Pattern Recognition Letters 192 (2025) 51–58

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

# Robust camera-independent color chart localization using YOLO

Luca Cogo *, Marco Buzzelli, Simone Bianco, Raimondo Schettini

*Department of Informatics, Systems and Communication, University of Milano-Bicocca, Viale Sarca 336, Building U14, Milan, 20126, Italy*

ARTICLE INFO

Editor: Song Wang

*Keywords:*
Object detection
Color target
Pose estimation

ABSTRACT

Accurate color information plays a critical role in numerous computer vision tasks, with the Macbeth ColorChecker being a widely used reference target due to its colorimetrically characterized color patches. However, automating the precise extraction of color information in complex scenes remains a challenge. In this paper, we propose a novel method for the automatic detection and accurate extraction of color information from Macbeth ColorCheckers in challenging environments. Our approach involves two distinct phases: (i) a chart localization step using a deep learning model to identify the presence of the ColorChecker, and (ii) a consensus-based pose estimation and color extraction phase that ensures precise localization and description of individual color patches. We rigorously evaluate our method using the widely adopted NUS and ColorChecker datasets. Comparative results against state-of-the-art methods show that our method outperforms the best solution in the state of the art achieving about 5% improvement on the ColorChecker dataset and about 17% on the NUS dataset. Furthermore, the design of our approach enables it to handle the presence of multiple ColorCheckers in complex scenes. Code will be made available after pubblication at: https://github.com/LucaCogo/ColorChartLocalization.

## 1. Introduction

Acquiring faithful color information constitutes a basic requirement for several digital imaging and recognition tasks, such as color constancy, device colorimetric characterization, faithful color rendering and reproduction (as for example skin tones), and color object segmentation and recognition. A commonly adopted approach involves employing a standard color target or color chart, that are commonly composed of several color patches whose color coordinates in a standard device-independent color space are known and certified. Usually one or more charts are positioned within the scene and their positions and orientations may be not known in advance. Therefore, the user needs to perform two tasks: (i) detect the color chart(s) in the scene; (ii) precisely segment the different patches to extract their average color in the device color space. At this point it is possible to establish the mathematical relationship among the extracted device-dependent color coordinates and their corresponding tabulated device-independent coordinates.

Although extracting color information from these patches is not inherently challenging, the repetitive nature of the process for several images transforms it into a tedious and time-consuming task; in the case of large datasets of images or videos, this approach is clearly unfeasible. Consequently, numerous methods have emerged over the years to automate this process. The first approaches were based on a semi-automatic methodology, whereas more recent advancements have introduced fully-automatic solutions. Despite these advances, many existing methods lack robustness and often cannot handle scenes with multiple targets. Furthermore, a majority of methods employ color information as a cue for target localization, making them less adaptable to their use when the images to be processed are acquired with cameras having RGB sensors with different transmittances, in situations with unconventional lighting conditions (e.g., colored led lights), or when imaging devices that do not acquire images in RGB are used (e.g., hyperspectral cameras, ir-cameras, etc.).

In this work we propose a fully-automatic method that:

- makes it possible to simultaneously detect multiple targets in the same scene;
- accurately segments the color patches within the detected targets whatever is their position, orientation and scale in the scene;
- is designed to operate without relying on color information; using only grey-scale information it is able to work whatever is the imaging device adopted;
- can be adapted to detect and process any color chart (e.g., [1]).

* Corresponding author.
*E-mail addresses:* luca.cogo@unimib.it (L. Cogo), marco.buzzelli@unimib.it (M. Buzzelli), simone.bianco@unimib.it (S. Bianco), raimondo.schettini@unimib.it (R. Schettini).

https://doi.org/10.1016/j.patrec.2025.03.022
Received 4 October 2024; Received in revised form 28 January 2025; Accepted 19 March 2025
Available online 28 March 2025

**Fig. 1.** Examples of real-world scenes containing a Macbeth ColorChecker, from the Shi-Gehler dataset [2] (gamma correction is applied for visibility purposes).

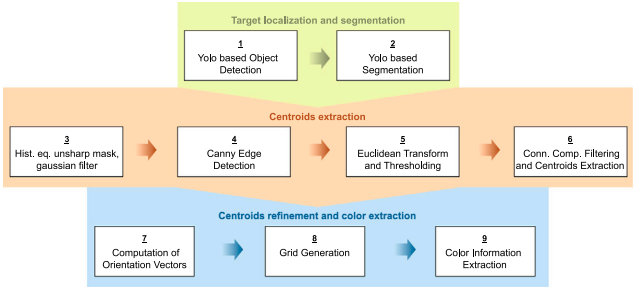



**Fig. 2.** Flow chart of the entire method.

In this work we limit our investigation to the detection of Macbeth ColorChecker since it is the most commonly used color target in imaging [3–6][7,8]

Coherently with previous works, our experiments were conducted on the standard ColorChecker [2] and NUS [9] datasets, and demonstrated that our method outperforms existing state-of-the-art techniques in both accuracy and reliability. Additional experiments are conducted to investigate the effect of image blur and how the size of the color target affects the detection performance.

## 2. Related works

Color targets play a crucial role in color imaging, with numerous target designs proposed to meet the diverse needs of the field. Among these target designs, the Macbeth ColorChecker (MCC) [10] stands out as the most widely recognized and utilized color target. Introduced in 1976 by the Macbeth Division of Kollmorgen Instruments Corporation, the chart is composed by a planar rectangular $4 \times 6$ array of square patches affixed to a black rigid support. The patches, coated with a matte paint, feature carefully selected and controlled color coordinates, ensuring comprehensive coverage of the CIE 1931 color space, making the chart suitable for representing the color gamut of a wide range of artificial and natural scenes. Fig. 1 shows some examples of real-world scenes containing the chart. The features of the Macbeth ColorChecker have made it a widely adopted tool in industry [11,12], prompting researchers to focus on developing detection methods specifically for this target.

Jackowski et al. [13] in 1997 presented a unified approach to correct color and geometry in images using the Macbeth ColorChecker. Their method modeled geometric distortions and color corrections with rational Gaussian surfaces and hypersurfaces. However, it required manual annotation of the patch corners, then refined automatically via template matching. Departing from manual intervention, Kapusi et al. [14] introduced a fully-automatic approach for detecting a customized color target with MCC's colors printed on a chessboard pattern. Their design allowed them to simply adopt a chessboard detection algorithm and region growing for precise reference circles localization. However, the approach is constrained by the need for a customized color target. Bianco and Cusano [15] proposed a strategy for automatic detection of Macbeth ColorCheckers, incorporating SIFT-based feature extraction, clustering of matching features, and hypothesis validation for pose selection. In their work, different local descriptors are experimented in addition to SIFT, namely rg-SIFT and Opponent-SIFT [16]. By considering alternative feature extraction techniques, they showcased the potential of different color spaces for increased robustness to illumination changes in the scene. In contrast, Hirakawa [17] proposed CCFind as an alternative approach that does not explicitly search for squares. Instead it learns recurring shapes inside an image and uses them to locate the patches. Since the method uses only shape information, it eliminates the need for color information, making it applicable to unconventional imaging devices and lighting conditions. However, Hirakawa's method on the contrary of ours, is constrained by licensing limitations on commercial applications.

The approach by Ernst et al. [18] is camera-dependent and minimizes a custom cost function to iteratively refine an initial guess of the target corners positions and its patches colors. The cost function takes into account the difference between estimated and reference color values, as well as the standard deviation of the colors within patch regions. The function is minimized using the Levenberg–Marquardt algorithm until a specified threshold is reached.

Kordecki et al. [19] suggested a flexible approach for detecting color targets. Their method employs a $k$-means clustering with 25 seeds (24 for the patches and one for the background), followed by a connected components analysis to keep only elements with a certain shape and size. Finally, a bounding parallelogram is constructed around the convex hull of the detected connected components.

Garcia et al. [20] introduced a pre-processing step based on saliency maps to delineate an approximate Region of Interest (ROI) for the color target and thus simplify the detection process. In their work they examined the impact of this preliminary step on both CCFind [17] and a template matching approach, demonstrating performance improvement for both methods.

Fernandez et al. [21] pioneered the integration of deep learning into color target detection. Their solution is based on two steps: first a deep learning model based on GoogLeNet [22] is trained on synthetic data to detect the ROI of the target; subsequently, a pose estimation pipeline is employed to extract the color information of the patches. Notably, their approach marks a milestone as the first method capable of detecting multiple color targets within a single image, showcasing the potential of deep learning based detectors in this domain. A version of their method is also officially released as part of MATLAB's Image Processing Toolbox. [23]

## 3. Proposed method

Our approach is designed to be: fully automatic, able to accurately detect multiple targets in the scene, camera-independent (being capable of operating on grey-scale information only), and fast, robust, and accurate. To achieve this, we structured our approach in three distinct phases:

1. A deep learning model is employed to detect and segment the color target(s) within the scene
2. The coordinates of the centroids the patches are extracted
3. The centroids are refined by using a consensus based approach, and the color information of the patches is extracted

The entire method is illustrated in Fig. 2 and described in this section, while Fig. 3 shows the results of the steps in phase 2 and phase 3

### 3.1. Target localization and segmentation

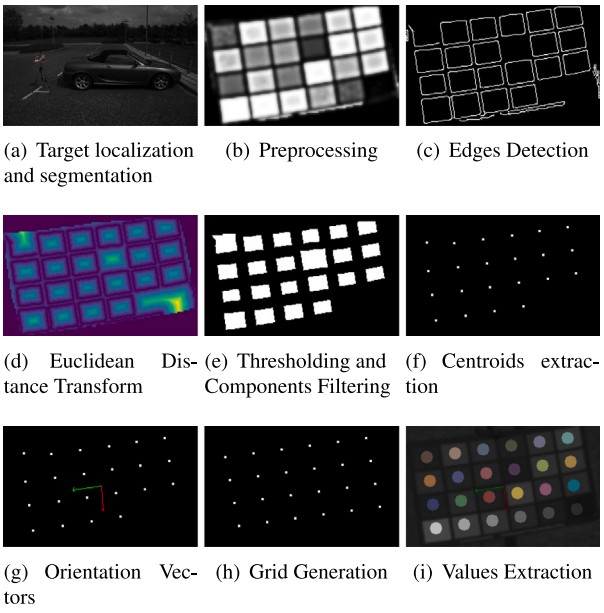
The localization of the target is accomplished through the use the YOLO object detection framework, specifically YOLOv8 [24]. This choice was made because of the versatility of the method, which supports various vision tasks, including object detection, segmentation, pose estimation, tracking, and classification. The authors of YOLO adopt a modified version of Darknet53 [25] and, for enhanced computational efficiency, they incorporate a spatial pyramid pooling layer that aggregates features into a fixed-size map. The YOLOv8 framework is released in five versions, with varying scales: YOLOv8n (nano), YOLOv8s (small), YOLOv8 m (medium), YOLOv8l (large), and YOLOv8x (extra large). For our specific objectives, we opted for the YOLOv8n model, recognized as the smallest and fastest variant, with 3.5 million parameters and 10.5 GFLOPs. The YOLOv8 framework is at the moment the most updated version which permits the training of the nano-scale detector on a custom dataset.

In our methodology, the model is employed to identify the presence of color targets within the scene. Upon detection of a Region of Interest, it is cropped and used as input for a second version of the model, whose architecture is dedicated to target segmentation and background removal. This two-step process ensures efficient and accurate localization and segmentation of the color target within the given scene. For faster inference times, the described step is performed on resized images ($640 \times 640$) and the obtained ROIs and masks are then upscaled to the original resolution.

### 3.2. Centroids extraction

Once the target is cropped and segmented, we apply a pipeline to extract the centroids' coordinates of the 24 patches in the Macbeth ColorChecker. The pipeline is designed to work on images in the [0-255] range and is structured as a sequence of steps in the following order:

1. A preprocessing step, aimed to enhance the image quality, that involves histogram equalization to enhance contrast, a gaussian filter to reduce noise, and unsharp masking to increase image sharpness. In details, we adopt a gaussian filter with a $9 \times 9$ kernel, while for the unsharp masking we use a $9 \times 9$ Gaussian kernel with $\sigma = 5$ and a multiplicative factor of 2.
2. An Edge detection step, aimed to provide an initial estimation of the edges of the color patches, performed using the Canny algorithm [26]. The Canny edge detection algorithm is subsequently performed without any additional Gaussian blur, with a $3 \times 3$ Sobel kernel for gradients computation. For the hysteresis procedure that refines the edges based on their gradients intensities and directions, we set the minimum and maximum threshold as $t_1 = 80$ and $t_2 = 170$ respectively (additional details about how these parameters were chosen can be found in supplementary materials).
3. A patches binarization step, aimed to separate the color patches from the dark background of the Macbeth ColorChecker, based on the application of the Euclidean distance transform [27] to the detected edges. This step is needed since the output of the edge detection does not guarantee that all the edges of the patches have been detected and closed. The usage of the Euclidean distance transform allows to obtain a probability map of the centroids positions, helping to compensate for eventual missed edges. The obtained probability map is finally binarized by keeping only the non-zero values. To further refine the binarization results, we filter the connected components, removing those with non-convex shapes or a total area significantly deviating from the average area of the detected patches.
4. A centroids extraction step, based on the computation of the center of mass of each binarized patch.



(a) Target localization and segmentation (b) Preprocessing (c) Edges Detection

(d) Euclidean Distance Transform (e) Thresholding and Components Filtering (f) Centroids extraction

(g) Orientation Vectors (h) Grid Generation (i) Values Extraction

**Fig. 3.** The steps of the centroids extraction and refinement phases on an example image.

### 3.3. Centroids refinement and color extraction

At this stage, the proposed approach generates a set of coordinates corresponding to the centroids of the Macbeth ColorChecker's patches. However, it is not inherently ensured that exactly 24 centroids are identified, matching the number of patches in the chart; there may be outliers or missing centroids. Additionally, the accuracy of the coordinates is not guaranteed. To address these issues, a refinement process is implemented to identify missing centroids, eliminate outliers, and ultimately produce a $4 \times 6$ grid of centroids. This refinement involves three steps:

1. Computation of 2D Chart Orientation Vectors: We calculate two orientation vectors, $v_1$ and $v_2$, whose directions and lengths correspond to the perspective orientation of the chart and the distance between two centroids in the respective directions. In more details, to determine the orientation vectors, we begin by computing their angles, $\alpha_1$ and $\alpha_2$, using a consensus approach. For each centroid, we identify the two nearest centroids and measure the angles of the connecting vectors. The two most frequent angles, with a tolerance of $t_\alpha = \pm\frac{\pi}{9}$, are selected. The chosen tolerance represents the angle between a centroid and the edges of a neighboring patch (additional details about this choice can be found in the supplementary materials). Then we compute the lengths of the vectors, $\ell_1$ and $\ell_2$, through a similar process: for each centroid, we measure the lengths of the shortest vectors connecting it to other centroids that have an orientation $\alpha_{\{1,2\}}$, within an angle tolerance of $t_\ell = \pm\frac{\pi}{9}$ (this was chosen following the same reasoning of $t_\alpha$). The median values of these measured lengths are then retained.
2. Grid Generation: Utilizing the orientation vectors, we generate a $4 \times 6$ grid of coordinates that best fits the initially identified centroids.
   The orientation vectors $v_1$ and $v_2$ are utilized to generate, for each centroid, a candidate $4 \times 6$ grid that provides an interpretation of its position with respect to the others. This is done by considering all the possible interpretations, accounting for both horizontal and vertical layouts, where each centroid could potentially correspond to 24 positions for each layout, totaling

**Table 1**
Comparison of the methods on the ColorChecker and NUS datasets. Cosine similarity is computed only on non-missed detections, while the success rate takes into account missed localizations and adopts a threshold of 0.9.

| Method | Test dataset | # images | Missed localizations ↓ | Cosine similarity ($S$) ↑ | Succ. rate ($R$) ↑ |
|---|---|---|---|---|---|
| MATLAB | ColorChecker | 56 | 35 (62.5%) | 98.68% | 35.71% |
| CCFind | ColorChecker | 56 | 3 (5.36%) | 97.84% | 94.64% |
| Ours | ColorChecker | 56 | **0 (0%)** | **99.80%** | **100.00%** |
| MATLAB | NUS | 1853 | 928 (50.08%) | 98.92% | 49.43% |
| CCFind | NUS | 1853 | 364 (19.64%) | 99.82% | 80.30% |
| Ours | NUS | 1853 | **43 (2.32%)** | **99.98%** | **97.67%** |

48 possible interpretations per centroid. To each candidate configuration is then assigned a score based on how many grid points match the previously extracted centroids. The score is determined by counting how many grid points are matched by a centroid within a tolerance radius $r = \min(\ell_1, \ell_2)/4$. The candidate configurations with the highest scores are then averaged to further refine the grid. Subsequently, to disambiguate the grid's orientation and correctly associate colors with each grid point, the black patch is identified as the corner grid point with the lowest intensity (computed as the average value over its $9 \times 9$ neighborhood). By determining the black patch, all the other patches are also uniquely identified.

3. Color Information Extraction: The refined coordinates grid is employed to extract the color information from the patches in the original image. In particular, we extract the color information by assigning to each point a circular mask with the tolerance radius $r$ defined in the previous step. Each mask is further refined by retaining only the pixels with intensity values within the 2nd and 3rd quartiles. The color information is finally determined by averaging the remaining values.

### 3.4. Visual summary of the method

A visual summary of the proposed method is depicted in Fig. 3:

(a) The target localization and segmentation phase takes a grayscale image as input, identifies the ROI containing a ColorChecker and segments it from the background.
(b) The cropped and segmented image is preprocessed using histogram equalization, gaussian filter, and unsharp masking. This helps to ehnance image quality and make the edges of the patches more detectable.
(c) The Canny algorithm detects the edges of the chart. Some edges of the patches could be missed, like the top-left patch, the one at row one and column four, and the two patches on the bottom right.
(d-e-f) The Euclidean distance (d) transform helps recovering the patches with incomplete edges. The thresholding and filtering step (e) helps removing eventual outliers, but this operation could also cause the loss of some patches. Therefore, when the centroids ere extracted (f), some might be missing.
(g-h) To refine the centroids, the chart orientation vectors are computed (g) and used to generate the best matching grid points (h). The missing centroids (e.g. the bottom-right ones) are recovered and the misaligned ones (e.g. the top-left one) are fixed.
(i) The orientation of the grid points is disambiguated by identifying the colors of the patches, and the color information is finally extracted from the original image.

## 4. Experiments

### 4.1. Datasets

For our target detection experiments, we employ two largely used datasets for color constancy research featuring the Macbeth Color Checker chart [28–31]: the ColorChecker dataset [32] and the NUS dataset [9].

**Fig. 4.** Instances of tested images where only the proposed method correctly localized the ColorChecker (gamma correction is applied for visibility purposes).

- *ColorChecker* dataset: Initially introduced by Gehler et al. this dataset comprises 568 images taken with two distinct cameras : Canon EOS-1Ds (86 images with resolution 2041 × 1359) and Canon EOS-5D (482 images with resolution 2193 $x$ 1460). The dataset provides for each scene the coordinates of the ColorChecker's corners, the RGB values of its patches and the groundtruth scene illuminant computed from them. Due to the complexity of having a faithful manual annotation of the chart position in the scene and of its patches, the dataset has undergone several corrections to its ground truth over time, including the "reprocessed" edition by Shi and Funt [2], and the more recent "recommended" version by Hemrit et al. [33]. For the purposes of our study, we utilized the most recent version.
- *NUS* dataset: This dataset is a collection of 1853 images curated by the National University of Singapore (NUS), captured with nine distinct cameras with resolution ranging between 6 Mpx and 24 Mpx. All images feature a Macbeth ColorChecker target, depicting a variety of scenes including indoor, outdoor, close-up, and people. The dataset also provides, for each ColorChecker chart, the coordinates of its bounding box and the RGB values of its patches.

### 4.2. Experimental setup

For training our target detection and segmentation models (as described in Section 3), we fine-tuned them on the ColorChecker dataset, starting from weights pre-trained on the COCO dataset [34].

The ColorChecker dataset was randomly split into 80% for training (456 images), and two 10% splits (56 images each) for validation and testing, respectively. The NUS dataset was reserved solely for cross-dataset evaluation, as only the ColorChecker dataset provides the required metadata — namely, the exact coordinates of the four chart corners — for training the segmentation model.

Training was conducted over 100 epochs with a batch size of 16, a learning rate of 0.01, and weight decay set at $5 \times 10^{-4}$. All training and evaluation procedures were carried out using a single NVIDIA GeForce GTX 1070 GPU with 8 GB of RAM.

For performance comparison, we selected two baseline methods: CCFind, proposed by Hirakawa [17], and the MATLAB implementation by Fernandez et al. [21]. These methods were chosen because they represent the most recent approaches in the state of the art, and they provide readily available code for comparison. These were evaluated alongside our proposed method.

The results are measured in terms of average cosine similarity between the extracted RGB colors of the patches on the detected target and the ground-truth RGB colors of the patches. The similarity score, denoted as $S$, is defined as follows:

$$S = \frac{1}{N} \sum_{i=1}^{N} \frac{\mu_{gt}^{i} \cdot \mu_{p}^{i}}{\|\mu_{gt}^{i}\| \, \|\mu_{p}^{i}\|} \tag{1}$$

where $\mu_{gt}^{i}$ and $\mu_{p}^{i}$ are the RGB ground truth and RGB predicted colors of the $i$th patch, respectively, and $N$ is the total number of patches (i.e., 24 for the Macbeth ColorChecker). The values of $S$ are in the range [0,1], where 1 is the best possible value and 0 is the worst. This metric provides an information of how closely the predicted patch colors match the actual ground-truth.

We chose this metric for two main reasons:

1. it ensures consistency with the evaluation of Fernandez et al. [21]
2. it guarantees a fair comparison between methods, directly evaluating how good they are at their final task, which is retrieving color information from images containing ColorCheckers.

The so defined cosine similarity score is computed for all the localizations, while we separately report the amount of missed localizations, as the number of times the methods mistakenly consider the scene as having no Macbeth ColorChecker in it. This value is particularly relevant in the context of automatic labeling of color constancy datasets: with the process of acquiring real-world scenes being slow and costly, each missed localization constitutes a wasted acquisition.

To jointly take into account the two measurements, we also compute a success rate $R$ as the percentage of non-missed localizations with a cosine similarity score greater than 0.9:

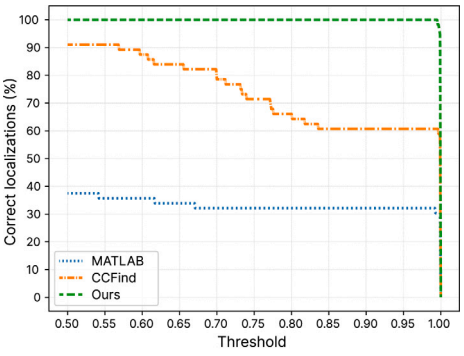
$$R = \frac{|E|}{N} \text{ where } E = \{e \in M \, : \, e \geq 0.9\} \tag{2}$$

where $M$ is the set of cosine similarities computed on all the localizations, and $N$ is the total number of elements in the test set. This value represents the amount of actually usable images after employing the methods for automatic labeling.

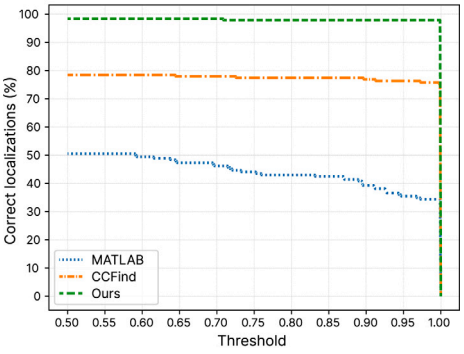
### 4.3. Experimental results

The experimental results, summarized in Table 1, demonstrate that our method consistently outperforms the compared approaches across both datasets. Specifically, a higher cosine similarity score is achieved on the correctly localized patches and fewer missed localizations are observed.

Fig. 5 presents a visual comparison of the percentage of correct chart localizations as a function of the cosine similarity score. As shown, our method maintains a higher accuracy throughout the range of similarity scores compared to CCFind and Fernandez et al.'s method. Fig. 6 further illustrates the superiority of our method through violin plots depicting the distribution of cosine similarity scores for all tested images. The proposed method consistently produces higher cosine similarities, indicating a more precise retrieval of color information. These results reveal that our method not only achieves higher localization accuracy but also significantly reduces the number of missed localizations, making it

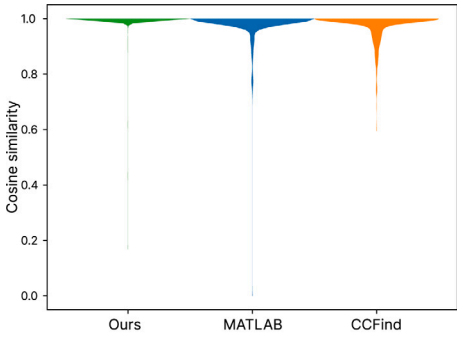



(a) ColorChecker

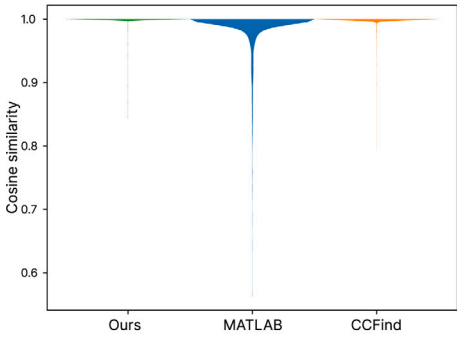



(b) NUS

**Fig. 5.** Plot of the rates of correct chart localization as a function of the cosine similarity obtained on (a) ColorChecker dataset, (b) NUS dataset.




(a) ColorChecker




(b) NUS

**Fig. 6.** Violin plots of the cosine similarities distribution obtained on (a) ColorChecker dataset, (b) NUS dataset.

**Table 2**
Comparison of the methods on the ColorChecker and NUS datasets with pre-cropped ROIs. Cosine similarity is computed only on non-missed detections, while the success rate takes into account missed localizations and adopts a threshold of 0.9.

| Method | Test dataset | # images | Missed localizations↓ | Cosine similarity ($S$) ↑ | Succ. rate ($R$) ↑ |
|---|---|---|---|---|---|
| MATLAB | ColorChecker | 56 | 38 (67.86%) | 98.34% | 32.14% |
| CCFind | ColorChecker | 56 | 32 (57.14%) | 96.34% | 37.5% |
| Ours | ColorChecker | 56 | **2 (3.57%)** | **99.80%** | **96.43%** |
| MATLAB | NUS | 1853 | 1584 (85.48%) | 95.80% | 14.52% |
| CCFind | NUS | 1853 | 1512 (81.60%) | 97.59% | 17.91% |
| Ours | NUS | 1853 | **62 (3.35%)** | **99.97%** | **96.65%** |

more suitable for automatic dataset annotation compared to the other methods considered.

To further isolate and evaluate the effectiveness of the pose estimation step, we conducted an additional set of experiments using pre-cropped ColorChecker charts as inputs. The results, summarized in Table 2, reveal that our method's performance remained stable, while the other approaches suffered a significant performance drop. This highlights the robustness of our approach in scenarios where pose estimation plays a crucial role in target detection. Fig. 4 illustrates some examples of scenes that were correctly handled by the proposed method and missed by the others. While the other approaches happen to miss even easy localizations, our approach manages to correctly localize even the hardest ones, containing ColorCheckers that are blurred, highly skewed or far from the camera.

### 4.4. Degradation test

In order to further evaluate the performance of the proposed method in challenging scenarios, we conducted additional experiments to assess its robustness, in terms of localization success rate, with respect to image blur and the size of the color target in the scene.

To assess the impact of blur, we applied Gaussian blur with varying standard deviations $\sigma$ to the input images, and used the corresponding kernel size $f_s$ (calculated via Eq. (3)).

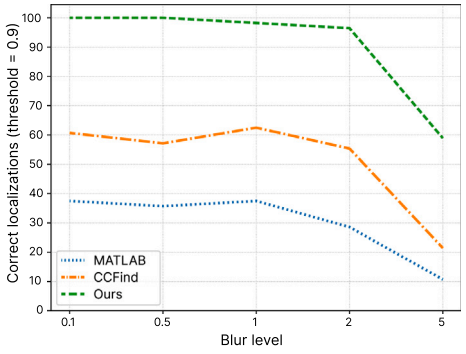
$$f_s = 2 \cdot \lceil 2\sigma \rceil + 1 \tag{3}$$

We then measured the degradation of the localization success rate as a function of the blur level.

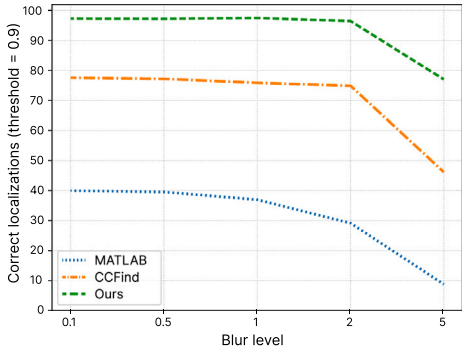
As shown in Fig. 7, our method exhibits a gradual decrease in performance as blur increases, but remains resilient under moderate blur levels. This analysis provides insights into the method's reliability under conditions of reduced image clarity, which is particularly relevant for real-world applications where images may be affected by factors such as motion blur or defocus.

Furthermore, we investigated the influence of the color target size on the localization performance. We took into account the size of the color targets within the input images and measured the corresponding success rate of target localization. The results, depicted in Fig. 8, indicate that our method maintains a high level of accuracy across a wide range of target sizes, demonstrating its ability to accurately detect targets of different scales. This analysis highlights the method's versatility and suitability for applications where color targets may vary in size or distance from the camera.

Overall, the results of these experiments further confirm the robustness and versatility of our method across challenging scenarios, validating its adaptability for diverse real-world applications.




(a) ColorChecker




(b) NUS

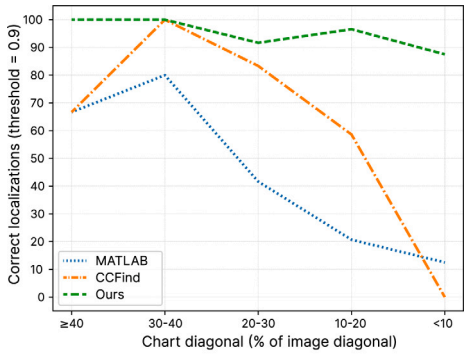
**Fig. 7.** Plot of the degradation of the localization performance with increasing blur, as function of $\sigma$.

## 5. Conclusions

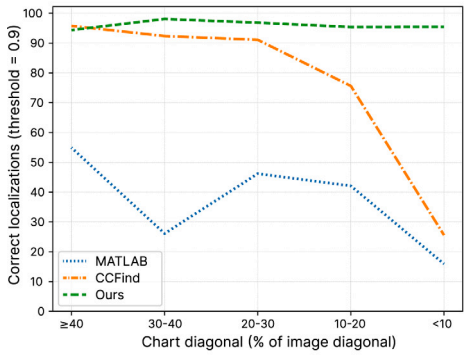
In this paper, we presented a fully-automatic method for the retrieval of color information from color targets, designed to address the limitations of existing approaches. Our method employs a two-phase process: an initial deep learning model for target detection and segmentation, followed by a consensus-based methodology for pose estimation and color extraction. The proposed solution demonstrates robust performance and can operate on grey-scale images.

Our experiments, conducted on the ColorChecker and NUS datasets, show that our method outperforms existing state-of-the-art techniques in both accuracy and reliability. Specifically, our method achieves higher cosine similarity scores and significantly lower misses in target localization compared to other existing methods. Additional experiments were conducted to evaluate the pose estimation step, and to assess the method robustness to degradations.

In summary, the proposed method not only advances the automation of color information retrieval from color targets but also offers a robust and versatile solution for a wide range of digital photography tasks. Future work could explore further enhancements, including optimization for real-time applications and adaptation to a broader spectrum of imaging conditions.




(a) ColorChecker




(b) NUS

**Fig. 8.** Plot of the degradation of the localization performance as a function of the ColorChecker size.

## CRediT authorship contribution statement

**Luca Cogo:** Writing – original draft, Software, Methodology, Investigation. **Marco Buzzelli:** Visualization, Software, Conceptualization. **Simone Bianco:** Writing – review & editing, Supervision, Methodology, Investigation, Conceptualization. **Raimondo Schettini:** Writing – review & editing, Supervision, Project administration, Conceptualization.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.patrec.2025.03.022.

## Data availability

Data and code will be published after pubblication on a Github repository, which is mentioned in the abstract of the manuscript.

## References

[1] M.R. Luo, The new preferred memory color (PMC) chart, Color Res. Appl. (2024).
[2] L. Shi, Re-processed version of the gehler color constancy dataset of 568 images, 2000, http://www.cs.sfu.ca/~{}color/data/.
[3] S. Zini, M. Buzzelli, S. Bianco, R. Schettini, COCOA: combining color constancy algorithms for images and videos, IEEE Trans. Comput. Imaging 8 (2022) 795–807.
[4] A. Raza, S. Jost, D. Dumortier, Low light hyperspectral imaging using HDR methods, in: International Workshop on Computational Color Imaging, Springer, 2024, pp. 105–116.
[5] A. Molada-Tebar, G.J. Verhoeven, D. Hernández-López, D. González-Aguilera, Practical RGB-to-XYZ color transformation matrix estimation under different lighting conditions for graffiti documentation, Sensors 24 (6) (2024) 1743.
[6] G.C. Guarnera, S. Bianco, R. Schettini, Turning a digital camera into an absolute 2D tele-colorimeter, in: Computer Graphics Forum, 38, (1) Wiley Online Library, 2019, pp. 73–86.
[7] J. Heagerty, S. Li, E. Lee, S. Bhattacharyya, S. Bista, B. Brawn, B.Y. Feng, S. Jabbireddy, J. JaJa, H. Kacorri, et al., HoloCamera: Advanced volumetric capture for cinematic-quality vr applications, IEEE Trans. Vis. Comput. Graphics (2024).
[8] C.-Y. Li, J.-C. Guo, R.-M. Cong, Y.-W. Pang, B. Wang, Underwater image enhancement by dehazing with minimum information loss and histogram distribution prior, IEEE Trans. Image Process. 25 (12) (2016) 5664–5677.
[9] D. Cheng, D.K. Prasad, M.S. Brown, Illuminant estimation for color constancy: why spatial-domain methods work and the role of the color distribution, J. Opt. Soc. Amer. A 31 (5) (2014) 1049–1058.
[10] C.S. McCamy, H. Marcus, J.G. Davidson, et al., A color-rendition chart, J. App. Photog. Eng 2 (3) (1976) 95–99.
[11] I. ISO, 17321-1: 2012 graphic technology and photography-colour characterisation of digital still cameras (DSCs)-part 1: Stimuli, metrology and test procedures, Int. Organ. Stand. (2017).
[12] B. Brown, Cinematography: theory and practice: image making for cinematographers and directors, Routledge, 2016.
[13] M. Jackowski, A. Goshtasby, S. Bines, D. Roseman, C. Yu, Correcting the geometry and color of digital images, IEEE Trans. Pattern Anal. Mach. Intell. 19 (10) (1997) 1152–1158.
[14] D. Kapusi, P. Prinke, R. Jahn, D. Vehar, R. Nestler, K.-H. Franke, Simultaneous geometric and colorimetric camera calibration, in: Tagungsband 16. Workshop Farbbildverarbeitung, 2010.
[15] S. Bianco, C. Cusano, Color target localization under varying illumination conditions, in: Computational Color Imaging: Third International Workshop, CCIW 2011, Milan, Italy, April 20-21, 2011. Proceedings 3, Springer, 2011, pp. 245–255.
[16] S. Bianco, D. Mazzini, D.P. Pau, R. Schettini, Local detectors and compact descriptors for visual search: a quantitative comparison, Digit. Signal Process. 44 (2015) 1–13.
[17] K. Hirakawa, ColorChecker finder, 2024, (Accessed: 15 April 2024)https://sites.google.com/a/udayton.edu/issl/software/macbeth-colorchecker-finder.
[18] A. Ernst, A. Papst, T. Ruf, J.-U. Garbas, Check my chart: A robust color chart tracker for colorimetric camera calibration, in: Proceedings of the 6th International Conference on Computer Vision/Computer Graphics Collaboration Techniques and Applications, 2013, pp. 1–8.
[19] A. Kordecki, H. Palus, Automatic detection of colour charts in images, Prz. Elektrotech. 90 (2014) 197–202.
[20] L.E. García Capel, J.Y. Hardeberg, Automatic color reference target detection, in: Color and Imaging Conference, 2014, (2014) Society for Imaging Science and Technology, 2014, pp. 119–124.
[21] P.D.M. Fernández, F.A.G. Peña, T.I. Ren, J.J. Leandro, Fast and robust multiple colorchecker detection using deep convolutional neural networks, Image Vis. Comput. 81 (2019) 15–24.
[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[23] MATLAB, Image Processing Toolbox version: 9.9 (R2020b), The MathWorks Inc., Natick, Massachusetts, United States, 2020, URL https://www.mathworks.com.
[24] G. Jocher, A. Chaurasia, J. Qiu, Ultralytics YOLOv8, 2023, URL https://github.com/ultralytics/ultralytics.
[25] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, 2018, arXiv preprint arXiv:1804.02767.
[26] J. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. (6) (1986) 679–698.
[27] G. Borgefors, Distance transformations in digital images, Comput. Vis. Graph. Image Process. 34 (3) (1986) 344–371.
[28] S. Bianco, C. Cusano, R. Schettini, Color constancy using CNNs, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2015, pp. 81–89.
[29] Y. Hu, B. Wang, S. Lin, Fc4: Fully convolutional color constancy with confidence-weighted pooling, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4085–4094.
[30] S. Bianco, C. Cusano, Quasi-unsupervised color constancy, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 12212–12221.

[31] M. Afifi, J.T. Barron, C. LeGendre, Y.-T. Tsai, F. Bleibel, Cross-camera convolutional color constancy, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 1981–1990.

[32] P.V. Gehler, C. Rother, A. Blake, T. Minka, T. Sharp, Bayesian color constancy revisited, in: 2008 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2008, pp. 1–8.

[33] G. Hemrit, G.D. Finlayson, A. Gijsenij, P. Gehler, S. Bianco, B. Funt, M. Drew, L. Shi, Rehabilitating the colorchecker dataset for illuminant estimation, 2018, arXiv preprint arXiv:1805.12262.

[34] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: Common objects in context, in: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13, Springer, 2014, pp. 740–755.