C$ ,  $\epsilon_{1:C}$ , and  $i_{1:C}$ , thus generating the synthetic training data  $\{(x^n, y^n)\}$ , where  $n$  is the total number of images.

The renderer  $R$  adjusts the illumination of each ColorChecker so that it is inserted in the scene more realistically. To yield a more realistic insertion of the CCC in the scene, an additional step would be necessary to generate an alpha matte so that a final composite image of CCC and background could be constructed. The luminance channel of the ColorChecker model  $I_{cc}$  is adjusted by multiplying it by the factor  $\frac{I_r}{I_{cc}}$  where  $I_r$  is the luminance of the region that contains it in the original image. Fig. 5 shows the difference between the adjusted luminance (b) and the non-adjusted luminance.

For the detection model, we applied the DetectNet. The Fully-Convolutional Network (FCN) sub-network of DetectNet has the same structure as the GoogLeNet without the data input layers, final pooling layer and output layers [19]. This has the benefit of allowing DetectNet to be initialized using a pre-trained GoogLeNet model, thereby reducing training time and improving final model accuracy. The fully connected layers predict the output probabilities and coordinates.

### B. Patch Color Recognition and Pose Estimation

After the CC localization, the images are cropped using estimated bounding boxes, and the checker recognition step is applied to the cropped images. The proposed recognition method can be applied to the images without the previous CC detection step, but the performance is affected as shown in the results section.

Fig. 6 shows the complete recognition system pipeline. Each component of this pipeline is labeled with numbers from (01) to (14) and described in this section. The input of the system is an RGB image (01), as shown in Fig. 7. The input image is rescaled for that the smallest of the dimensions is equal to 400 and to keep the same aspect ratio. Also, we used Wiener filtering for noise suppression and RGB color normalization.

<sup>8</sup><https://github.com/NVIDIA/DIGITS/tree/master/examples/object-detection>





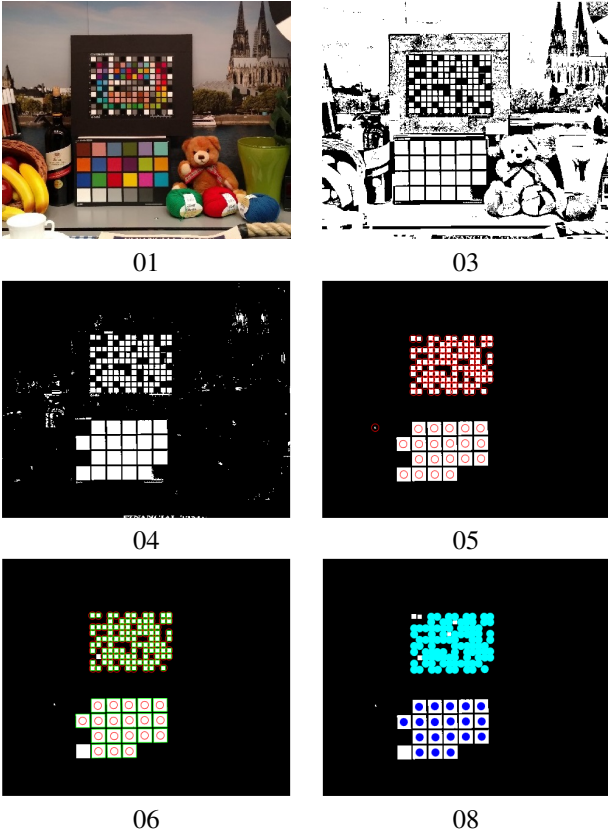


Fig. 7. The output of the image for the steps 01, 03, 04, 05, 06 and 08 from the pipeline sequence for the ColorChecker recognition, as shown in Fig. 6.

present in the patch. Since color patches belonging to ColorCheckers are designed to be homogeneous, we define  $hc = -\sum_i p(x_i) \log_2 p(x_i)$ . The criterion of  $hc < 4.9$  was applied.

From the obtained region in the previous step, we are interested in the regions that are shaped as quadrilaterals (05). Quadrilaterals are the result of perspective deformations of the CC model. To determine these type of regions, the Ramer-Douglas-Peucker algorithm [20], [21] is applied to approximate the region into a polygonal shape. Then, the polygons that contain only four points are selected. The minimal bounding parallelogram is applied to improve the results [6].

The second objective of this procedure is to assess all possible ColorChecker candidates in the scene (steps 08, 09, 10 and 11). Because of that, a variant of the Hierarchical Compact Algorithm (HCA) [22] is applied to the obtained groups of patches, as shown in Fig. 7, step (08). This method of clustering is based on the graph representation and the  $B_0$ -similarity [22] concept for generating graphs. In this case, the graph vertices are the centers of the detected patches. A weight, based on the area ratio of the patches (defined in equation 2), between each pair of nodes of the similarity graph is also established. The distance function is defined as:

$$d_{ij} = w_{ij} * \|X_i - X_j\|_2 \quad (2)$$

where  $X_i$  is the  $i$ -th patch center,  $w_{ij} = \frac{|A_i - A_j|}{(A_i + A_j)}$ , and  $A_i$  is

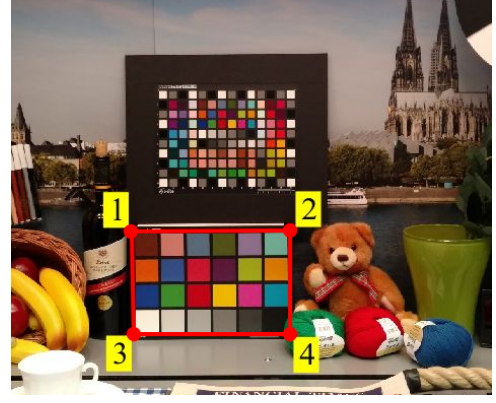


Fig. 8. The estimation of the ColorChecker Classic position and orientation, showing the four corners of the parallelogram.

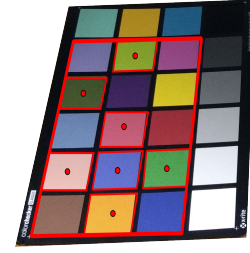


Fig. 9. The quadrilateral that best fits the ColorChecker is obtained applying the MEQ algorithm. The image shows an example of the output of Algorithm 1.

the area corresponding to the  $i$ -th patch (node in the graph). For all pairs of vertices of the graph  $\{X_i, X_j\}$ , there exists an edge if  $d_{ij} < B_{0i}$ . A dynamic value for  $B_{0i}$  was applied which is defined as:

$$B_{0i} = AxisMax_i * 1.65 \quad (3)$$

The groups of patches are formed by taking the connected components of the graph. Subsequently, each group is analyzed, and groups that have few elements (less than 4 in this case) are eliminated. In the next step, a CC estimated position is calculated (10). Also, the CC orientation in the image is identified, as shown in Fig 8.

The quadrilateral that best fits the ColorChecker in that group is calculated for each obtained group. An example is shown in Fig. 9 and the procedure is described in Algorithm 1. Unlike the method proposed by Schwarz et al. [6], this algorithm is more robust to affine transformations under several different conditions. Given the obtained minimum enclosing quadrilateral (MEQ), the center position of the missing patches and the homography matrix  $H$  are estimated with respect to the plane of the CC model, as shown in Fig. 10.

For missing point estimation, all centers points are projected into  $x$  and  $y$  axis of the estimated quadrilateral (see Fig. 11). The new points set is created with the combination of  $x$  and  $y$  projections.

Then, the parameters  $\theta$  and  $\delta$  are estimated. They represent, respectively, the rotation and the shift of the detected ColorChecker patches with respect to the original CC model.

---

**Algorithm 1** Minimum Enclosing Quadrilateral (MEQ).

**Require:**  $Ch$  set of charts, where  $ch_i \in Ch : ch_i = \{p_1, p_2, p_3, p_4\}$  is a sort clockwise set of 4 corner points.

- 1: Let  $L$  be the set of lines formed by any two consecutive points on a chart (for a chart  $ch_i = \{p_1, p_2, p_3, p_4\}$ ,  $ch_i \in Ch$ , only four straight lines can be obtained  $l_{12}, l_{23}, l_{34}, l_{41}$ ).

$$L = \{l_{ij} = p_i \times p_j, i = 1, 2, 3, 4; j = 2, 3, 4, 1\}, \forall ch_i$$

- 2: The straight lines are sorted according to how far they are from the all points set  $P$ . For each  $l_i \in L$  with  $l_i = [n_x, n_y, d]$  in homogeneous coordinates:

$$ds_i = \sum (P * [n_x, n_y]_i^T + d < 0)$$

- 3: The first four lines were selected, such that:

$$\theta > 30^\circ$$

with

$$\cos(\theta) = \frac{l_i l_{i+1}}{\|l_i\| \|l_{i+1}\|}$$

- 4: The output  $B = \{p_1, p_2, p_3, p_4\}$ , represent the intersection of the selected lines in the previous step.
- 

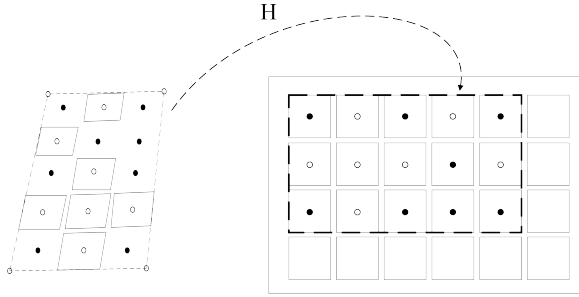


Fig. 10. The projection of the detected ColorChecker patches, showing the subset on the original CC that represents the obtained MEQ, the center position of the missing ColorChecker patches and the homography matrix  $H$ .

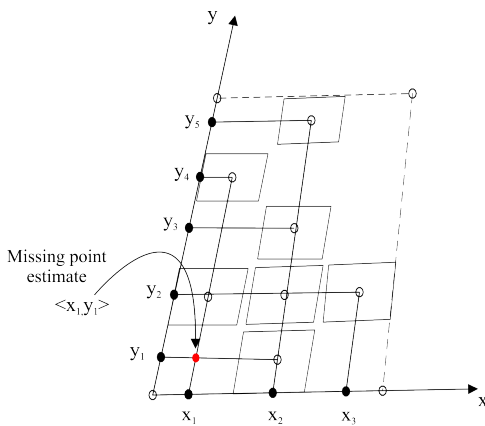


Fig. 11. Missing point estimate. The red point is an example of the estimation missing point.

For the parameters estimation, the following cost function is defined:

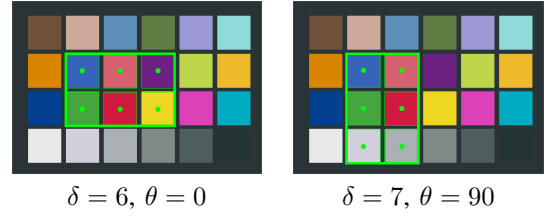


Fig. 12. Examples for the estimation of the parameters  $\theta$  and  $\delta$  that represent, respectively, the rotation and the shift of the detected ColorChecker patches.

$$J(\theta, \delta) = \|X^* - Rt(Shf(X_c, \delta), \theta)\|_2^2 \quad (4)$$

where  $Rt$  is a rotation operator and  $Shf$  is a shift operation,  $X_c$  is the detected ColorChecker patches subset, and  $X^*$  is the original color model template CC. The parameters  $\theta$  and  $\delta$  that minimize this cost function is defined as (see Fig. 12):

$$[\hat{\theta}, \hat{\delta}] = \arg \min_{\theta, \delta} J(\theta, \delta), \quad (5)$$

for angle  $\theta \in \{0, 90, 180, 270\}$  and step  $\delta \in \{1, 2, \dots, n\}$ .

An estimated position of the CC model is obtained with these parameters. The inverse of the homography matrix  $H$  is applied to obtain a model projection (hypothesis) in the image as shown in Fig. 8. For the validation of the obtained hypothesis (11) a cost function similar to the one proposed by Ernst et al. [1] is used:

$$E(p) = \sum_k \left( 1 - \frac{\mu_{k,p} r_k}{\|\mu_{k,p}\| \|r_k\|} \right) + \sum \|\sigma_{k,p}\|^2, \quad (6)$$

where  $\mu_{k,p}$  and  $\sigma_{k,p}$  denotes the mean colors and standard deviations for each hypothesis  $p$ . The parameter  $r_k$  is a reference color in the CC model. In this case,  $k$  varies from 1 to 24 since the ColorChecker Classic has 24 patches.

In this case, differently, from Ernst et al. [1], the cosine of the angle is used as a similarity function. This function is more robust to color changes than the Euclidean distance.

Due to the clustering process (step 8), several hypotheses of the same object could be generated. The last step (13) aims at eliminating the redundant hypotheses and select those that represent a CC in the image. The redundant candidates are those that represent the same CC in the scene. To eliminate redundancy, the hypotheses that present an overlap are selected. For that, the intersection over union area of the bounding box of each candidate is used, and the lower cost,  $E(p)$ , hypotheses are selected. In general, the number of CCs present in the scene is known a priori. One of the advantages of this approach is that it can handle several CCs in the image. Assuming that  $N$  ColorChecker were used in the scene, the selection of the hypotheses would be determined as follows: (1) select the  $N$  lower cost hypotheses; (2) hypotheses presenting a cost smaller than a threshold are selected.

### III. EXPERIMENTS AND RESULTS

In this section, the performance of the proposed methods are evaluated. The experiment is performed on a synthetic and a real ColorChecker dataset (GMCC) [23]. The proposed

method is referred to as MCCNetFind and a variation of this method, named MCCFind, which does not contain the localization step is also analyzed, showing the importance of the DetectNet in the detection process.

The proposed method is compared with the CCFind and MacDuff, using the following metrics for the Intersection Over Union (IOU): True Positive (TP), False Positive (FP), False Negative (FN), Accuracy (Acc), Precision (Prec), Recall (Rec), F-Measure (F-Meas) and mean Average Precision (mAP). To analyze the quality of the results, three metrics based on IOU and cosine similarity are also used:

$$a_0 = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})}$$

$$a_1 = \frac{1}{N} \sum_i \frac{\text{area}(C_p^i \cap C_{gt}^i)}{\text{area}(C_p^i \cup C_{gt}^i)}$$

$$a_2 = \frac{1}{N} \sum_i \cos(\mu_{gt}^i, \mu_p^i) = \frac{1}{N} \sum_i \frac{\mu_{gt}^i \mu_p^i}{\|\mu_{gt}^i\| \|\mu_p^i\|}$$

where  $N$  is the number of charts,  $B_p$  is the predicted bounding box,  $B_{gt}$  is the ground truth bounding box,  $C_p^i$  and  $C_{gt}^i$  are predicted areas and ground truth areas of patches  $i$  respectively,  $\mu_p$  and  $\mu_{gt}$  the mean color vector for each patch color. The performance measure  $a_0$  is used to identify the correctly located CC color target. The metrics  $a_1$  and  $a_2$  define the recognition performance of all charts and its associated colored patches, respectively.

#### A. Training the DetectNet

For the CCC localization, the DetectNet neural network using the Caffe framework<sup>9</sup> was applied. We defined a new renderer layer to generate the synthetic data and the following hyperparameters were used: size of training set 5000; size of validation set 1000; image resolution  $1024 \times 640$ ; learning rate  $10^{-4}$ ; epochs 30; batch size 30; learning rate policy fixed; momentum 0.9; weight decay  $10^{-6}$ . Fig. 13 shows the evaluation metrics of the training process. The mAP, precision and recall values obtained for the trained model on the validation set was 92.67%, 95.11% and 97.09% respectively.

#### B. Experiment using synthetic images

1) *Protocol*: The experiments are performed as follows: Number of instances (images)  $N = 1000$ ; Rigid transformation parameters, the rotation matrix  $Rt = [r_x r_y r_z]$  where  $r \in [-\pi/2, \pi/2]$ ; and the translation matrix  $Tr = [t_x t_y t_z]$  where  $t \in [-10, -30]$ ; Background (BG) image was obtained from the gratisography website<sup>10</sup>; Number of the ColorChecker models  $M = \{1, 2, 3, 4, 5\}$ .

For each iteration, a new position of the ColorChecker model is obtained randomly (this is the matrix  $[Rt|Tr]$ ) and projected over a randomly selected BG image. Fig. 14 depicts some of the 1000 generated images used for this experiment.

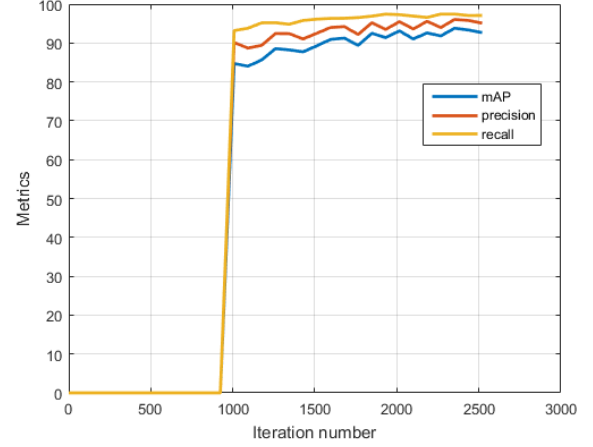


Fig. 13. Plot of the validation metrics, precision, recall and mean average precision vs the number of iterations of the DetectNet.



Fig. 14. Examples of images from the synthetic dataset with single and multiple ColorChecker.

2) *Results*: Table I displays the results obtained using the synthetic database. For each method, all images were visually analyzed and the following measures: TP, FP, FN, Acc, Prec, Rec and F-Measure, were calculated. As can be seen, the proposed MCCNetFind method presents a precision of 0.97, which indicates that the method is successful for automatic ColorChecker detection. In all cases, the proposed method exhibits an accuracy of 0.85 and an F-Measure of 0.92 which are better results compared to the results by CCFind and MacDuff. The synthetic dataset presents very complex examples with changes in position and color of the CC, for which the MCCNetFind method presented a much better recall values compared to the other methods.

The  $a_0$ ,  $a_1$  and  $a_2$  metrics were calculated to measure the quality of MCCNetFind method (only for the CCs detected, TP and FP). Fig. 15 shows the localization accuracy of the detected CC as a function of the correct localization. The  $a_1$  metric value shows that more than 70% of detected

<sup>9</sup><https://github.com/NVIDIA/caffe>

<sup>10</sup><https://gratisography.com/>



TABLE I  
RESULTS FOR THE SYNTHETIC DATASET WITH A SINGLE COLORCHECKER.

Methods	TP	FP	FN	Total	Acc	Prec	Rec	F-Meas
CCFind	334	142	524	1000	0.33	0.70	0.39	0.50
MacDuff	29	199	772	1000	0.03	0.13	0.04	0.06
MCCFind	536	3	461	1000	0.54	<b>0.99</b>	0.54	0.70
MCCNetFind	855	29	116	1000	<b>0.85</b>	0.97	<b>0.88</b>	<b>0.92</b>

TP: true positive, FP: false positive, FN: false negative, Acc: accuracy, Prec: precision, Rec: recall, F-Meas: f-measure.

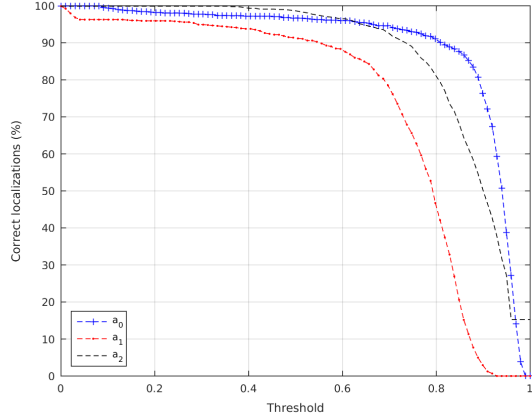


Fig. 15. Plot of the accuracy of detected CC (TP+FP) against  $a_0$ ,  $a_1$  and  $a_2$  values for the MCCNetFind method.

ColorCheckers were obtained for values smaller than 0.75 and in the case of  $a_0$  and  $a_2$  more than 95%. Therefore, it is clear that not only the method is accurate, but the results show a high quality standard.

The experiments reveals that the proposed method can perform detection of multiple CC in an image, while both CCFind and MacDuff are unable to detect multiple ColorChecker's. The Table II presents the results of the method using a synthetic database with multiple ColorChecker. In the MCCFind case, it is necessary to know a priori the number of ColorChecker's in the scene.

TABLE II  
RESULTS FOR THE SYNTHETIC DATASET WITH MULTIPLE COLORCHECKER.

Methods	TP	FP	FN	Total	Acc	Prec	Rec	F-Meas
MCCFind	1287	11	1154	2452	0.52	<b>0.99</b>	0.53	0.69
MCCNetFind	2039	122	291	2452	<b>0.83</b>	0.94	<b>0.88</b>	<b>0.91</b>

TP: true positive, FP: false positive, FN: false negative, Acc: accuracy, Prec: precision, Rec: recall, F-Meas: f-measure.

Fig. 16 illustrates examples of the results in images of multiple CCCs. A desirable feature of this method is the high processing speed since most of the methods described in the literature are slow. The algorithms implementation was done using Matlab and C++/Python. The experiments were conducted on an Intel®Core™ i7 with 2.8 GHz and a Nvidia GeForce GTX 980 Ti. The system required on average  $0.89 \pm 0.43$  seconds per image to detect the CCC in the test sequence with the C++/Python implementation. The C++/Python version<sup>11</sup> was made available in a repository.

<sup>11</sup><https://github.com/pedrodiamel/colorchacker-detection>



Fig. 16. Results of the MCCNetFind method for the synthetic dataset.

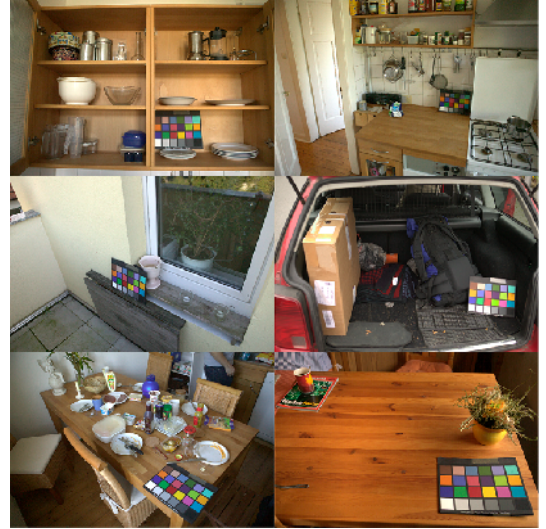


Fig. 17. Examples of images from the GMCC dataset that contain images with real ColorChecker's.

### C. Experiment using real images

1) *Protocol*: This experiment was performed on the GMCC dataset [23]. The images were captured using a high-quality digital SLR camera in RAW format, as shown in Fig. 17, so it is free from any color correction. Using the freely available software dcRAW<sup>12</sup> the images were demosaiced and converted into uncompressed linear 16-bit files. This process was done with particular attention to converting the images using always the same multiplicative gains to bypass the camera Automatic White Balance (AWB) estimation. The dataset consists of 569 images.

2) *Results*: Table III shows the results obtained for the GMCC dataset. The precision values for the proposed methods is maintained at 0.990. The ColorChecker in real images usually presents less complex transformations than in the synthetic images, so the detection method most of the time

<sup>12</sup><http://www.cybercom.net/~dcoffin/dcraw/>

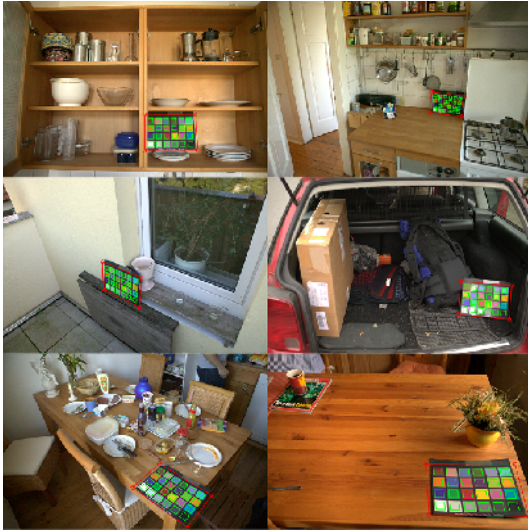


Fig. 18. Results of the MCCNetFind method for the GMCC dataset.

shows better results with improved recall rate. In this case, the results of the proposed MCCFind method present an accuracy rate of 0.920 and F-Measure of 0.960, showing that it improves the performance compared to the other methods. The proposed MCCNetFind method also outperforms (0.972 accuracy) previous algorithms from the Table III and obtains an F-Measure of 0.986 with very high precision confirming its capability for automatic ColorChecker detection. Fig. 18 portrays the MCCNetFind method's results for samples from the GMCC dataset.

TABLE III  
RESULTS FOR THE GMCC DATASET.

Methods	TP	FP	FN	Total	Acc	Prec	Rec	F-Meas
Kordecki [7]	440	110	19	569	0.770	0.800	0.960	0.870
X-Rite	306	241	22	569	0.540	0.560	0.930	0.700
CCFind	430	36	103	569	0.760	0.920	0.810	0.860
MacDuff	41	180	348	569	0.070	0.190	0.110	0.130
MCCFind	523	3	43	569	0.920	0.990	0.920	0.960
MCCNetFind	553	3	13	569	<b>0.972</b>	<b>0.995</b>	<b>0.977</b>	<b>0.986</b>

TP: true positive, FP: false positive, FN: false negative, Acc: accuracy, Prec: precision, Rec: Recall, F-Meas: F-Measure.

Most errors in the localization step (13 errors out of the 16) occurs where the ColorChecker's are too far away from the camera (see Fig. 19). The neural network does not manage to generalize these cases because the render layer generates the patterns in a defined range distance from the camera in which examples of this type do not appear. The errors by the recognition system (3 errors) are due to blurring in ColorChecker's that makes it difficult the segmentation of the charts (see Fig. 20).

In this work, we demonstrate the feasibility of the use of synthetic images to train convolutional neural networks to detect ColorChecker's. Future works will be aimed at the creation of an end-to-end model based on deep convolutional networks for the task of detection, color patch recognition and estimation of the pose of multiple ColorChecker's types.



Fig. 19. Examples of images that present errors in the localization.

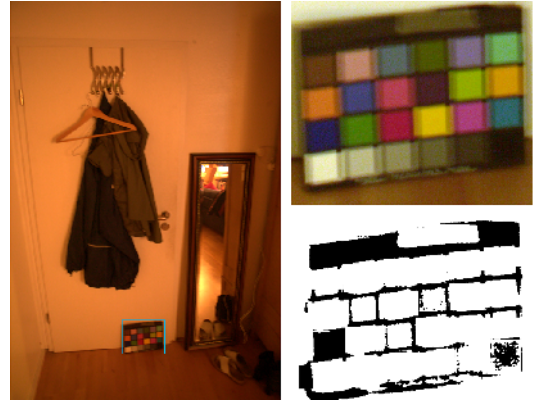


Fig. 20. Examples of images that present errors in the recognition.

#### IV. CONCLUSION

We presented a deep learning based ColorChecker detection method that showed a high accuracy and precision rates. The proposed solution is fast and completely automatic. Also, an algorithm to find the checker minimum enclosing as well as a variation of the clustering algorithm HCA [22] ensuring that the method can detect multiple CC were presented. A synthetic dataset was generated to evaluate the results. The proposed method showed an accuracy improvement of over 0.20 other methods in the state-of-the-art and high precision and recall rates. The GMCC dataset was used to evaluate real images, and the obtained accuracy was 0.972, demonstrating a significant increase compared to other methods in the literature. We also tested the influence of the deep learning detection step by applying the recognition method directly in the image. Results show that the deep learning detection improves the accuracy in 4% in the GMCC dataset, but for the synthetic dataset, the improvement was of 31%.

#### ACKNOWLEDGMENTS

This work was supported by the research cooperation project between Motorola Mobility LLC (a Lenovo Company) and CIN-UFPE. Tsang Ing Ren, Pedro D. Marrero Fernandez and Fidel A. Guerrero-Peña gratefully acknowledge financial support from the Brazilian government agency FACEPE. The authors would also like to thank Leonardo Coutinho



de Mendonça, Alexandre Cabral Mota, Rudi Minghim and Gabriel Humpire for valuable discussions.

## REFERENCES

- [1] Andreas Ernst, Anton Papst, Tobias Ruf, and Jens-Uwe Garbas, “Check My Chart: A Robust Color Chart Tracker for Colorimetric Camera Calibration,” in *6th International Conference on Computer Vision / Computer Graphics Collaboration Techniques and Applications*, 2013.
- [2] Touraj Tajbakhsh and Rolf-Rainer Grigat, “Semiautomatic Color Checker Detection in Distorted Images,” in *5th IASTED International Conference on Signal Processing, Pattern Recognition and Applications*, 2008, pp. 347–352.
- [3] Daniel Kapusi, Philipp Prinke, Rainer Jahn, Darko Vehar, Rico Nestler, and Karl-Heinz Franke, “Simultaneous geometric and colorimetric camera calibration,” in *Tagungsband 16. Workshop Farbbildverarbeitung*, 2010.
- [4] Gary Bradski and Adrian Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, 2008.
- [5] Simone Bianco and Claudio Cusano, “Color Target Localization Under Varying Illumination Conditions,” in *3rd International Workshop on Computational Color Imaging*, 2011, pp. 245–255.
- [6] Christian Schwarz, Jrgen Teich, Alek Vainshtein, Emo Welzl, and Brian L Evans, “Minimal enclosing parallelogram with application,” in *11th Annual Symposium on Computational Geometry*, 1995, pp. 434–435.
- [7] Andrzej Kordecki and Henryk Palus, “Automatic detection of colour charts in images,” *Przegląd Elektrotechniczny*, vol. 90, no. 9, pp. 197–202, 2014.
- [8] Tuan Anh Le, Atilim Giineş Baydin, Robert Zinkov, and Frank Wood, “Using synthetic data to train neural networks is model-based reasoning,” in *International Joint Conference on Neural Networks, (IJCNN 2017)*, 2017, pp. 3514–3521.
- [9] P Y Simard, D Steinkraus, and John C Platt, “Best practices for convolutional neural networks applied to visual document analysis,” in *7th International Conference on Document Analysis and Recognition*, 2003, pp. 958–963.
- [10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.
- [11] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, “Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition,” *arXiv:1406.2227v4 [cs.CV]*, pp. 1–10, 2014.
- [12] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, “Reading Text in the Wild with Convolutional Neural Networks,” *International Journal of Computer Vision*, vol. 116, no. 1, pp. 1–20, 2016.
- [13] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman, “Synthetic Data for Text Localisation in Natural Images,” in *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR 2016)*, 2016, pp. 2315–2324.
- [14] IJ Goodfellow, Yaroslav Bulatov, and Julian Ibarz, “Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks,” *arXiv:1312.6082v4 [cs.CV]*, pp. 1–13, 2014.
- [15] Luis Von Ahn, Benjamin Maurer, Colin Mcmillen, David Abraham, and Manuel Blum, “reCAPTCHA: Human-Based Character Recognition via Web Security Measures,” *Science*, vol. 321, no. 12, pp. 1465–1468, 2008.
- [16] Fabian Stark, Rudolph Triebel, and Daniel Cremers, “CAPTCHA Recognition with Active Deep Learning,” in *37th German Conference on Pattern Recognition*, 2015.
- [17] Zhangyang Wang, Jianchao Yang, Hailin Jin, Eli Shechtman, Aseem Agarwala, Jonathan Brandt, and Thomas S. Huang, “DeepFont: Identify Your Font from An Image,” in *23rd ACM International Conference on Multimedia*, 2015, pp. 451–459.
- [18] Samuel J Gershman and Noah D Goodman, “Amortized Inference in Probabilistic Reasoning,” in *36th Annual Conference of the Cognitive Science Society, (CogSci 2014)*, 2014, pp. 517–522.
- [19] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, “Going Deeper with Convolutions,” in *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR 2015)*, 2015, pp. 1–9.
- [20] Urs Ramer, “An Iterative Procedure for the Polygonal Approximation of Plane Curves,” *Computer Graphics and Image Processing*, vol. 1, no. 3, pp. 244–256, 1972.
- [21] David H. Douglas and Thomas K. Peucker, “Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature,” *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.
- [22] Reynaldo J Gil-Garcia, Jos Manuel Badia-Contelles, and Aurora Pons-Porrata, “A general framework for agglomerative hierarchical clustering algorithms,” in *IEEE 18th International Conference on Pattern Recognition, (ICPR 2006)*, 2006, pp. 569–572.
- [23] Peter Vincent Gehler, Carsten Rother, Andrew Blake, Tom Minka, and Toby Sharp, “Bayesian color constancy revisited,” in *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR 2008)*, 2008, pp. 1–8.