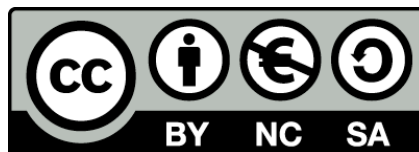




UNIVERSITAT_{DE}
BARCELONA

Automated color correction for colorimetric applications using barcodes

Ismael Benito Altamirano



Aquesta tesi doctoral està subjecta a la llicència **Reconeixement- NoComercial – CompartirIgual 4.0. Espanya de Creative Commons.**

Esta tesis doctoral está sujeta a la licencia **Reconocimiento - NoComercial – CompartirIgual 4.0. España de Creative Commons.**

This doctoral thesis is licensed under the **Creative Commons Attribution-NonCommercial-ShareAlike 4.0. Spain License.**

ISMAEL BENITO-ALTAMIRANO

AUTOMATED COLOR CORRECTION
FOR COLORIMETRY APPLICATIONS
USING BARCODES

UNIVERSITAT DE BARCELONA
PHD IN ENGINEERING AND APPLIED SCIENCES
DIRECTOR: JOAN DANIEL PRADES



UNIVERSITAT DE
BARCELONA

AUTOMATED COLOR CORRECTION FOR COLORIMETRY APPLICATIONS USING BARCODES

Programa de doctorat en Enginyeria i Ciències Aplicades

Autor: Ismael Benito-Altamirano

Director: Dr. Joan Daniel Prades

Tutor: Dr. Ángel Diéguez Barrientos


Ismael BA.



UNIVERSITAT DE
BARCELONA

Copyright © 2022 Ismael Benito-Altamirano

PUBLISHED BY UNIVERSITAT DE BARCELONA
PHD IN ENGINEERING AND APPLIED SCIENCES
DIRECTOR: JOAN DANIEL PRADES
TUTOR: ÀNGEL DIÉGUEZ

This work is licensed by a Creative Commons license .

First printing, January 2022

Acknowledgements

First, this thesis is dedicated to my family. We are a small family: my parents, my grandparents and my uncles and aunt. Specially, to my mother, who never has failed to encourage me to follow what I like to study, also for her constancy in response to my chaos. Also, specially to my father, from whom I got the passion for photography and computer science, knowing this one could understand better this thesis. To my paternal grandparents who are no longer with us. To my maternal grandmother who always has good advice, recently she said to my mother: "the kid has studied enough, since you send it to kindergarten with 3 years, he hasn't stopped", referring to this thesis!

To my friends, beginning with Anna, who is my flatmate and my partner; and who has supported me during these final thesis months. To other friends from the high school, both neighborhoods I grew up and those friends from the faculty, also. To other colleagues with whom I have shared the fight for a better university model, and now we share other fights, to my comrades!

To the Department of Electronics and Biomedical Engineering of *Universitat de Barcelona*, to all its members. Specially, to Dr. A. Cornet for being a nice host at the department. Specially, to Dr. A. Herms, to encourage me to pursue the thesis in this department and to contact Dr. J. D. Prades, director of this thesis. Also, to other colleagues from the MIND research group i from the Laboratory from 'the o floor': to Dr. C. Fàbrega, to Dr. O. Canals, and many others!

To Dr. J. D. Prades himself, for the opportunity by accepting this thesis proposal, and embrace the idea I presented, leading to the creation of ColorSensing. To the ColorSensing team, beginning with Maria Eugenia Martín, co-funder and CEO of ColorSensing. Without forgetting, all the other teammates: to Josep Maria, to Hanna, to Dani, to Maria, to Ferran and to Miriam (Dr. M. Marchena). But also, to the former teammates: to Peter, to Oriol (Dr. O. Cusola), to Arnau, to Carles, to Pablo, to Gerard, to Hamid and to David. Thank you very much all for this journey.

This thesis has been funded in part by the European Research Council under the H2020 Framework Program ERC Grant Agreements no. 727297 and no. 957527. Also, by the Eurostarts programa with the Agreement no. 11453. Secondary funding sources have been: AGAUR - PRODUCTE (2016-PROD-00036), BBVA Leonardo, and ICREA Academia programs.

Agraïments

Aquesta tesi va dedicada en primera instància a la meua família. Som una família petita: a mons pares, als meus avis i als meus tiets. Especialment, a ma mare, perquè mai a fallat en animar-me per perseguir el que m'agrada estudiar, també per la seva constància davant del meu desordre. Especialment també, al meu pare, per la seva passió amb la fotografia i la informàtica que des de petit m'ha inculcat, així hom pot entendre aquesta tesi molt millor. Als meus avis paterns que ja no estan. A la meua àvia materna que sempre té bons consells, i fa poc li va dir a ma mare: "si el nen ja ha estudiat prou, d'ençà que el vas portar amb tres anys (al col·legi) no ha parat d'estudiar", referint-se a aquesta tesi!

Als meus amics, començant per l'Anna, que és la meua companya de pis, i la meua parella; que m'ha recolzat durant aquests darrers mesos a casa mentre redactava la tesi. A tots aquells amics de l'institut, del barri, de la 'urba' i de la facultat. També a aquelles companyes amb les quals hem compartit lluites a la universitat des de l'època d'estudi i ara seguim compartint altres espais polítics, els i les meves camarades.

Al Departament d'Enginyeria Electrònica i Biomèdica de la Universitat de Barcelona, a tots els seus membres. Especialment, al Dr. A. Cornet per la seva acollida al departament. Al Dr. A. Herms, per animar-me a fer la tesi al Departament i contactar al Dr. J. D. Prades, director d'aquesta tesi. També, a altres companyes i companys del MIND, el nostre grup de recerca, i del Laboratori 'de planta o': al Dr. C. Fàbrega, la Dra. O. Casals, i tots els altres!

Al mateix Dr. J. D. Prades, per l'oportunitat acceptant aquesta tesi, i acollir la idea que li vaig presentar fins al punt d'impulsar la creació de ColorSensing. A tot l'equip de ColorSensing, començant per la Maria Eugènia Martín, cofundadora i CEO de ColorSensing. Però, per suposat, a la resta de l'equip: al Josep Maria, a la Hanna, al Dani, a la Marí, al Ferran i a la Miriam (Dra. M. Marchena). Però també als seus antics membres amb qui hem coincidit: al Peter, a l'Oriol (Dr. O. Cusola), a l'Arnau, al Carles, al Pablo, al Gerard, a l'Hamid i al David. Moltes gràcies a tots i totes per fer aquest viatge conjuntament.

Index

Abstract	11
1 Introduction	15
1.1 Objectives	19
1.2 Thesis sctructure	20
2 Background and methods	21
2.1 The image consistency problem	21
2.1.1 Color reproduction	22
2.1.2 Image consistency	23
2.1.3 Color charts	24
2.2 2D Barcodes: the Quick-Response Code	26
2.2.1 Scalability	28
2.2.2 Data encoding in QR Codes	29
2.2.3 Computer vision features of QR Codes	32
2.2.4 Readout of QR Codes	34
2.3 Data representation	37
2.3.1 Color spaces	37
2.3.2 Color transformations	38
2.3.3 Images as bitmaps	41
2.4 Computational implementation	43
3 QR Codes on challenging surfaces	45
3.1 Proposal	46
3.1.1 Fundamentals of projections	47
3.1.2 Proposed transformations	48

3.2	Experimental details	52
3.2.1	Datasets	52
3.3	Results	55
3.3.1	Qualitative surface fitting	55
3.3.2	Quantitative data readability	58
3.4	Conclusions	61
4	Back-compatible Color QR Codes	63
4.1	Proposal	64
4.1.1	Color as a source of noise	65
4.1.2	Back-compatibility proposal	68
4.2	Experimental details	73
4.2.1	Color generation and substitution	73
4.2.2	Placing colors inside the QR Code	75
4.2.3	QR Code versions and digital IDs	75
4.2.4	Channels	75
4.3	Results	77
4.3.1	Embedding colors in QRs codes: empty channel	77
4.3.2	Image augmentation channel	78
4.3.3	Colorimetry setup as channel	79
4.3.4	Readability	81
4.3.5	Example of use case	85
4.4	Conclusions	86
5	Image consistency using an improved TPS_{3D} method	89
5.1	Proposal	90
5.1.1	Linear corrections	91
5.1.2	Polynomial corrections	93
5.1.3	Thin-plate spline correction	95
5.2	Experimental details	98
5.2.1	Dataset and pipeline	100
5.2.2	Benchmark metrics	103
5.3	Results	105
5.3.1	Detecting failed corrections	105

5.3.2	Color correction performance	106
5.3.3	Execution time performance	109
5.4	Conclusions	112
6	Application: Colorimetric indicators	115
6.1	Proposal	118
6.1.1	Early prototypes	118
6.1.2	A machine-readable pattern for colorimetric indicators	120
6.1.3	A Color QR Code for colorimetric indicators	122
6.2	Experimental details	124
6.2.1	Sensor fabrication	124
6.2.2	Experimental setup	125
6.2.3	Expected response model	128
6.3	Results	129
6.3.1	The color response	129
6.3.2	Model fitting	132
6.4	Conclusions	140
7	Conclusions	143
7.1	Thesis conclusions	143
7.2	Future work	145
	List of Figures	147
	List of Tables	160
	Bibliography	163

Abstract

Color-based sensor devices often offer qualitative solutions, where a material change its color from one color to another, and this change is observed by a user who performs a manual reading. These materials change their color in response to changes in a certain physical or chemical magnitude. Nowadays, we can find colorimetric indicators with several sensing targets, such as: temperature, humidity, environmental gases, etc. The common approach to quantize these sensors is to place *ad hoc* electronic components, e.g. a reader device.

With the rise of smartphone technology, the possibility to automatically acquire a digital image of those sensors and then compute a quantitative measure is near. By leveraging this measuring process to the smartphones, we avoid the use of *ad hoc* electronic components, thus reducing colorimetric application cost. However, there exists a challenge on how-to acquire the images of the colorimetric applications and how-to do it consistently, with the disparity of external factors affecting the measure, such as ambient light conditions or different camera modules.

In this thesis, we tackle the challenges to digitize and quantize colorimetric applications, such as colorimetric indicators. We make a statement to use 2D barcodes, well-known computer vision patterns, as the base technology to overcome those challenges. We focus on four main challenges: (I) to capture barcodes on top of real-world challenging surfaces (bottles, food packages, etc.), which are the usual surface where colorimetric indicators are placed; (II) to define a new 2D barcode to embed colorimetric features in a back-compatible fashion; (III) to achieve image consistency when capturing images with smartphones by reviewing existent methods and proposing a new color correction method, based upon thin-plate splines mappings; and (IV) to demonstrate a specific application use case applied to a colorimetric indicator for sensing CO_2 in the range of modified atmosphere packaging –MAP–, one of the common food-packaging standards.

Resum

Els dispositius de sensat basats en color, normalment ofereixen solucions qualitatives, en aquestes solucions un material canvia el seu color a un altre color, i aquest canvi de color és observat per un usuari que fa una mesura manual. Aquests materials canvien de color en resposta a un canvi en una magnitud física o química. Avui en dia, podem trobar indicadors colorimètrics que amb diferents objectius, per exemple: temperatura, humitat, gasos ambientals, etc. L'opció més comuna per quantitzar aquests sensors és l'ús d'electrònica addicional, és a dir, un lector.

Amb l'augment de la tecnologia dels telèfons intel·ligents, la possibilitat d'automatitzar l'adquisició d'imatges digitals d'aquests sensors i després computar una mesura quantitativa és a prop. Desplaçant aquest procés de mesura als telèfons mòbils, evitem l'ús d'aquesta electrònica addicional, i així, es redueix el cost de l'aplicació colorimètrica. Tanmateix, existeixen reptes sobre com adquirir les imatges de les aplicacions colorimètriques i de com fer-ho de forma consistent, a causa de la disparitat de factors externs que afecten la mesura, com per exemple la llum ambient o les diferents càmeres utilitzades.

En aquesta tesi, encarem els reptes de digitalitzar i quantitzar aplicacions colorimètriques, com els indicadors colorimètrics. Fem una proposta per utilitzar codis de barres en dues dimensions, que són coneguts patrons de visió per computador, com a base de la nostra tecnologia per superar aquests reptes. Ens focalitzem en quatre reptes principals: (I) capturar codis de barres sobre de superfícies del món real (ampolles, safates de menjar, etc.), que són les superfícies on usualment aquests indicadors colorimètrics estan situats; (II) definir un nou codi de barres en dues dimensions per encastar elements colorimètrics de forma retro-compatible; (III) aconseguir consistència en la captura d'imatges quan es capturen amb telèfons mòbils, revisant mètodes de correcció de color existents i proposant un nou mètode basat en transformacions geomètriques que utilitzen splines; i (IV) demostrar l'ús de la tecnologia en un cas específic aplicat a un indicador colorimètric per detectar CO₂ en el rang per envasos amb atmosfera modificada –MAP–, un dels estàndards en envasos de menjar.

Chapter 1. Introduction

The rise of the smartphone technology developed in parallel to the popularization of digital cameras enabled an easier access to photography devices to the people. Nowadays, modern smartphones have onboard digital cameras that can feature good color reproduction for imaging uses [1].

Alongside with this phenomenon, there has been a popularization of color-based solutions to detect biochemistry analytes [2]. Both phenomena are probable to be linked. As the first one eases the second. Scientists who want to pursue research to discover new or improve existent color-based analytics found themselves with better and better imaging tools, spending fewer and fewer resources.

Color-based sensing [2] is often preferred over electronic sensing [3] for three reasons: one, the rapid detection of the analytes; two, the high sensitivity; and three, the high selectivity of colorimetric sensors. Nevertheless, imaging acquisition on smartphone devices still presents some acquisition challenges, and how to overcome those challenges is still an open debate [4].

This is why, the ERC-StG BetterSense project (ERC n. 336917) was granted the extension ERC-PoC GasApp project (ERC n.727297). BetterSense was an ERC-funded project which aimed to solve high power consumption and the poor selectivity of electronic gas sensor technologies [5]. GasApp was an ERC-funded project that aimed to bring the capability to detect gases to smartphone technology, relying on color-based sensor technology [6].

The accumulated knowledge from BetterSense was translated into the GasApp project to create colorimetric indicators to sense target gases, the GasApp proposal is detailed in Figure 1.1. Later on, the SnapGas project (Eurostars n. 11453) was also granted to carry on this research topic, and apply the new technology to other colorimetric indicators to sense environmental gases [7].



The GasApp proposal was based upon changing the electronic devices to colorimetric indicators, thus leveraging the electronic components of the sensor readout to handheld smartphones. To do so, GasApp projected a solution implementing an array with colorimetric indicators displayed on top of a card-sized substrate to be captured by a smartphone device (see Figure 1.1).

The design of this array of colorimetric indicators presented several challenges, such as: detecting and extracting the card and the desired region of interest (sensors), embedding one or more color charts and later perform color correction techniques to achieve adequate sensor readouts at any possible scenario a mobile phone could take a capture.

The research of this thesis started in this context, then the work here presented aims to tackle these problems and resolve them with an integral solution. Let us go deeper in some of these challenges to properly formulate our thesis proposal.

First, the fabrication of the color-based sensors presents a challenge itself. There exists a common starting point in printed sensors technologies to use ink-jet printing as the first approach to the problem to fabricate a printed sensor [8; 9]. However, ink-jet printing is an expensive and often limited printing technology from the standpoint of view of mass-production [10].

Second, color reproduction is a wide-known challenge of digital cameras [11]. Often, when a digital camera captures a scene it can produce several artifacts during the capture (i.e. underexposure, overexposure, ...), this is represented in see Figure 1.2.

The problem of color reproduction, involves a directly linked problem: the problem of achieving image consistency among datasets [12]. While *color reproduction* aims at matching the color of a given object when reproduced in another device as an image (e.g. a painting, a printed photo, a digital photo on a screen, etc.), *image consistency* is the problem of taking different images of the same object in different illumination conditions and with different capturing devices, to finally obtain the same apparent colors for this object.

Figure 1.1: The GasApp proposal is presented. Left, GasApp changed the core sensing technology from electronic to colorimetric indicators. Right, the initial idea of the GasApp project, a card where colorimetric dyes are printed alongside with color charts and a QR Code.

Usually, both problems are solved with the addition of *color rendition charts* to the scene. Color charts are machine-readable patterns which contain several color references [13]. Color charts bring a systematic way of solving the image consistency problem by increasing the amount of color references to create subsequently better color corrections than the default white-balance [14; 15].

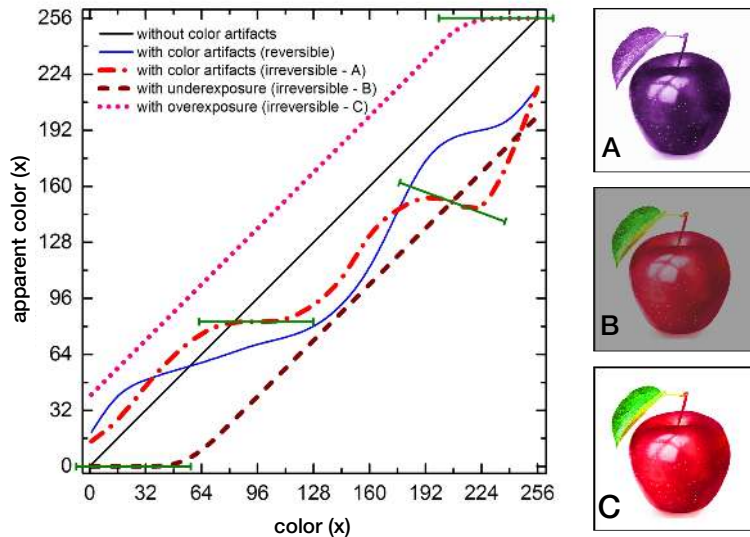


Figure 1.2: Simplified 1D representation of the color reproduction problem in reversible and in non-reversible conditions. For clarity only one color coordinate has been represented: x stands for R, G, or B, and x' stands for R' , G' , or B' . Object colors (x) appear to be different (x') after being acquired by digital means. In some situations, these alterations cannot be removed, because the transformation from x' to x is not single-valued (the critical color ranges where this problem occurs are highly lighted with the green marker).

Third, using smartphones to acquire image data often presents computer vision challenges. On one hand, authors preferred to enclose the smartphone device in a fixed setup [16; 17]. On the other hand, there exists a consolidated knowledge on computer vision techniques, which it could be applied to readout colorimetric sensors with handheld smartphones [18].

Computer vision often seeks to extract features from the captured scene to be able to perform the desired operations on the image, such as: projective corrections, color readouts, etc. These features are objects with unique contour metrics or shapes, like the *ArUco codes* (see Figure 1.3) used in augmented reality technology [19].

Moreover, 2D barcode technology is based upon this principle: encode data into machine-readable patterns which are easy to extract from a scene thanks to their uniqueness. QR Codes are the most known 2D barcodes [20].

This is why, other authors had proposed solutions to print QR Codes with using colorimetric indicators as their printing ink. Rendering QR Codes which change its color when the target substance is detected [21]. Even, using colorimetric dyes as actuators, where authors enhanced the QR Code capacity instead of sensing any material [22].

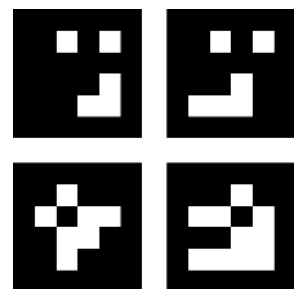


Figure 1.3: Four examples of ArUco codes. These codes present certain feature uniqueness (rotation, non-symmetry, etc.), which enables easy location and identification on a scene.

Altogether, the presented solutions did not fully resolve what GasApp needed: an integrated, disposable, cost-effective machine-readable pattern to allocate colorimetric environmental sensors. The state-of-the-art research presented partial solutions, i.e. the colorimetric indicator was tackled, but there was not a proposal on how to perform automated readouts. Or, the sensor was arranged in a QR Code layout, but color correction was not tackled. Or, the color calibration problem was approached, but any of the other two problems were tackled. Etc.

To solve those challenges, we proposed the creation of an integrated machine-readable pattern based on QR Codes, which would embed both the color correction patches and the colorimetric indicators patches. And, those embedding ought to be back-compatible with the QR Code standard, to maintain the data storage capabilities of QR Codes for traceability applications [20]. A representation of this idea is portrayed in Figure 1.4.



Figure 1.4: Our thesis proposal to create machine-readable patterns that can accommodate colorimetric sensors and color charts, alongside with the digital information of the QR Code.

The novelty of the idea led us to submit a patent application in 2018, which was granted worldwide in 2019, and now is being evaluated in national phases [23]. Moreover, we launched *ColorSensing*, a spin-off company from *Universitat de Barcelona* to develop further the technology in industrial applications [24].

The strength points of the back-compatible proposal were:

- the use of pre-existent computer vision algorithms to locate QR Codes, freeing the designed pattern of redundant computer vision features, as those 'circles' seen outside the GasApp card (Figure 1.4), which are redundant with the finder patterns of the QR Code (corners of the QR Code);
- the reduced scale represented by a QR Code, Figure 1.4 is rescaled for displaying purposes, but the original GasApp proposal was to set to a business card size (3.5×2.0 inches), while our QR Code proposal smaller (1×1 inch);

- reducing the barrier between the new technology and the final users, as the back-compatible proposal maintains the mainstream standard of the QR Codes, one could simply encode a desired URL in the QR Code data alongside with the color information and always be able to redirect the final user to a download link of the proper reader which enables the color readout;
- and, the capacity to increase the color references embedded in a color chart, while also reducing the global size of the chart, e.g. the usual size of a commercial ColorChecker is about 11×8.5 inches, and it encodes 24 color patches, using modern machine-readable standard such as QR Codes as an encoding base enables a systematic path increase the capacity per surface unit, and subsequently according to color correction theory, leading to a better color corrections having more color references.

1.1 Objectives

All in all, the thesis proposes a new approach to automate color correction for colorimetry applications using barcodes, namely Color QR Codes featuring colorimetric indicators. Let us enumerate the objectives of the thesis:

- I **Capture machine-readable patterns placed on top of challenging surfaces**, which are captured with handheld smartphones. These surfaces can be non-rigid surfaces presented in real-world applications, such as: bottles, packaging, food, etc.
- II **Define a back-compatible QR Code modification to extend QR Codes to act as color charts**, which back-compatibility ensures that the digital data of the QR Code remains readable during the whole modification process.
- III **Achieve image consistency using color charts for any camera or light setup, enabling colorimetric applications to yield quantitative results**, and doing so by specifying a color correction method that takes into account arbitrary modifications in the capture scene, such as: light source, smartphone device, etc.
- IV **Demonstrate a specific application of the technology based on colorimetric indicators**, where the accumulated results from objectives I to III are applied.

1.2 Thesis structure

In this thesis, we tackled the above-mentioned objectives. Prior to that, we introduced a chapter to present the backgrounds and methods applied to this thesis. Then, we presented four thematic chapters related to each one of the objectives. These chapters were prepared with a coherent structure: a brief introduction, a proposal, an experimental details section, the results presentation and the conclusion discussion. Later, a general conclusion chapter was added to close the thesis. Let us briefly present the content of each thematic chapter.

First, in chapter 3 we reviewed the state-of-the-art method to extract QR Codes from different surfaces. Then, we focused on a novel approach to readout *QR Codes on challenging surfaces*, such as those found in food packages, such as cylinders or any non-rigid plastic [25; 26].

Second, in chapter 4 we introduced the main proposal of the thesis, the *back-compatible Color QR Codes* [23]. Here, we also introduced not only the machine-readable pattern proposal but also we benchmarked the different possible approaches to embed colors in a QR Code by taking into account its data encoding (which colors are to be embedded where, etc.) and how it affected the QR Code final readability.

Third, in chapter 5 we sought for a unified framework of color corrections based upon affine [14], polynomial [27; 28], root-polynomial [28] and thin-plate splines [15] color corrections. Within that framework, we presented our new proposal for an *improved TPS_{3D} method to achieve image consistency*.

Finally, in chapter 6 we surveyed the different color sensors where we already used partial approaches to our solution [29; 30]. Then, we also studied how to apply our proposal to an actual *application of a colorimetric indicator* that sensed CO₂ levels [31] in modified atmosphere packaging [32].

Chapter 2. Background and methods

2.1 The image consistency problem

Color reproduction is one of the most studied problems in the audio-visual industry, that is present in our daily lives, long before today's smartphones, when color was introduced to the cinema, also with color analog cameras and color home TVs [11]. In the past years, reproducing and measuring color has also become an important challenge for other industries such as health care, food manufacturing and environmental sensing. Regarding health care, dermatology is one of the main fields where color measurement is a strategic problem, from measuring skin-tones to avoid dataset bias [33] to medical image analysis to retrieve skin lesions [34; 35]. In food manufacturing, color is used as an indicator to solve quality control and freshness problems [36; 37; 38]. As for environmental sensing [4], colorimetric indicators are widely spread to act as humidity [39], temperature [40] and gas sensors [41; 42].

In this section, we focus on image consistency, a reduced problem from color reproduction. While *color reproduction* aims at matching the color of a given object when reproduced in another device as an image (e.g. a painting, a printed photo, a digital photo on a screen, etc.), *image consistency* is the problem of taking different images of the same object in different illumination conditions and with different capturing devices, to finally obtain the same apparent colors for this object. In this problem, the apparent colors of an object do not need to match its "real" spectral color, they only rather have to be similar in each instance captured in different scenarios. In other words, all instances should match the first or the best capture, and not the real-life color. Therefore, image consistency is the actual problem to solve in the before-mentioned applications, in which it is more important to compare acquired images between them, so that consistent conclusions can be drawn with all instances, than comparing them to an actual reflectance spectrum.

2.1.1 Color reproduction

Color reproduction is the problem of matching the reflectance of an object with an image of this object [11]. This can be seen in Figure 2.1.a, where an object (an apple) which has a reflectance $R(\lambda)$, is illuminated by a light source $I(\lambda)$ and captured by a camera with a sensor response $D(\lambda)$. In fact, digital cameras contain more than one sensor targeting different ranges of the visible spectrum, commonly they hold 3 types of sensors centered in red, green and blue colors [11].

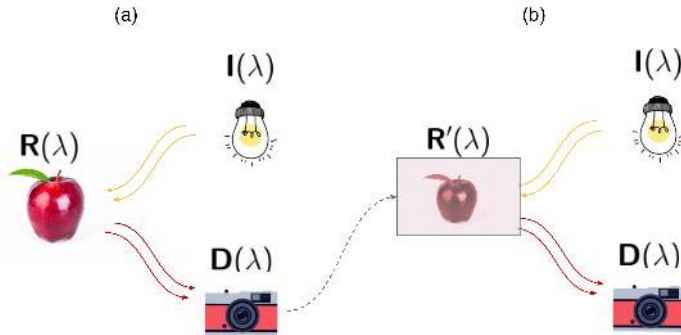


Figure 2.1: The color reproduction problem is represented: (a) a certain light source ($I(\lambda)$) illuminates a certain object with a certain reflectance ($R(\lambda)$), this scene is captured by a certain camera with its sensor response ($D(\lambda)$) and (b) the reproduced image of the object ($R'(\lambda)$) is then illuminated and captured again.

In general, the signal acquired by one of the sensors inside the camera device can be modeled as [43]:

$$S_k \propto \int_{-\infty}^{\infty} I(\lambda) R(\lambda) D_k(\lambda) d\lambda \quad (2.1)$$

where $k \in \{1, \dots, N\}$ are the channels of the camera, N is the total number of channels and λ are the visible spectra wavelengths. Then, Figure 2.1.b portrays the color reproduction of the object, where now a new reflectance will be recreated and captured with the same conditions. Since our image is a printed image, the new reflectance will be:

$$R'(\lambda) = \sum_{i=0}^M f_i(S_1, \dots, S_N) \cdot R_i(\lambda) \quad (2.2)$$

where $R_i(\lambda)$ are the reflectance spectra of the M reproduction inks, which will be printed as a function of the acquired S_k channel contributions. The color reproduction problem now can be written as the minimization problem to the distance of both reflectances:

$$\|R'(\lambda) - R(\lambda)\| \rightarrow 0 \quad (2.3)$$

for each wavelength, for each illumination and for each sensor.

The same formulation could be written when displaying images on a screen by changing $R(\lambda)$ for $I(\lambda)$.

Color reproduction is a wide open problem, and with each step towards its general solution, the goal of achieving image consistency when acquiring image datasets is nearer. Since color reproduction solutions aim at attaining better acquisition devices and better reproduction systems, the need for solving the image consistency problem will eventually disappear. But this is not yet the case.

2.1.2 Image consistency

However, the image consistency problem is far simpler than the color reproduction problem. The image consistency problem can be seen as the problem to match the acquired signal of any camera, under any illumination for a certain object. This can be seen in Figure 2.2.a: an object (an apple), which has a reflectance $R(\lambda)$, is illuminated by a light source $I(\lambda)$ and it is captured by a camera with a sensor response $D(\lambda)$. Now, in Figure 2.2.b, the object is not reproduced but exposed again over different illumination conditions $I'(\lambda)$ and captured by a different camera $D'(\lambda)$.

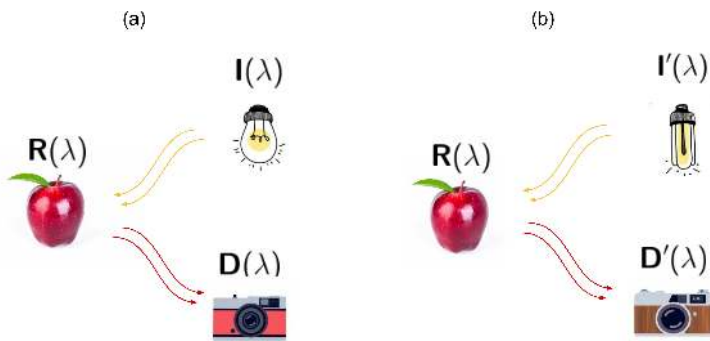


Figure 2.2: The imaging consistency problem is represented: (a) a certain light source ($I(\lambda)$) illuminates a certain object with a certain reflectance ($R(\lambda)$), this scene is captured by a certain camera with its sensor response ($D(\lambda)$) and (b) the same object is now illuminated by another light source ($I'(\lambda)$) and captured by another camera ($D'(\lambda)$).

Under its respective illumination, each camera will follow Equation 2.1 providing three different S_k channels. Considering we can write a vector signal from the camera as:

$$\mathbf{s} = (S_1, \dots, S_N), \quad (2.4)$$

the image consistency problem can be written as the minimization problem to the distance between acquired signals:

$$\|\mathbf{s}' - \mathbf{s}\| \rightarrow 0 \quad (2.5)$$

for each camera, for each illumination for a given object.

The image consistency problem is easier to solve, as we have changed the problem from working with continuous spectral distributions (see Equation 2.3) to N-dimensional vector spaces (see Equation 2.5). These spaces are usually called *color spaces*, and the mappings between those spaces are usually called *color conversions*. Deformations or corrections inside a given color space are often referred to as *color corrections*. In this thesis, we will be using RGB images from digital cameras. Thus, we will work with *device-dependent color spaces*.

This means that the mappings will be performed between RGB spaces. Then, we can rewrite the color vector definition for RGB colors following Equation 2.4 as:

$$\mathbf{s} = (r, g, b), \quad \mathbf{s} \in \mathbb{R}^3, \quad (2.6)$$

where \mathbb{R}^3 represents here a generic 3-dimensional RGB space. In subsection 2.3.1, we detail how color spaces are defined according to their bit resolution and color channels.

2.1.3 Color charts

The traditional approach to achieve a general purpose color correction is the use of color rendition charts, introduced by C.S. McCamy et al. in 1976 [13] (see Figure 2.3). Color charts are machine-readable patterns placed in a scene that embed reference patches of a known color, where in order to solve the problem, several color references are placed in a scene to be captured and then used in a post-capture color correction process.

These color correction processes involve algorithms to map the color references seen in the chart to their predefined nominal colors. This local color mapping is then extrapolated and applied to the whole image. There exists many ways to correct the color of images to achieve consistency.

The most extended way to do so is to search for *device-independent color spaces* (i.e. CIE Lab, CIE XYZ, etc.) [11]. But in the past decade, there have appeared solutions that involve direct corrections between *device-dependent color spaces* without the need to pass through device-independent ones.

The most simple color correction technique is the white balance, that only involves one color reference [44]. A white reference inside the image is to be mapped to a desired white color and then the entire image is transformed using a scalar transformation. Beyond that, other techniques that use more than one color reference can be found elsewhere, using affine [44], using polynomial [27; 28], root-polynomial [28] or thin-plate splines [15] transforms.



Figure 2.3: A ColorChecker chart. The first row shows a set of six “natural colors”; the second one shows a set of “miscellaneous colors”; the third, primary and secondary colors; and the last row, a gray scale gradient. This set of colors samples the RGB space in a limited way, but it is convenient to carry out a few color corrections manually.

It is safe to say that, in most of these post-capture color correction techniques, increasing the number and quality of the color references offers a systematic path towards better color calibration results. This strategy however, comes along with more image area dedicated to accommodate these additional color references and therefore, a compromise must be found.

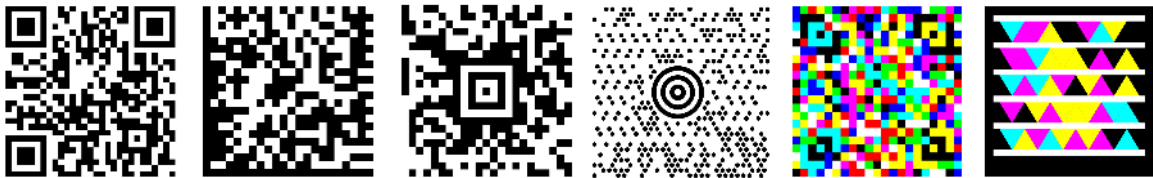
This led X-Rite (a Pantone subsidiary company), to introduce improved versions of the ColorChecker, like the ColorChecker Passport Photo 2[®] kit (see Figure 4.a). Also in this direction, Pantone presented in 2020 an improved color chart called Pantone Color Match Card[®] (see Figure 4.b), based on the Aruco codes introduced by S. Garrido-Jurado et al. in 2015 [19] to facilitate the location of a relatively large number of colors. Still, the size of these color charts is too big for certain applications with size constraints (e.g. smart tags for packaging [45; 30]).



Figure 2.4: Previous state-of-the-art color correction charts from Pantone and X-Rite. (a) The X-Rite ColorChecker Passport Photo 2[®] kit. (b) The Pantone Color Match Card[®].

2.2 2D Barcodes: the Quick-Response Code

Quick-Response Codes, popularized as QR Codes, are 2D barcodes introduced in 1994 by Denso Wave [20], which aimed at replacing traditional 1D barcodes in the logistic processes of this company. However, the use of QR Codes has escalated in many ways and are now present in manifold industries: from manufacturing to marketing and publicity, becoming a part of the mainstream culture. In all these applications, QR Codes are either printed or displayed and later acquired by a reading device, which normally includes a digital camera or barcode scanner. Also, there has been an explosion of 2D barcode standards [46; 47; 48; 49; 50] (see Figure 2.5).



The process of encoding and decoding a QR Code could be considered as a form of communication through a visual channel (see Figure 2.6): a certain message is created, then split into message blocks, these blocks are encoded in a binary format, and finally encoded in a 2D array. This 2D binary array is an image that is transmitted through a visual channel (printed, observed under different illuminations and environments, acquired as a digital image, located, resampled, etc.). On the decoder side, the binary data of the 2D binary array is retrieved, the binary stream is decoded, and finally the original message is obtained.

From the standpoint of a visual communication channel, many authors before explored the data transmission capabilities of the QR Codes, especially as steganographic message carriers (data is encoded in a QR Code, then encoded in an image) due to their robust error correction algorithm [51; 52].

Figure 2.5: Different 2D barcode standards. From left to right: a QR Code, a DataMatrix, an Aztec Code, a MaxiCode, a JAB Code and a HCC Barcode.

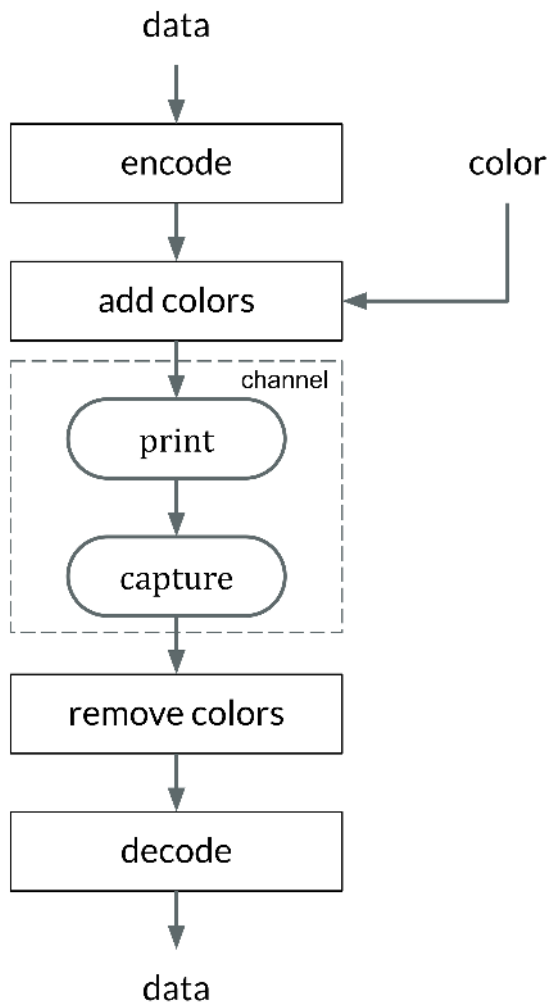


Figure 2.6: Block diagram for a general encoding-decoding process of a QR Code which features the embedding of a color layer. This color layer could be used for a wide range of applications, such as placing a brand logo inside a QR Code. The process can be seen as a global encoding process (digital encode and color encode), followed by a channel (print and capture) and a global decoding process (remove colors and decode digital information).

2.2.1 Scalability

Many 2D barcode standards allow modulating the amount of data encoded in the barcode. For example, the QR Code standard implements different barcode versions from version 1 to version 40. Each version increases the edges of the QR Code by 4 *modules*, from the starting 21×21 (v1) modules up to 144×144 modules (v40) [20].

For each version, the location of every computer vision feature is fully specified in the standard (see Figure 2.7), in subsection 2.2.3 we will focus on these features. Some other 2D barcode standards are flexible enough to cope with different shapes, such as rectangles in the DataMatrix codes (see Figure 2.8), which can be easier to adapt to different substrates or physical objects [46].



Figure 2.7: Some examples of QR Code versions. From left to right: Micro QR-Code (version M3), version 3 QR Code, and version 10 QR Code. Each of them can store up to 7, 42, 213 bytes, respectively, using a 15% of error correction capacity.



Figure 2.8: Some examples of DataMatrix codes. From left to right: rectangular DataMatrix code, square DataMatrix code and four square DataMatrix combined. Each of them can store up to 14, 28, 202 bytes, respectively, using approximately a 20% of error correction capacity.

These different possible geometries must be considered when adding colors to a 2D barcode. In the case of the QR Codes and DataMatrix codes, the larger versions are built by replicating a basic squared block. Therefore, the set of color references could be replicated in each one of these blocks, to gain in redundancy and in a more local color correction. Alternatively, different sets of color references could be used in each periodic block to facilitate a more thorough color correction based on a larger set of color references.

Regarding this size and shape modularity in 2D barcode encoding, there exist a critical relationship between the physical size of the modules and the pixels in a captured image. This is a classic sampling phenomena [53], for a fixed physical barcode size and a fixed capture (same pixels) as the version of the QR Code increases the amount of modules in a given space increases.

Thus, the apparent size of the module in the captured image decreases, when this size is near a bunch of pixels we start to see aliasing problems [54]. In turn, this problem leads to a point that QR Codes cannot be fully recognized by the QR-Code decoding algorithm. This is even more important if we substitute these black and white modules with colors, where the error in finding the right reference area may lead to huge errors in the color correction. Therefore, this sampling problem will accompany the implementation of our proposal taking into account the size of the final QR Code depending on the application field and the typical resolution of the cameras used in those applications.

2.2.2 Data encoding in QR Codes

The QR Code standard presents a complex encoding layout (see Figure 2.9). Encoding a message into a QR Code form implies several steps.

First, the message is encoded as binary data and split into various bytes, namely *data blocks*, QR Codes can support different data types, the binary encoding for those data types will be different in order to maximize the amount of data to encode in the barcode (see Table 2.1).

Second, additional *error correction blocks* are computed based on the Reed-Solomon error correction theory [55]. Third, the minimal version of the QR Code is determined, which defines the size of the 2D array to “print” the error correction and data blocks, as a binary image. When this is done, the space reserved for the error correction blocks is larger than the space reserved for the data blocks (see Figure 2.10).

Finally, a binary mask is implemented in order to randomize as maximum as possible the QR Code encoding [20].

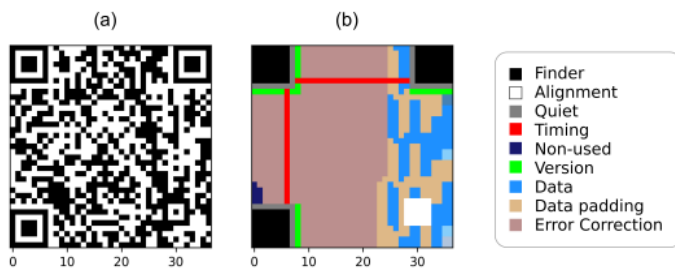


Figure 2.9: QR Code encoding defines a complex layout with several patterns to be considered, some of them are non-variant patterns found in each QR Code, others may appear depending on the size of the QR Code, and area related to the data changes for each encoding process. (a) A QR Code with high error correction level and version 5. (b) The complex pattern structure of the pattern.

ECC level	Bits	Numeric	Alphanumeric	Binary	Kanji
Version 1					
L	152	41	25	17	10
M	128	34	20	14	8
Q	104	27	16	11	7
H	72	17	10	7	4
Version 2					
L	272	77	47	32	20
M	224	63	38	26	16
Q	176	48	29	20	12
H	128	34	20	14	8
Version 39					
L	22,496	6,743	4,087	2,809	1,729
M	17,728	5,313	3,22	2,213	1,362
Q	12,656	3,791	2,298	1,579	972
H	9,776	2,927	1,774	1,219	750
Version 40					
L	23,648	7,089	4,296	2,953	1,817
M	18,672	5,596	3,391	2,331	1,435
Q	13,328	3,993	2,42	1,663	1,024
H	10,208	3,057	1,852	1,273	784

Table 2.1: A summary of QR Code data encoding capacity is shown. The total capacity for each configuration is expressed in *symbol capacity*. Columns are ordered left to right from higher to lower capacity.

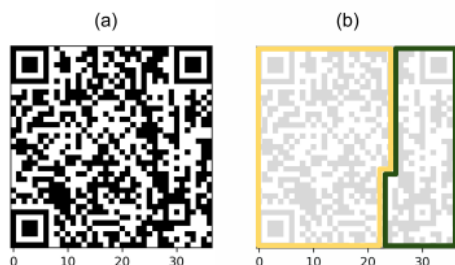


Figure 2.10: QR Code simplified areas corresponding to the encode process. (a) A QR Code with high error correction level and version 5. (b) Simplified view of the QR patterns, yellow frame corresponds to the "error correction" area and dark green frame corresponds to the "data" area.

During the generation of a QR Code, the level of error correction can be selected, from high to low capabilities: H (30 %), Q (25%), M (15%) and L (7%). This should be understood as the maximum number of error bits that a certain barcode can support (maximum Bit Error Ratio, detailed in chapter 4). Notice the error correction capability is independent of the version of the QR Code. However, both combined define the maximum data storage capacity of the QR Code, for a fixed version, higher error correction implies a reduction of the data storage capacity of the QR Code.

This error correction feature is indirectly responsible for the popularity of QR Codes, since it makes them extremely robust while allowing for a large amount of pixel tampering to accommodate aesthetic features, like allocating brand logos inside the barcode [56; 57] (see Figure 2.11 and Figure 2.12). In this thesis, we will take advantage of the encoding features of QR Codes, such as error correction to embed reference colors inside a QR Code.



Figure 2.11: Different examples of *Halftone QR Codes*, introduced by HK. Chu et al. [56]. These QR Codes exploit the error correction features of the QR Code to achieve back-compatible QR Codes with apparent grayscale–halftone– colors.

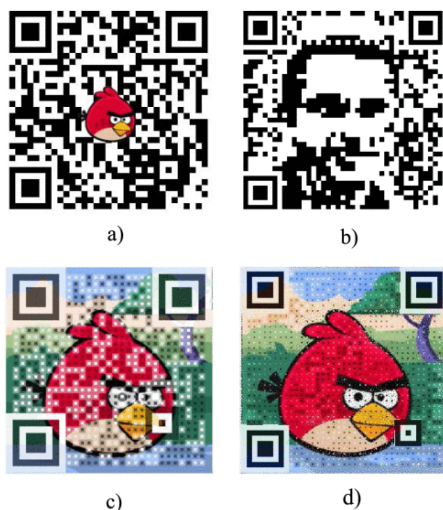
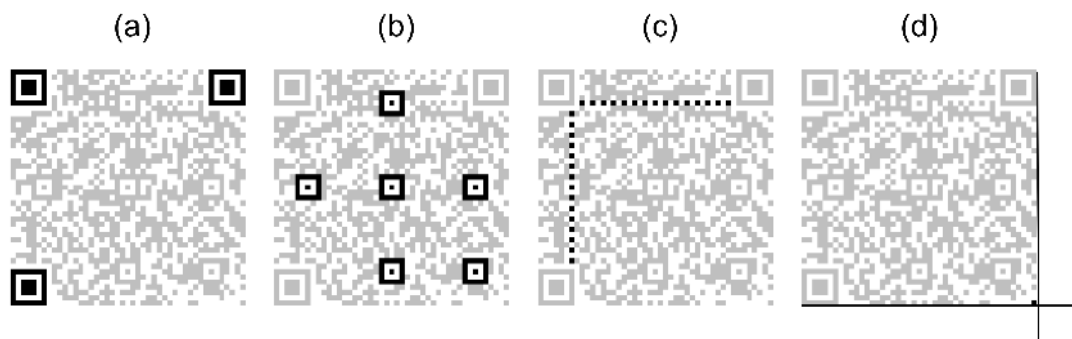


Figure 2.12: Original figure from Garateguy et al. [57], different QR Codes with color art are shown: (a) a QR Code with a logo overlaid; (b) a *QArt Code* [58], (c) a *Visual QR Code*; and (d) Garateguy et al. proposal.

2.2.3 Computer vision features of QR Codes

Besides the data encoding introduced before, a QR Code embeds computer vision features alongside with the encoded digital data. These features play a key role when applying computer vision transformations to the acquired images containing QR codes. Usually, they are extracted to establish a correspondence between their apparent positions in the captured image plane and those in the underlying 3D surface topography. The main features we focus on this thesis are:

- **Finder patterns** are the corners of the QR Code, it has 3 of them to break symmetry and orient the QR in a scene (see Figure 2.13.a).
- **Alignment patterns** are placed inside the QR Code to help in the correction of noncoplanar deformations (see Figure 2.13.b).
- **Timing patterns** are located alongside two borders of the QR Code, between a pair of finder patterns, to help in the correction of coplanar deformations (see Figure 2.13.c).
- The **fourth corner** is the one corner not marked with a finder pattern. It can be found as the crosspoint of the straight extensions of the outermost edges of two finder patterns (see Figure 2.13.d). It is useful in linear, coplanar and noncoplanar deformations.



These features are easy to extract due to their spatial properties. They are well-defined as they do not depend on the version of the QR Code, nor the data encoding. The lateral size for a finder pattern is always 7 modules. For an alignment pattern, 5 modules. And timing patterns grow along with each version, but their period is always 2 modules (one black, one white).

Finder patterns implement a sequence of modules along both axes that follows: 1 black, 1 white, 3 black, 1 white and 1 black, often written as a 1:1:3:1:1 relation (see Figure 2.14). Alignment patterns implement a sequence of modules along both axes that follows: 1 black, 1 white, 1 black, 1 white and 1 black, a 1:1:1:1:1 relation (see Figure 2.15).

Figure 2.13: Computer vision patterns featured in a QR Code. (a) Three finder or position patterns, (b) six alignment patterns, (c) two timing patterns and (d) the fourth corner that can be inferred from the external edges of the finder patterns.

Thus, the relation between white and black pixels provides a path to use pattern recognition techniques to extract these features, as these relations are invariant to perspective transformations. Moreover, these linear relations can be expressed as squared relations, and are still invariant under perspective transformations. This is specially useful when using extraction algorithms based upon contour recognition [18; 59], for finder patterns the relation becomes $7^2:5^2:3^2$ (see Figure 2.14); and for alignment patterns, $5^2:3^2:1^2$ (see Figure 2.15).

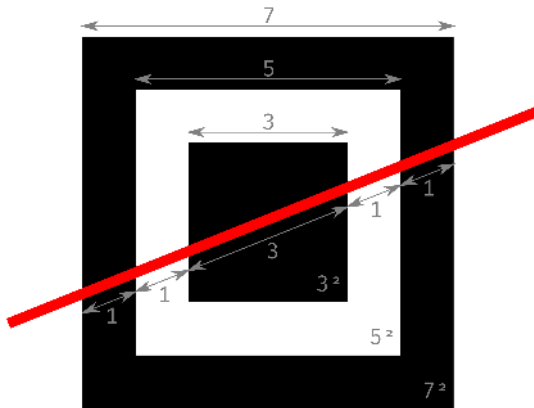


Figure 2.14: Finder pattern definition in terms of modules. Finder pattern measures always 7×7 modules. If scanned with a line barcode scanner the $1:1:3:1:1$ ratio is maintained no matter the direction of the scanner. If scanned using contour extraction the aspect ratio $7^2:5^2:3^2$ is maintained as well if the QR Code is captured within a projective scene (i.e. a handheld smartphone).

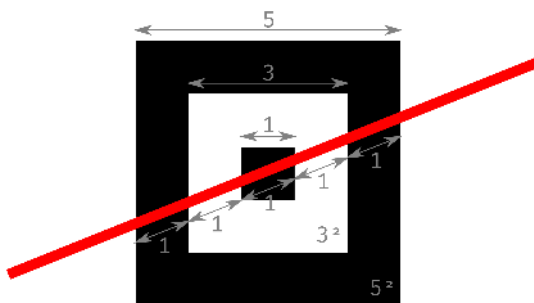


Figure 2.15: Alignment pattern definition in terms of modules. Alignment pattern measures always 5×5 modules. If scanned with a line barcode scanner the $1:1:1:1:1$ ratio is maintained no matter the direction of the scanner. If scanned using contour extraction the aspect ratio $5^2:3^2:1^2$ is maintained as well if the QR Code is captured within a projective scene (i.e. a handheld smartphone).

2.2.4 Readout of QR Codes

Let us explore a common pipeline towards QR Code readout. First, consider a QR Code captured from a certain point-of-view in a flat surface which is almost coplanar to the capture device (e.g. a box in a production line). Note that more complex applications, such as bottles [60], all sorts of food packaging [61], etc., which are key to this thesis, are tackled down in chapter 3.

Due to perspective, the squared shape of the QR Code will be somehow deformed following some sort of projective transformation (see Figure 2.16.a). Then, in order to find the QR Code itself within the image field, the three finder patterns are extracted applying contour recognition algorithms based on edge detection [18; 59] (see Figure 2.16.b). As explained in subsection 2.2.3, each finder pattern candidate must hold a very specific set of area relationships, no matter how they are projected if the projection is linear. The contours that fulfill this area relationship are labeled as candidates finder patterns (see Figure 2.16.c).



Figure 2.16: The QR Code contour detection method. a) A QR Code from a certain perspective. b) All the contours detected in the image. c) The location of the position patterns following the area rule. Their respective centers of mass are indicated.

Second, the orientation of the QR Code must be recognized, as in a general situation, the QR Code captured in an image can take any orientation (i.e. rotation). The above-mentioned three candidate finder patterns are used to figure out the orientation of the barcode. To do so, we should bear in mind that one of these corners will correspond to the top-left one and the other two will be the end points of the opposite diagonal (see Figure 2.17.a). By computing the distances between the three candidate finder pattern centers and comparing them we can find which distance corresponds to the diagonal and assign the role of each pattern in the QR Code. The sign of the slope of the diagonal m and the sign of the distance to the third point s are computed and analyzed to solve the final assignment of the patterns. The four possible combinations result in 4 possible different orientations: north, east, south, west (see Figure 2.17.b). Once the orientation is found, the three corner candidates are labeled following the sequence L, M, N .

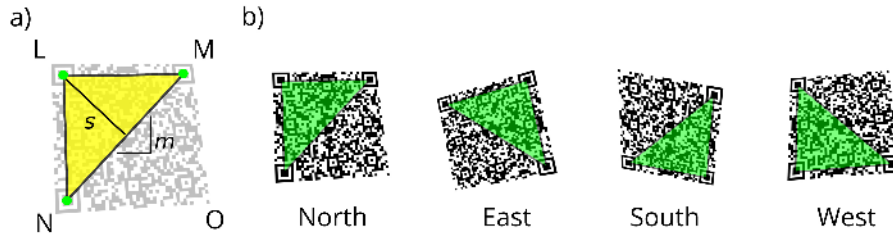


Figure 2.17: The different orientations of a QR Code are shown. (a) Representation of the slope of the diagonal connecting the corners m and the diagonal segment linked to the top-left corner s . (b) The four possible orientations of a QR-Code.

Third, a projection correction is performed to retrieve the QR Code from the scene. The finder patterns can then be used to correct the projection deformation of the image in the QR Code region. If the deformation is purely affine, e.g. a flat surface laying coplanar to the reader device, we can perform the correction with these three points. But, if a more general deformation is presented, e.g. handheld capture in a perspective plane, one need at least one additional point to carry out such transformation: the remaining fourth corner O (see Figure 2.17.a). As the edges around the previous corners were previously determined (see Figure 2.18.a), the fourth corner O is localized using the crossing points of two straight lines from corners M and N (see Figure 2.18.b). With this set of 4 points, a projective transformation that corrects the perspective effect on the QR-Code is carried out (see Figure 2.18.c).

Moreover, notice the calculation of the fourth corner O can accumulate the numerical error of the previous steps. This might lead to inaccurate results in the bottom-right corner of the recovered code (see Figure 2.18.c) and, in some cases, to a poor perspective correction. This effect is especially strong in low resolution captures, where the modules of the QR Code measure a few pixels. In order to solve this issue, the alignment patterns are localized (see Figure 2.18.d) in a more restricted and accurate contour search around the bottom-right quarter of the QR Code (see Figure 2.18.e). With this better estimation of a grid of reference points of known (i.e. tabulated) positions a second projective transformation is carried out (see Figure 2.18.f). Normally, having more reference points than strictly needed to compute projective transformations is not a problem thanks to the introduction of maximum likelihood estimation (MLE) solvers for the projection fitting [62].

Finally, the QR Code readout is performed, this means the QR Code is down-sampled to a resolution where each of the modules occupies exactly one pixel. After this, the data is extracted following a reverse process of the encoding: the data blocks are interpreted as binary data, also the error correction blocks. The Reed-Solomon technique to resolve errors is applied, and the original data is retrieved.

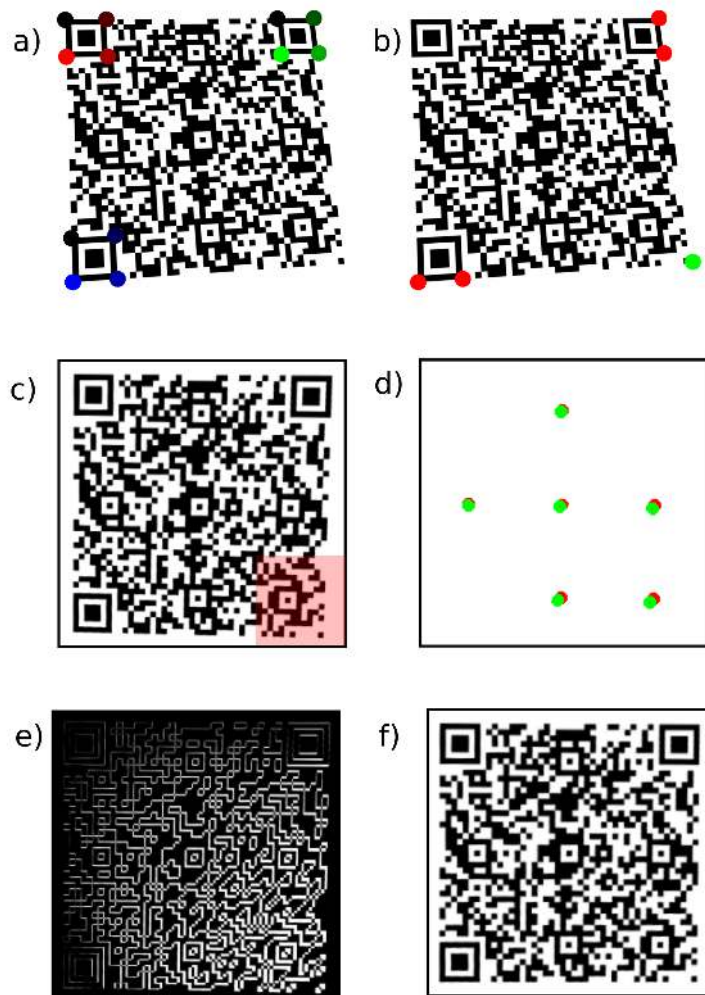


Figure 2.18: The QR Code projective correction steps. a) The orientation is deduced from the centers of the 3 finder patterns L , M , N . In this step, their contour corners are found. b) The fourth corner O is found, based on the previous three corners. c) A first projective transformation is carried out, but still subject to significant error shifts around the bottom-right corner; d) The alignment patterns are localized in a restricted contour search. The centers of the alignment patterns (shifted centers after the first projective correction) (green) and the reference centers are both found (red). e) The error committed at this stage is shown by subtraction of the images. f) Finally, a second projective transformation recovers the final QR Code image, based on the reference, tabulated, positions of the alignment patterns.

2.3 Data representation

2.3.1 Color spaces

In section 2.1 we introduced the image consistency problem alongside with a simplified description of the reflectance model (see Figure 2.19):

$$S_k \propto \int_{-\infty}^{\infty} I(\lambda) R(\lambda) D_k(\lambda) d\lambda \quad (2.7)$$

where a certain light source, $I(\lambda)$, illuminates a certain object with a certain reflectance, $R(\lambda)$, this scene is captured by a sensor with its response, $D_k(\lambda)$. And, S_k represents the signal captured by this sensor. This model specifically links the definition of color to the sensor response, not only to the wavelength distribution of the reflected light. Thus, our color definition depends on the observer.

Let the sensor $D_k(\lambda)$ be the human eye, this is model becomes the well-known tristimulus model of the human eye. In the tristimulus model, a *standard observer* is defined from studying the human vision. This was first studied in 1931 by the International Commission of Illumination, which defined the CIE 1931 RGB and CIE 1931 XYZ color spaces [63; 64]. Since then, the model has been revisited many times defining new color spaces: in 1960 [65], in 1964 [66], in 1976 [67] and so on [68].

Commonly, color spaces referred to a *standard observer* are called *device-independent color spaces*. As explained before, we are going to use images which are captured by digital cameras. These images will use *device-dependent color spaces*, despite the efforts of their manufacturers to solve the color reproduction problem, as they try to match the camera sensor to the tristimulus model of the human eye [69]. Let a color \mathbf{s} be defined by the components of the camera sensor:

$$\mathbf{s} = (S_r, S_g, S_b) \quad (2.8)$$

where S_r , S_g and S_b are the responses of the three sensors of the camera for the *red*, *green* and *blue* channels, respectively. Cameras do imitate the human tristimulus vision system by placing sensors in the wavelength bands representing those where human eyes have more sensitivity.

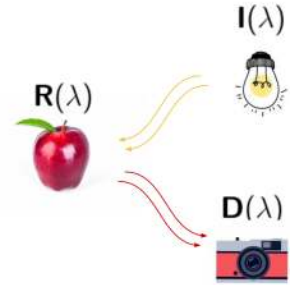


Figure 2.19: A reduced representation of the reflectance model. For more details see Figure 2.1.

Note that \mathbf{s} is defined as a vector in Equation 2.8. Although, its definition lacks the specification of its vector space:

$$\mathbf{s} = (r, g, b) \in \mathbb{R}^3 \quad (2.9)$$

where r, g, b is a simplified notation of the channels of the color, and \mathbb{R}^3 is a generic *RGB color space*. As digital cameras store digital information in a finite discrete representation, \mathbb{R}^3 should become $\mathbb{N}_{[0,255]}^3$ for 8-bit images (see Figure 2.20). This discretization process of the measured signal in the camera sensor is a well-known phenomenon in signal-processing, it is called *quantization* [70]. All to all, we can write some common color spaces in this notation:

- $\mathbb{N}_{[0,255]}$ is the grayscale color space of 8-bit images.
- $\mathbb{N}_{[0,255]}^3$ is the RGB color space of 24-bit images (8-bits/channel).
- $\mathbb{N}_{[0,4096]}^3$ is the RGB color space of 36-bit images (12-bits/channel).
- $\mathbb{N}_{[0,65536]}^3$ is the RGB color space of 48-bit images (16-bits/channel).
- $\mathbb{N}_{[0,255]}^4$ is the CMYK color space of 32-bit images (8-bits /channel).
- $\mathbb{R}_{[0,1]}^3$ is the RGB color space of a normalized image, specially useful when performing computer vision algorithms.

2.3.2 Color transformations

The introduction of color spaces as vector spaces brings all the mathematical framework of geometric transformations. We can now define a *color conversion* as the application between two color spaces.

For example, let f be a color conversion between an RGB and a CMYK space:

$$f : \mathbb{N}_{[0,255]}^3 \rightarrow \mathbb{N}_{[0,255]}^4 \quad (2.10)$$

this color conversion can take any form. In section 2.1, we saw that the reflectance spectra of the image of an object would be a linear combination of the inks reflectance spectra used to reproduce that object. If we recover that expression from Equation 2.2 and combine it with the RGB color space from Equation 2.9, we obtain:

$$R'(\lambda) = \sum_j^{c,m,y,k} f_j(r, g, b) \cdot R_j(\lambda) \quad (2.11)$$

Now, $R'(\lambda)$ is a linear combination of the reflectance spectra of the *cyan, magenta, yellow* and *black* inks. The weights of the combination is the CMYK color derived from the RGB color.

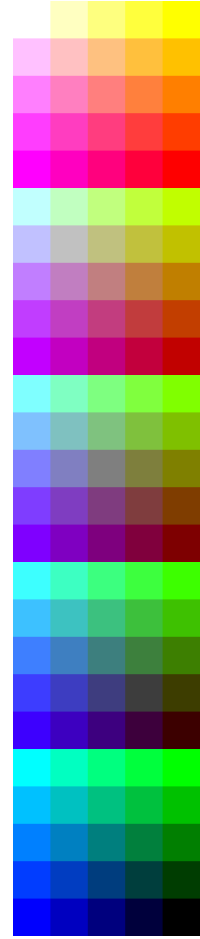


Figure 2.20: 125 colors of an RGB color space. Each channel of the color space has been sampled 5 times. Assuming the space is a 24-bit color space, the values of the sampled colors correspond to: 0, 61, 127, 193 and 255. The combination (255, 255, 255) is the white color and (0, 0, 0) the black color.

In turn, we can express the CMYK color also as a linear combination of the RGB color channels, $f_i(r, g, b)$ is our color correction here, then:

$$R'(\lambda) = \sum_j^{c,m,y,k} \left[\sum_k^{r,g,b} a_{jk} \cdot k \right] \cdot R_j(\lambda) \quad (2.12)$$

Note that we have defined f_i as a linear transformation between the RGB and the CMYK color spaces, doing so is the most common way to perform color transformations between color spaces.

This is the foundation of the ICC Profile standard [71]. Profiling is a common technique when reproducing colors. For example, take Figure 2.20, if the colors are seen displayed on a screen they will show the RGB space of the LED technology of the screen. However, if they have been printed, the actual colors the reader will be looking at will be the linear combination of CMYK inks representing the RGB space, following Equation 2.12. ICC profiling is present in each color printing process.

Alongside with the described example, here below, we present some of the most common color transformations we will use during the development of this thesis, that include *normalization*, *desaturation*, *binarization* and *colorization* transformations.

2.3.2.1 Normalization

Normalization is the process to map a discrete color space with limited resolution ($\mathbb{N}_{[0,255]}$, $\mathbb{N}_{[0,255]}^3$, $\mathbb{N}_{[0,4096]}^3$, ...) to a color space which is limited to a certain range of values, normally from 0 to 1 $\mathbb{R}_{[0,1]}$, but offers theoretically infinite resolution¹. All our computation will take place in such normalized spaces. Formally the normalization process is a mapping that follows:

$$f_{normalize} : \mathbb{N}_{[0, 2^n]}^K \rightarrow \mathbb{R}_{[0,1]}^K \quad (2.13)$$

where K is the number of channels of the color space (i.e. $K = 1$ for grayscale, $K = 3$ for RGB color spaces, etc.) and n is the bit resolution of the color space (i.e. 8, 12, 16, etc.).

Note that a normalization mapping might not be that simple that only implies a division by a constant. For example, an image can be normalized using an exponential law to compensate camera acquisition issues [72; 73].

¹ The infinite resolution that represents \mathbb{R} is not computationally feasible. However, the computational representation of a \mathbb{R} space, a `float` number, handles a higher precision than other former space before normalization.

2.3.2.2 Desaturation

Desaturation is the process to map a color space to a grayscale representation of this color space. Thus, formally this mapping will always be a mapping from a vector field to a scalar field. We will assume the color space has been previously normalized following a mapping (see Equation 2.13). Then:

$$f_{desaturate} : \mathbb{R}_{[0,1]}^K \rightarrow \mathbb{R}_{[0,1]} \quad (2.14)$$

where K is still the number of channel the input color space has. There exist several ways to desaturate color spaces, for example, each CIE standard incorporates different ways to compute their luminance model [64].

2.3.2.3 Binarization

Binarization is the process to map a grayscale color space to a binary color space, this means the color space gets reduced only to a representation of two values. Formally:

$$f_{binarize} : \mathbb{R}_{[0,1]} \rightarrow \mathbb{N}_{[0,1]} \quad (2.15)$$

Normally, these mappings need to define some kind of *threshold* to split the color space representation into two subsets. Thresholds can be as simple as a constant threshold or more complex [74].

2.3.2.4 Colorization

Colorization is the process to map a grayscale color space to a full-featured color space. We can define a colorization as:

$$f_{colorize} : \mathbb{R}_{[0,1]} \rightarrow \mathbb{R}_{[0,1]}^K \quad (2.16)$$

where K is now the number of channel the output color space has. This process is more unusual than the previous mappings presented here. It is often implemented in those algorithms that pursue image restoration [75]. In this work, colorization will be of a special interest in chapter 4.

2.3.3 Images as bitmaps

A digital image is the result of capturing a scene with an array of sensors [11], following Equation 2.7. Take a monochromatic image I , this means we only have one color channel in our color space. This image can be seen as a mapping between a vector field, the 2D plane of the array of sensors, and a scalar field, the intensity of light captured by each sensor:

$$I : \mathbb{R}^2 \rightarrow \mathbb{R} \quad (2.17)$$

where \mathbb{R}^2 is the capture plane of the sensors and \mathbb{R} is a generic grayscale color space. Figure 2.21 shows an example of this: an Airy disk [76] is represented first as an image, where the center of the disk is visualized as a spot; also, the Airy disk is shown to be a function of the space distribution.

Altogether, we can extend Equation 2.17 definition to images that are not grayscale. This means each image can be defined as a mapping from the 2D plane of the array of sensors to a color space, which is in turn also a vector space:

$$I : \mathbb{R}^2 \rightarrow \mathbb{R}^K \quad (2.18)$$

where \mathbb{R}^K is now a vector field also, thus the color space of the image can be RGB, CMYK, etc. Note digital cameras can capture more than the above-mentioned color bands, and there exists a huge field of multi-spectral cameras [77], which is not the focus of our research.

As we pointed out when defining color spaces, digital images are captured using discrete variable color spaces. But this process also affects the spatial domain of the image. The process of discretizing the plane \mathbb{R}^2 is called *sampling*. And, the process of discretizing the illumination data in \mathbb{R} data is called *quantization*. Following this, Equation 2.17 is rewritten as:

$$I : \mathbb{N}_{[0,n]} \times \mathbb{N}_{[0,m]} \rightarrow \mathbb{N}_{[0,255]} \quad (2.19)$$

which represents an 8-bit grayscale² image of size (n, m) . This definition of an image allows us to differentiate the *domain* transformations of the image, i.e. geometrical transformations to the perspective of the image; from the *image* transformations, i.e. color corrections to the color space to the image.

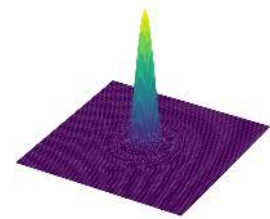
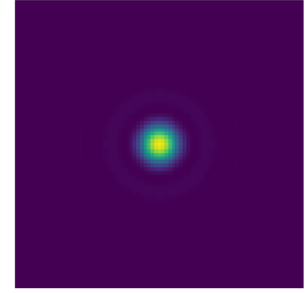


Figure 2.21: An Airy disk is shown as a grayscale image with a color map (top) and as a function (bottom) with the same color map.

² This example uses a grayscale image of 8-bit resolution, however any of the formats specified in the subsection 3.3.2 could be used here.

In chapter 3, when dealing with the extraction of QR Codes from challenging surfaces we used the definition in Equation 2.17 to refer to the capturing plane of the image and how it relates to the underneath surface where the QR Code is placed by projective laws.

In chapter 4 we used the definition of Equation 2.19 to detail our proposal encoding process of colored QR Codes. In this scenario, it is interesting reducing the notation of image definition taking into account images can be seen as matrices. So, Equation 2.19 can be rewritten in a compact form as:

$$I \in [0, 255]^{n \times m} \quad (2.20)$$

where I is now a matrix which exist in a matrix space $[0, 255]^{n \times m}$. This vector space contains both the definition of the spatial coordinates of the image and the color space.

As before, we can use this notation to represent different image examples:

- $I \in [0, 255]^{n \times m}$, is an 8-bit grayscale image with size (n, m) .
- $I \in [0, 255]^{n \times m \times 3}$, is an 8-bit RGB image with size (n, m) .
- $I \in [0, 1]^{n \times m}$, is a float normalized grayscale image with size (n, m) .
- $I \in \{0, 1\}^{n \times m}$, is a binary image with size (n, m) .

Finally, we can redefine the color space transformations (from Equation 2.13 to Equation 2.16) transformations of these image spaces:

- **Normalization:**

$$f_{normalize} : [0, 255]^{n \times m \times 3} \rightarrow [0, 1]^{n \times m \times 3} \quad (2.21)$$

- **Desaturation:**

$$f_{desaturate} : [0, 1]^{n \times m \times 3} \rightarrow [0, 1]^{n \times m} \quad (2.22)$$

- **Binarization:**

$$f_{binarize} : [0, 1]^{n \times m} \rightarrow \{0, 1\}^{n \times m} \quad (2.23)$$

- **Colorization:**

$$f_{colorize} : [0, 1]^{n \times m} \rightarrow [0, 1]^{n \times m \times 3} \quad (2.24)$$

2.4 Computational implementation

In 1990, Guido van Rosum released the first version of Python, an open-source, interpreted, high-level, general-purpose, multi-paradigm (procedural, functional, imperative, object-oriented) programming language [78]. Since then, Python has released three major versions of the language: Python 1 (1990), Python 2 (2000) and Python 3 (2008) [79].

At the time we started to work in the thesis development, Python was one of the popular programming languages both in the academia and in the industry [80]. As Python is an interpreted language, the actual code of Python is executed by the Python Virtual Machine (PVM), this opens the door to create different PVM written with different compiled languages, the official *Python distribution* is based upon a C++ PVM, that is why the mainstream Python distribution is called 'CPython' [81].

CPython allows the user to create *bindings* to C/C++ libraries, this was specially useful for our research. OpenCV is a widely-known tool-kit for computer vision applications, which is written in C++, but presents bindings to other languages like Java, MATLAB or Python [82].

Altogether, *we decided to use Python as our main programming language*. Both achieving rapid script capabilities that Python offers alongside with standard libraries from Python and C++. The research started with the use of Python 3.6 and ended with the use of Python 3.8, due to Python development cycle.

Let us detail the stack of standard libraries used during the development of the thesis:

- **Python environment:** we started using the Anaconda, an open-source Python distribution that contained pre-compiled packages ready to be used, such as OpenCV [83]. We adopted also pyenv, a tool to install Python distributions and manage virtual environments [84]. Later on, we started to use docker technology, light virtual machines to enclose the PVM and our programs [85].
- **Scientific and data:** we adopted the well-known numpy / scipy / matplotlib stack,
 - numpy is a C++ implementation of array representation (MATLAB-like) for Python [86],
 - scipy is a compendium of common mathematical operations fully compatible with NumPy arrays, often SciPy implements bindings to consolidated calculus frameworks written in C++ and Fortran, such as OpenBlas [87],

- `matplotlib` is a 2D graphics environment we used to represent our data [88].

NumPy, SciPy and Matplotlib are the entry point to a huge ecosystem of packages which use them as their core. When processing datasets two main packages were used,

- `pandas` is an abstraction layer to the previous stack, data is organized in spreadsheets (like Excel, Origin Lab, etc.) [89],
 - `xarray` is another abstraction layer to the previous stack, with labeled N-dimensional arrays, `xarray` can be regarded as the N-dimensional generalization of `pandas` [90].
- **Images manipulation:** there is a huge ecosystem regarding image manipulation in Python, previous to computer vision, we adopted some packages to read and manipulate images,
 - `pillow` is the popular fork from the unmaintained Python Imaging Library, we used Pillow specially to manipulate the image color spaces, i.e. profile an image from RGB to be printed in CMYK [91],
 - `imageio` was used as an abstraction layer from Pillow, which uses Pillow and other I/O libraries (such as `rawpy`) to read images and convert them directly to NumPy matrices, we standardized our code to read images using this package instead of using other solutions (`SciPy`, `Matplotlib`, `Pillow`, `OpenCV`, ...) [92],
 - `imgaug` was used to enhance image datasets, by tuning randomly parameters of the image (illumination, contrast, etc.), this is a well-known technique to increase dataset when training computer vision models [93].
 - **Computer vision:** we mainly adopted `OpenCV` as our main framework to perform feature extraction algorithms, affine and perspective corrections and other operations [59]. Despite this, other popular frameworks were used for some applications, such as `scikit-learn` [94], `scikit-image` [95], `keras` [96], etc.
 - **QR Codes:** regarding the encoding of QR Codes we adopted mainly the package `python-qrcode` and use it as a base to create our Color QR Codes [97]; regarding the decoding of the QR Codes, there exists different frameworks we worked with,
 - `zbar` is a light C++ barcode scanner, which decodes QR Codes and other 1D and 2D barcodes [98], among the available Python bindings to this library we chose the `pyzbar` library [99],
 - `zxing` is a Java bar code scanner, similar to `ZBar`, formerly maintained by Google, and it is the core of most of Android QR Code scanners [100], as this library was not written in Python we did not use it in a daily basis, but we kept it as secondary QR Code scanner.

Chapter 3. QR Codes on challenging surfaces

In chapter 2 we have introduced the popular QR codes [20], which have become part of mainstream culture. With the original applications in mind (e.g. a box in a production line), the QR Codes were designed, first, to be placed on top of flat surfaces, second, laying coplanar to the reader device.

But today, users also apply QR Codes to non-planar surfaces like bottles [60], all sorts of food packaging [61] (like meat [101], fish [102] and vegetables [103]), vehicles, handrails, etc. (see Figure 3.1.a). Also, QR Codes can incorporate biomedical [104], environmental [105] and gas [30] sensors. All these applications involve surfaces that pose challenges to their readout, especially when the QR Codes are big enough to show an evident curvature or deformation.

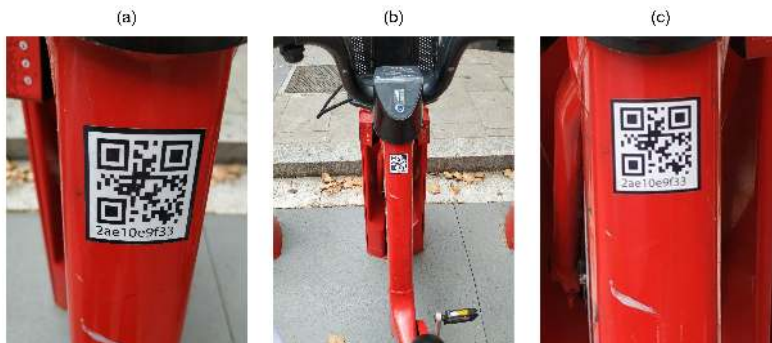


Figure 3.1: An example of an adverse situation, image of a QR Code in a bike-sharing service in Barcelona, where the QR Code is bent over the bike frame. User experience shows that capturing these QR Codes is difficult when approaching the camera to the QR Code due to the bending. (a) An image captured near the QR Code (~ 20 cm), (b) an image captured farther (~ 1 m) and (c) a zoomed version of (b) which despite the blur performs better because the QR Code resembles more to a flat QR Code.

On top of that, in the most common uses, readout is carried out by casual users holding handheld devices (like smartphones) in manifold angles and perspectives. Surprisingly, these perspective effects are not tackled by the original QR Code standard specification, but are so common that are addressed in most of the state-of-the-art QR Code reader implementations [59; 98; 100]. Still, the issues caused by a non-flat topography remain mostly unsolved, and the usual recommendation is just acquiring the QR Code image from a farther distance, where curvature effects turn apparently smaller thanks to the laws of perspective (see Figure 3.1.b and Figure 3.1.c). This however is a stopgap measure rather than a solution, that fails frequently when the surface deformation is too high or the QR Code is too big.

Therefore, reading QR codes from complex, arbitrary surfaces remains an open problem till now. Other authors have already demonstrated that it is possible to use the QR Code itself to fit the surface underneath to a pre-established topography model. These proposals only work well with surfaces that resemble the shape model assumed (e.g. a cylinder, a sphere, etc.) and mitigate the problem just for a limited set of objects and surfaces, for which analytical topography models can be written.

Regarding perspective transformation models, Sun et al. proposed the idea of using these transformations as a way to enhance readability in handheld images from mobile phones [106]. This idea was explored also by Lin and Fuh, showing that their implementation performed better than ZXing [107], a commercial QR Code decoder formerly developed by Google [100]. Concerning cylindrical transformations, Li, X. et al. [108], Lay et al. [109; 110] and Li, K. [111] reported results on QR Codes placed on top of cylinders. More recently, Tanaka introduced the idea of correcting cylindrical deformation using an Image-to-Image Translation Network [112]. Finally, the problem of arbitrary surface deformations has just been explored very recently. Huo et al. suggested a solution based on Back-Propagation Neural Networks [113]. Kikuchi et al. presented a radically different approach from the standpoint of additive manufacturing by 3D printing the QR codes inside those arbitrary surfaces, and thus solving the inverse problem by rendering apparent planar QR Codes during capture [114].

3.1 *Proposal*

Here, since a general solution for the decoding of QR Codes placed on top of arbitrary topographies is missing, we present our proposal on this matter based on the thin-plate spline 2D transformation [115]. Thin-plate splines (TPS) are a common solution to fit arbitrary data and have been used before in pattern recognition problems: Bazen et al. [116] and Ross et al. [117] used TPS to match fingerprints; Shi et al. used TPS together with Spatial Transformer Networks to improve handwritten character recognition by correcting arbitrary deformations [118], and Yang et al. reviewed the usage of different TPS derivations in the point set registration problem [119].

In order to investigate the advantages of the TPS with respect to former approaches, we take here the above-mentioned geometric surface fittings as reference cases, namely: (i) affine coplanar transformations (see Figure 3.2.a), (ii) projective transformations (see Figure 3.2.b), and (iii) cylindrical transformations (see Figure 3.2.c).

Then we introduce our proposal for arbitrary surfaces based on (iv) the thin-plate spline 2D transformation (see Figure 3.2.d) and

benchmark against each other. With all four methods we use a commercial barcode scanner, ZBar [98], to decode the corrected image and observe the impact of each methodology, not just on the geometrical correction but also on the actual data extraction.

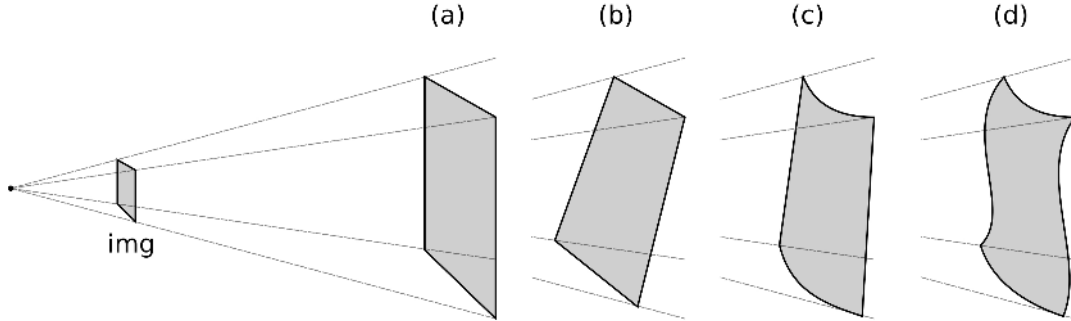


Figure 3.2: Projection of different surfaces into the capture plane (*img*) when acquiring images from a digital camera. A QR Code placed on each one of these surfaces will show different deformations (a) an affine (coplanar) plane, (b) a projective (noncoplanar) plane, (c) a cylindrical surface and (d) a thin-plate spline surface, it is continuous and derivable.

3.1.1 Fundamentals of projections

In chapter 2 we have defined images as mappings from a \mathbb{R}^2 plane to a scalar field \mathbb{R} , assuming they are grayscale. Figure 3.2 shows this \mathbb{R}^2 plane and labels it as **img**. Let us define a projective transformation of this plane as an application between two planes:

$$f : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \quad (3.1)$$

Also, let the points $(x, y) \in \mathbb{R}^2$ and $(x', y') \in \mathbb{R}^2$, we can then define an analytical projective mapping between those two points as:

$$\begin{aligned} x' &= f_x(x, y) = a_{0,0} \cdot x + a_{0,1} \cdot y + a_{0,2} \\ y' &= f_y(x, y) = a_{1,0} \cdot x + a_{1,1} \cdot y + a_{1,2} \end{aligned} \quad (3.2)$$

where $a_{i,j} \in \mathbb{R}$ are the weights of the projective transform. For a more compact notation, (x, y) and (x', y') can be replaced by homogeneous coordinates [120] $(p_0, p_1, p_2) \in \mathbb{P}^2\mathbb{R}$ and $(q_0, q_1, q_2) \in \mathbb{P}^2\mathbb{R}$, respectively, that allow expressing the transformation in a full matrix notation¹:

$$\begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & 1 \end{pmatrix} \cdot \begin{pmatrix} p_0 \\ p_1 \\ 1 \end{pmatrix} = \begin{pmatrix} q_0 \\ q_1 \\ 1 \end{pmatrix} \quad (3.3)$$

Finally, we can simplify this expression by naming our matrices as:

$$\mathbf{A} \cdot \mathbf{P} = \mathbf{Q} \quad (3.4)$$

¹ Homogeneous coordinates introduce an additional coordinate p_2 and q_2 in our system, which extends the point representation from a plane (\mathbb{R}^2) to a projective space ($\mathbb{P}^2\mathbb{R}$). We will define that $p_2 = q_2 = 1$ only for our landmarks [120].

Here, we will work with four projective transformations: the affine transformation (AFF), the projective transformation (PRO), the cylindrical transformation (CYL) and the thin-plate spline transformation (TPS). We can define all of them as subsets or extensions of projective transformations, so we will have to specifically formulate \mathbf{A} for each one of them. To do so, we need to know the landmarks in the captured image (acting as vector \mathbf{Q}) and their “correct” location in a non-deformed corrected image (acting as vector \mathbf{P}).

3.1.2 Proposed transformations

Affine (AFF). This transformation uses the landmarks to fit a coplanar plane to the capturing device sensor (see Figure 3.3). It can accommodate translation, rotation, zoom and shear deformations [120]. An affine transformation can be expressed in terms of Equation 3.3, only taking $a_{2,0} = a_{2,1} = 0$:

$$\begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_0 \\ p_1 \\ 1 \end{pmatrix} = \begin{pmatrix} q_0 \\ q_1 \\ 1 \end{pmatrix} \quad (3.5)$$

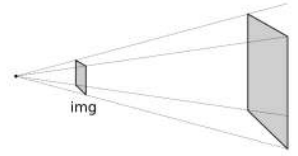


Figure 3.3: Projection of an affine surface into the capture plane (img) when acquiring images from a digital camera.

This yields to a system with only 6 unknown $a_{i,j}$ weights. Thus, if we can map at least 3 points in the QR Code surface to a known location (e.g. finder pattern centers) we can solve the system for all $a_{i,j}$ using the expression of Equation 3.4 with:

$$\begin{aligned} \mathbf{A} &= \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ 0 & 0 & 1 \end{pmatrix}, \\ \mathbf{P} &= \begin{pmatrix} p_{0,0} & p_{0,1} & p_{0,2} \\ p_{1,0} & p_{1,1} & p_{1,2} \\ 1 & 1 & 1 \end{pmatrix} \text{ and} \\ \mathbf{Q} &= \begin{pmatrix} q_{0,0} & q_{0,1} & q_{0,2} \\ q_{1,0} & q_{1,1} & q_{1,2} \\ 1 & 1 & 1 \end{pmatrix}. \end{aligned} \quad (3.6)$$

Projective (PRO). This transformation uses landmarks to fit a non-coplanar plane to the capturing plane (see Figure 3.4). Projective transformations use Equation 3.3 without any further simplification. Also, Equation 3.4 is still valid, but now we have up to 8 unknown $a_{i,j}$ weights to be determined. Therefore, we need at least 4 landmarks to solve the system for \mathbf{A} , then:

$$\begin{aligned} \mathbf{A} &= \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & 1 \end{pmatrix}, \\ \mathbf{P} &= \begin{pmatrix} p_{0,0} & p_{0,1} & p_{0,2} & p_{0,3} \\ p_{1,0} & p_{1,1} & p_{1,2} & p_{1,3} \\ 1 & 1 & 1 & 1 \end{pmatrix} \text{ and} \\ \mathbf{Q} &= \begin{pmatrix} q_{0,0} & q_{0,1} & q_{0,2} & q_{0,3} \\ q_{1,0} & q_{1,1} & q_{1,2} & q_{1,3} \\ 1 & 1 & 1 & 1 \end{pmatrix}. \end{aligned} \quad (3.7)$$

Notice that the four points in must not be collinear three-by-three, neither the points of if we want the mapping to be invertible [120].

Cylindrical (CYL). This transformation uses landmarks to fit a cylindrical surface, which can be decomposed into a projective transformation and a pure cylindrical deformation (see Figure 3.5). Thus, the cylindrical transformation extends the projective general transformation (Equation 3.2) and adds a non-linear term to the projection:

$$\begin{aligned} x' &= f_x(x, y) = a_{0,0} \cdot x + a_{0,1} \cdot y + a_{0,2} + w_0 \cdot g(x, y) \\ y' &= f_y(x, y) = a_{1,0} \cdot x + a_{1,1} \cdot y + a_{1,2} + w_1 \cdot g(x, y) \end{aligned} \quad (3.8)$$

where $g(x, y)$ is the *cylindrical term*, which takes the form of [111; 108]:

$$g(x, y) = \begin{cases} \sqrt{r^2 - (c_0 - x)^2} & \text{if } r^2 - (c_0 - x)^2 \geq 0 \\ 0 & \text{if } r^2 - (c_0 - x)^2 < 0 \end{cases} \quad (3.9)$$

where $r \in \mathbb{R}$ is the radius of the cylinder, and $c_0 \in \mathbb{R}$ is the first coordinate of any point in the centerline of the cylinder. Now, Equation 3.3 becomes extended with another dimension for cylindrical transformations:

$$\begin{pmatrix} w_0 & a_{0,0} & a_{0,1} & a_{0,2} \\ w_1 & a_{1,0} & a_{1,1} & a_{1,2} \\ w_2 & a_{2,0} & a_{2,1} & 1 \end{pmatrix} \cdot \begin{pmatrix} g(p_0, p_1) \\ p_0 \\ p_1 \\ 1 \end{pmatrix} = \begin{pmatrix} q_0 \\ q_1 \\ 1 \end{pmatrix} \quad (3.10)$$

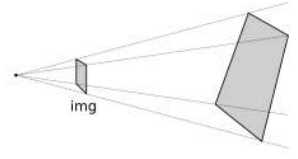


Figure 3.4: Projection of a projective surface into the capture plane (img) when acquiring images from a digital camera.

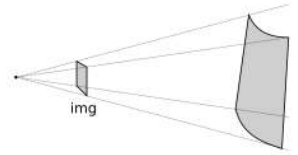


Figure 3.5: Projection of a cylindrical surface into the capture plane (img) when acquiring images from a digital camera.

Applying the same reasoning as before, we have now 8 unknown $a_{i,j}$ plus 3 unknown w_j weights to fit. Equivalent matrices (Equation 3.4) for cylindrical transformations need now at least 6 landmarks and looks like:

$$\mathbf{A} = \begin{pmatrix} w_0 & a_{0,0} & a_{0,1} & a_{0,2} \\ w_1 & a_{1,0} & a_{1,1} & a_{1,2} \\ w_2 & a_{2,0} & a_{2,1} & 1 \end{pmatrix},$$

$$\mathbf{P} = \begin{pmatrix} g(p_{0,0}, p_{1,0}) & \dots & g(p_{0,5}, p_{1,5}) \\ p_{0,0} & \dots & p_{0,5} \\ p_{1,0} & \dots & p_{1,5} \\ 1 & \dots & 1 \end{pmatrix} \text{ and} \quad (3.11)$$

$$\mathbf{Q} = \begin{pmatrix} q_{0,0} & \dots & q_{0,5} \\ q_{1,0} & \dots & q_{1,5} \\ 1 & \dots & 1 \end{pmatrix}.$$

Thin-plate splines (TPS). This transform uses the landmarks as centers of radial basis splines to fit the surface in a non-linear way that resembles the elastic deformation of a metal thin-plate bent around fixed points set at these landmarks [115] (see Figure 3.6). The radial basis functions are real-valued functions:

$$h : [0, \infty) \rightarrow \mathbb{R} \quad (3.12)$$

that take into account a metric on a vector space. Their value only depends on the distance to a reference fixed point:

$$h_c(v) = h(\|v - c\|) \quad (3.13)$$

where $v \in \mathbb{R}^n$ is the point in which the function is evaluated, $c \in \mathbb{R}^n$ is the fixed point, h is a radial basis function. Equation 3.13 reads as " $h_c(v)$ is a kernel of h in c with the metric $\|\cdot\|$ ". Similarly to cylindrical transformations (Equation 3.8), we extended the affine transformation (Equation 3.2) with N nonlinear spline terms:

$$\begin{aligned} x' &= f_x(x, y) = a_{0,0} \cdot x + a_{0,1} \cdot y + a_{0,2} + \sum_{k=0}^{N-1} w_{0,k} \cdot h_k((x, y)) \\ y' &= f_y(x, y) = a_{1,0} \cdot x + a_{1,1} \cdot y + a_{1,2} + \sum_{k=0}^{N-1} w_{1,k} \cdot h_k((x, y)) \end{aligned} \quad (3.14)$$

where $w_{j,k}$ are the weights of the spline contributions, and $h_k(x, y)$ are kernels of h in N landmark points.

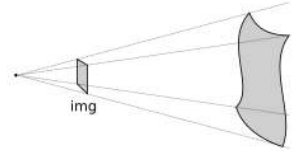


Figure 3.6: Projection of an arbitrary surface into the capture plane (img) when acquiring images from a digital camera.

These radial basis function remains open to multiple definitions. Bookstein [115] found that the second order polynomial radial basis function is the proper function to compute splines in \mathbb{R}^2 mappings in order to minimize the bending energy, and mimic the elastic behavior of a metal thin-plate. Thus, let h be:

$$h(r) = r^2 \ln(r) \quad (3.15)$$

with the corresponding kernels computed using the euclidean metric:

$$\|(x, y) - (c_x, c_y)\| = \sqrt{(x - c_x)^2 + (y - c_y)^2} \quad (3.16)$$

Finally, in matrix representation, terms from Equation 3.4 are expanded as follows:

$$\mathbf{A} = \begin{pmatrix} w_{0,0} & \dots & w_{0,N-1} & a_{0,0} & a_{0,1} & a_{0,2} \\ w_{1,0} & \dots & w_{1,N-1} & a_{1,0} & a_{1,1} & a_{1,2} \end{pmatrix},$$

$$\mathbf{P} = \begin{pmatrix} h_0(p_{0,0}, p_{1,0}) & \dots & h_0(p_{0,N-1}, p_{1,N-1}) \\ \vdots & \dots & \vdots \\ h_{N-1}(p_{0,0}, p_{1,0}) & \dots & h_{N-1}(p_{0,N-1}, p_{1,N-1}) \\ p_{0,0} & \dots & p_{0,N-1} \\ p_{1,0} & \dots & p_{1,N-1} \\ 1 & \dots & 1 \end{pmatrix} \text{ and } (3.17)$$

$$\mathbf{Q} = \begin{pmatrix} q_{0,0} & \dots & q_{0,N-1} \\ q_{1,0} & \dots & q_{1,N} \end{pmatrix}.$$

First, notice that only $a_{i,j}$ affine weights are present, since this definition does not include a perspective transformation. Second, in contrast with previous transformations, this system is unbalanced: we have a total of $2N + 6$ weights to compute ($2N$ $w_{j,k}$ spline weights plus 6 $a_{i,j}$ affine weights) however we only have defined N landmarks. In the previous transformations, we used additional landmarks to solve the system, but Bookstein imposed an additional condition of the spline contributions: the sum of $w_{j,k}$ coefficients to be 0, and also their cross-product with the $p_{i,k}$ landmark coordinates [115]. Such condition makes spline contributions tend to 0 at infinity, while affine contributions prevail. This makes our system of equations solvable and can be expressed as:

$$\begin{pmatrix} w_{0,0} & \dots & w_{0,N-1} \\ w_{1,0} & \dots & w_{1,N-1} \end{pmatrix} \cdot \begin{pmatrix} p_{0,0} & \dots & p_{0,N-1} \\ p_{1,0} & \dots & p_{1,N-1} \\ 1 & \dots & 1 \end{pmatrix}^T = 0 \quad (3.18)$$

3.2 *Experimental details*

Experiments were designed to reproduce the QR Code life-cycle in different scenarios, which can be regarded as a digital communication channel: a message made of several bytes with their corresponding error correction blocks is encoded in the black and white pixels of the QR Code, that is transmitted through a visual channel (typically, first displayed or printed and then captured by a camera), and finally decoded and the original message retrieved (see Figure 3.8.a).

In this context, the effects of the challenging surface topographies can be seen as an additional step in the channel, where the image is deformed in different ways prior to the capture. To investigate these effects we attached our QR codes to real complex objects to collect pictures with relevant deformations (see details below). Then, in order to expand our dataset, we incorporated an image augmentation step that programmatically added additional random projective deformations to the captured images [121]. Finally, we considered the surface fitting and correction as an additional step in the QR Code processing workflow, prior to attempting decoding. This proved more effective than directly attempting the QR Code decoding based on the distorted image with deformed position and feature patterns due to the surface topography (see Figure 3.8.b).

3.2.1 *Datasets*

We created 3 datasets to evaluate the performance of different transformations in different scenarios with arbitrary surface shapes.

- **Synthetic QR Codes (SYNT).** This dataset was intended to evaluate the impact of data and geometry of the QR Code on the proposed deformation correction methods. To that end, we generated the QR Codes as digital images, without printing them, and applied to them affine and projective transformations directly with image augmentation techniques (see Figure 3.7.a). This dataset contained 12 QR Code versions (from version 1 to 12), each of them repeated 3 times with different random data (IDs), and 19 augmented images plus the original one. The mutual combination of all these variations yielded a total of 720 images to be processed by the proposed transformations (see Table 3.1).
- **QR Codes on flat surfaces (FLAT).** In this dataset, we only encoded 1 QR Code version 7 and printed it. We placed this QR Code on different flat surfaces and captured images (see Figure 3.7.b). Thus, we only expected projective deformations in this dataset, to be used as a reference. We also augmented the captured images to match the same quantity of images from the previous dataset (see Table 3.1).

- QR Codes on challenging surfaces (SURF).** In this dataset, we used the same QR Code we used in the FLAT dataset, but placed on top of challenging surfaces, such as bottles, or manually deforming the QR Codes (see Figure 3.7.c). We expected here to have cylindrical and arbitrary deformations in the dataset. Also, we augmented the captured images to match the size of the other datasets (see Table 3.1).

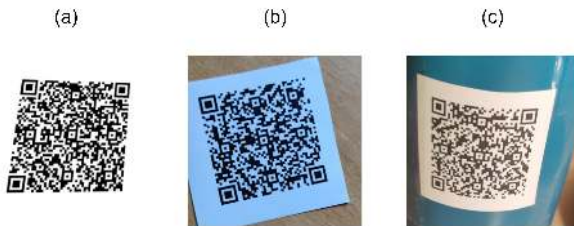


Figure 3.7: Example images from the three datasets - (a) SYNT, (b) FLAT and (c) SURF - showing similar QR codes in different surface deformations.

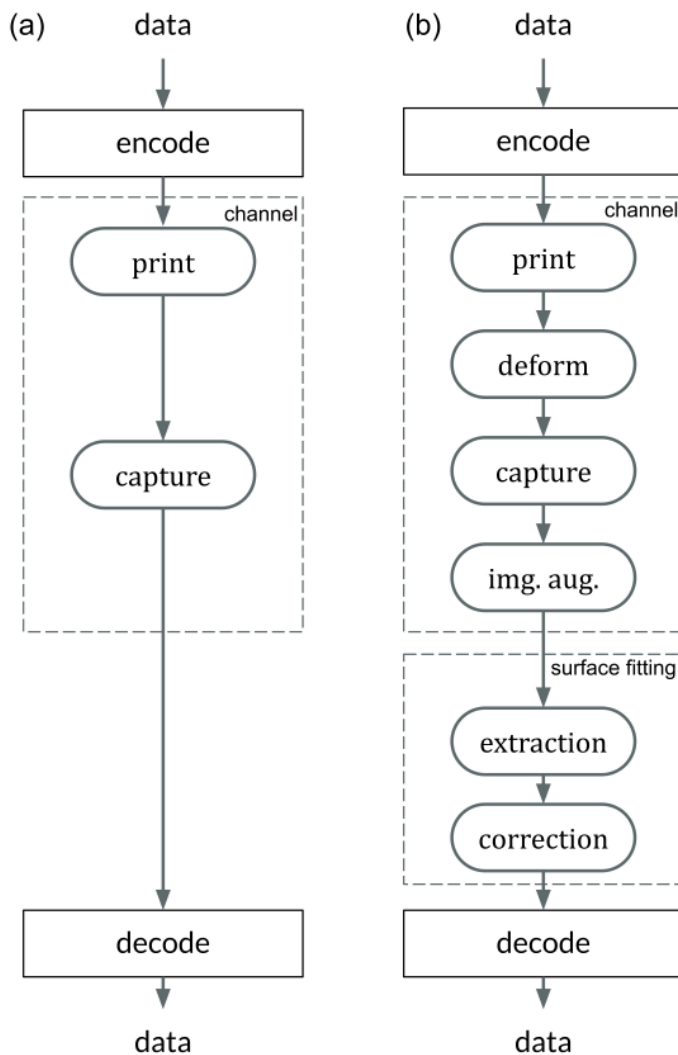


Figure 3.8: (a) Block diagram for a general encoding-decoding process of a QR Code. (b) A modified diagram with the addition of a deformation due to a noncoplanar surface topography and surface fitting stage which contains a correction steps where image deformation is reverted to improve readout. In our experiments, also, an image augmentation step was added to be used in the proposed experiments for this work.

SYNT	Values	Dataset size
Version	from 1 to 13	12
IDs (per version)	random	3
Captures		1
Image augmentation		20
<i>Total</i>		<i>720</i>
FLAT	Values	Dataset size
Version	7	1
IDs (per version)	https://color-sensing.com/	1
Captures		48
Image augmentation		15
<i>Total</i>		<i>720</i>
SURF	Values	Dataset size
Version	7	1
IDs (per version)	https://color-sensing.com/	1
Captures		48
Image augmentation		15
<i>Total</i>		<i>720</i>

Table 3.1: Summary of dataset sizes. All datasets attempt to have the same size employing QR Code generation, different captures or image augmentation.

3.3 Results

3.3.1 Qualitative surface fitting

We fitted the four transformations (AFF, PRO, CYL and TPS) to the surface underneath all the QR Code samples from the three datasets (SYNT, FLAT and SURF). To evaluate visually how accurate each transformation was, a squared lattice of equally spaced points on the predicted QR Code surface was back-projected into the original image space. For illustration purposes, results on representative samples of the SYNT, FLAT and SURF datasets can be seen in Figure 3.9, Figure 3.10 and Figure 3.11, respectively.

Our first dataset, SYNT, contained samples with affine (Figure 3.9.a) and projective (Figure 3.9.b) deformations. We observed that all four transformations achieved good qualitative fittings with images presenting affine deformations. This is an expected result, since all transformations implement affirm terms. Consequently, when it comes to projective deformations, the AFF transformation failed to adjust the fourth corner (the one without any finder pattern, see Figure 2.13.a), as expected. Comparatively, the PRO and the CYL transformations lead to similarly good results, since both can accommodate perspective effects. Finally, TPS fitted the surface well, specially inside the QR Code, and a slight non-linear deformation was present outside the boundaries of the barcode, but these are irrelevant for QR Code decoding purposes.

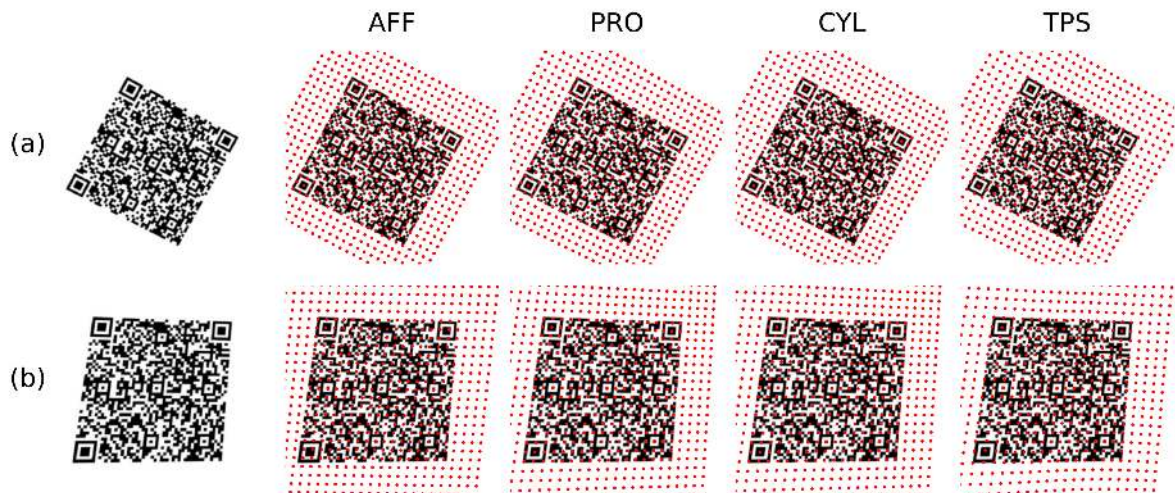


Figure 3.9: Two examples (a), (b) from the SYNT dataset. The surfaces were fitted by the four methods described (AFF, PRO, CYL and TPS). The surface fitting is shown as a lattice of red points back-projected onto the original image.

The FLAT dataset involved QR Codes that were actually printed and then imaged with a smartphone camera. These QR Codes were captured in projective deformations (Figure 3.10.b), some of them resembling affine deformations (Figure 3.10.a), and most of them just a combination of both. Qualitative performance comparison is similar to that of the SYNT dataset. Again, the AFF transformation failed to correctly approach the fourth corner. Also, we confirmed that PRO, CYL and TPS performed well under the FLAT images, but TPS showed a non-linear, irrelevant, overcorrection outside the barcode.

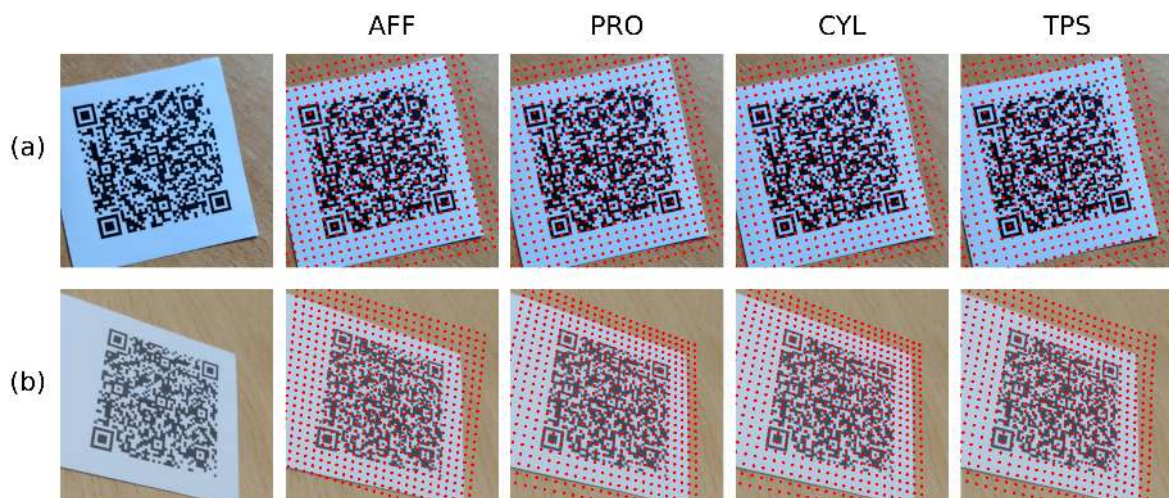


Figure 3.10: Two examples (a), (b) from the FLAT dataset. The surfaces were fitted by the four methods described (AFF, PRO, CYL and TPS). The surface fitting is shown as a lattice of red points back-projected onto the original image.

The SURF dataset was the most challenging dataset in terms of modeling adverse surface topographies. QR Codes here were imaged again with a smartphone, but in this case the surface under the barcode was distorted in several ways: randomly deformed by hand (Figure 3.11.a), placed on top of a small bottle (Figure 3.11.b), a large bottle (Figure 3.11.c), etc. Results showed that AFF, PRO and CYL methods were not able to correctly match a random surface (i.e. deformed by hand), as expected. Instead, TPS worked well in these conditions, being a great example of the power of the spline decomposition to match slow varying topographies, if a sufficiently high number of landmarks is available. For cylindrical deformations (i.e. QR Codes in bottles), AFF and PRO methods were again unsuccessful. CYL performed better with the small bottles than with the large ones. Apparently, higher curvatures (i.e. lower bottle radius r) facilitate the fitting of this parameter and improve the quality of the overall prediction radius of the projected cylinder before fitting the surface. Thus, the CYL method properly fits the cylinder radius from one of the sides of the QR Codes with 2 finder patterns and often fails to fit the opposite side. Interestingly, The TPS method performed opposite to the CYL method in the cylindrical deformations, tackling better surfaces with mild curvatures.

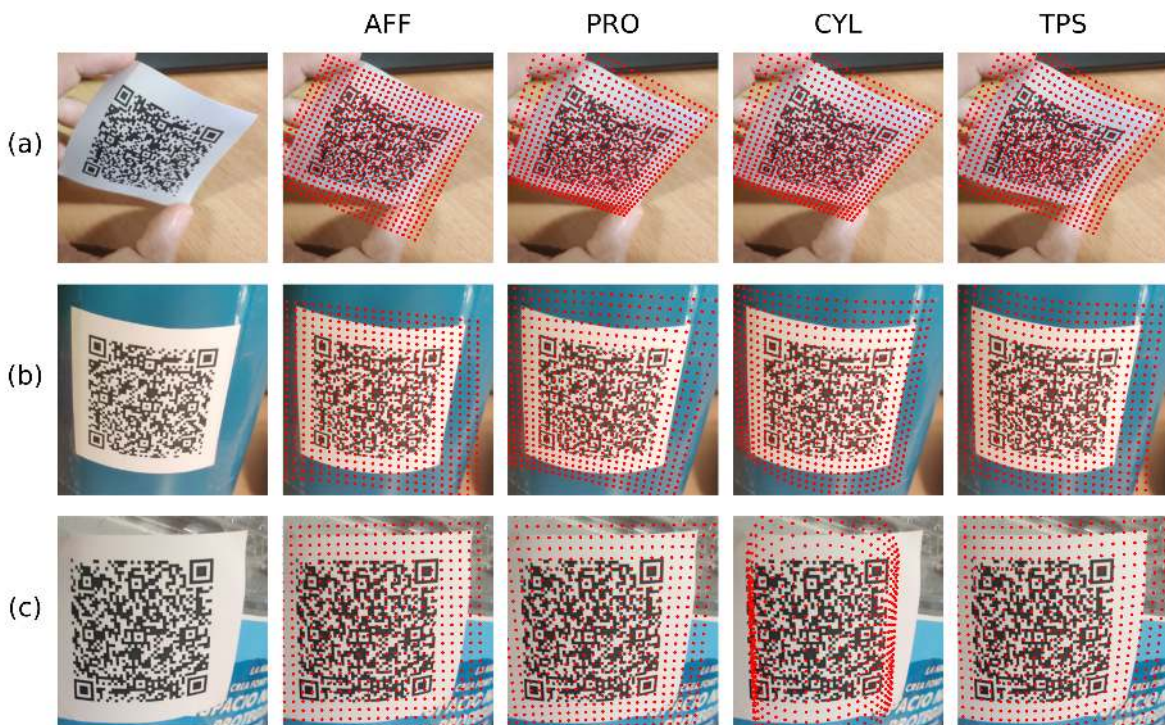


Figure 3.11: Three examples (a), (b), (c) from the SURF dataset. The surfaces were fitted by the four methods described (AFF, PRO, CYL and TPS). The surface fitting is shown as a lattice of red points back-projected onto the original image.

3.3.2 Quantitative data readability

In order to evaluate the impact of these surface prediction capabilities on the actual reading of the QR Code data, we run the full decoding pipeline mentioned in Figure 3.8 for all the images in the three datasets (SYNT, FLAT and SURF) with the four transformations (AFF, PRO, CYL and TPS). There, once surface deformation was corrected, the QR Code data was extracted with one of the most widespread barcode decoder (ZBar [56; 122]). Therefore, in this experiment we are actually evaluating how the error made on the assessment of the QR Code geometry, due to surface and perspective deformations, impacts on the evaluation of the individual black/white pixel bits; and to what extent the native QR Code error correction blocks (based on Reed-Solomon according to the standard).

We then defined a success metric of data readability[25] (\mathcal{R}) as:

$$\mathcal{R} = 100 \cdot \frac{N_{\text{decoded}}}{N_{\text{total}}} [\%] \quad (3.19)$$

where N_{decoded} is the number of QR Codes successfully decoded and N_{total} is the total amount of QR Codes of a given dataset and transformation. Such a number has a direct connection with the user experience. In a manual reading scenario, it tells us how often the user will have to repeat the picture (e.g. $\mathcal{R} = 95\%$ means 5 repetitions out of every 100 uses). In applications with automated QR Code scanning, this measures how long it will take to pick up the data.

Figure 3.12 summarizes the readability performance of the four transformations with the three datasets. For the SYNT and FLAT datasets, PRO, CYL and TPS scored at 100% or close. AFF scored only a 78% and 60% for the SYNT and the FLAT datasets, respectively. This is because AFF lacks the perspective components that PRO and CYL incorporate to address this problem. It is noteworthy that the TPS scored similar to the PRO and CYL for these two datasets: despite TPS does not include perspective directly, it is composed of affine and non-linear terms, and the later ones can fit a perspective deformation.

This behavior is also confirmed for the segregated data on the SYNT dataset (see Figure 3.13), where the TPS performed slightly worse on images with a perspective deformation, similarly to the AFF. Also in Figure 3.13, we see that AFF showed its best performance (97%) in the subset of images where only affine transformation was present, rendering lower in the projective ones (70%).

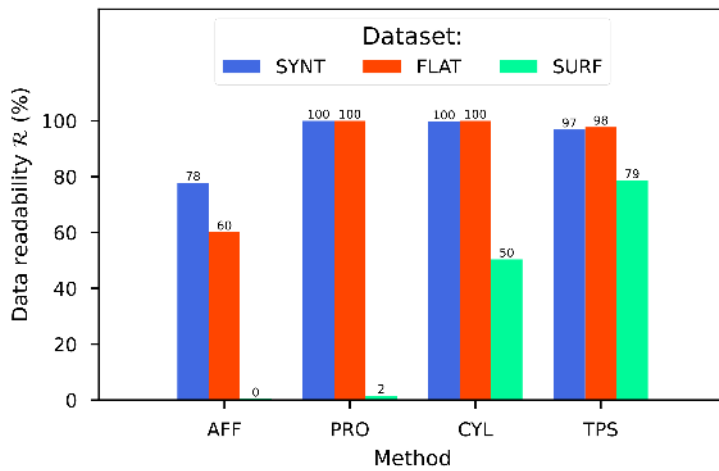


Figure 3.12: Data readability (\mathcal{R}) of each dataset (SYNT, FLAT, SURF) for each transformation method (AFF, PRO, CYL and TPS).

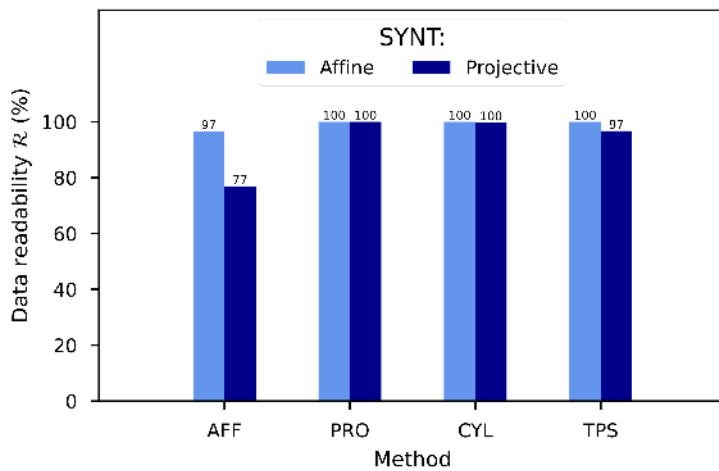


Figure 3.13: Data readability (\mathcal{R}) of the SYNT dataset, segregated by the kind of deformation (affine or perspective) that the QR Codes were exposed to, for each transformation method (AFF, PRO, CYL and TPS).

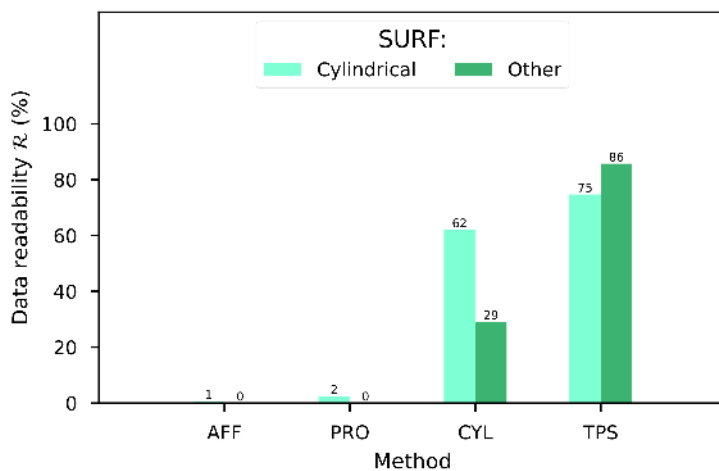


Figure 3.14: Data readability (\mathcal{R}) of the SURF dataset segregated by the kind of deformation (cylindrical or other) that the QR Codes were exposed to, for each transformation method (AFF, PRO, CYL and TPS).

Figure 3.14 shows the segregated data for the SURF dataset, neither the AFF nor the PRO transformations decoded almost any QR Code (1%-2%). CYL performed well for cylindrical surfaces in the SURF dataset (62%), but got beaten by the TPS results by 13 points (from 62% to 75%). Moreover, CYL scored less than 30% in images without explicit cylindrical deformations, as expected; while the TPS remained well over 85%. This is a remarkable result for the TPS, considering that the rest of transformations failed completely at this task.

Finally, we wanted to benchmark the methodology proposed here with a popular, state-of-the-art decoder like ZBar. To that end, we fed ZBar with all our datasets of images (without pre-processing and with surface geometry corrections made). Figure 3.15 shows that ZBar implementation to read QR Code pixels out of reading each line of the QR Code as one dimensional barcode[98] performs very well with the SYNT dataset. But, in the more realistic smartphone-captured images from FLAT, ZBar performed poorly, succeeding only in approximately in $\frac{2}{3}$ of the dataset.

Surprisingly, ZBar was still able to decode some SURF dataset images. We compared these results with a combined sequence of CYL and TPS transformations that can be regarded as TPS with a fall-back to the CYL method, since CYL has its own fall-back into PRO. Our solution, slightly improved the good results of ZBar in the SYNT dataset, obtained a perfect score in the FLAT dataset where ZBar struggles (100% vs 75%), and displayed a remarkable advantage (84% vs 19%) in decoding the most complex SURF dataset. We can therefore state that the here-proposed methodology outperforms the state-of-the-art when facing complex surface topographies.

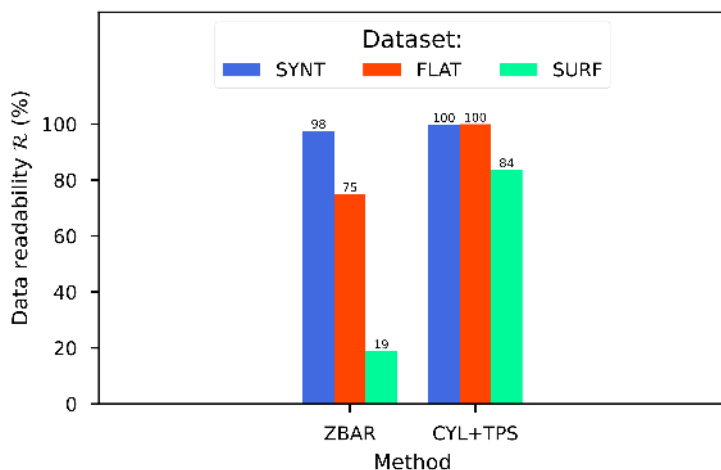


Figure 3.15: Data readability (\mathcal{R}) of the three datasets (SYNT, FLAT and SURF) when processed with ZBar and our combined CYL and TPS methods.

3.4 Conclusions

We have presented a method to increase the readout performance of QR Codes suffering surface deformations that pose a challenge to existing solutions. The thin-plate splines (TPS) transformation has proven to be a general solution for arbitrary deformations that outperforms other transformations proposed in the literature (AFF, PRO, CYL), and the commercial implementation ZBar, by more than 4 times.

TPS presented a few corner cases when approaching high perspective projective transformations (i.e. the QR Code is way noncoplanar with the capture device in a flat surface), where CYL and PRO methods performed very well. The results presented here point at an optimum solution based on a sequential combination of the three methods as fall-back alternatives (i.e. $TPS \rightarrow CYL \rightarrow PRO$).

This work has demonstrated how the TPS method is a suitable candidate to correct images where QR Codes are present using traditional feature extraction using the QR Codes features themselves. Futures developments could involve some enhancements to this methodology, we explored them, and we expose now some ideas.

First, one could enhance the TPS definition to incorporate perspective components into the TPS fittings, which one of the differences between the CYL and the TPS method. This was done by Bartoli et al., in their work they renamed the TPS method as *DA-Warp*, standing for 'Deformable Affine Warp', and introduced three new methods: the *RA-Warp* – 'Rigid Affine Warp' –, the *DP-Warp* – 'Deformable Perspective Warp' – and the *RP-Warp* – 'Rigid Perspective Warp' –; their framework could be applied to images with QR Codes to increase the performance of our solution and avoid the fall-back $TPS \rightarrow CYL \rightarrow PRO$ [123].

Second, approximating the radial basis contributions to the TPS fittings is a well-know technique to relax the condition that each landmark must be mapped directly to its respective landmark in the corrected image [124; 125]. This is usually done by adding a smoothing factor λ to the diagonal of the \mathbf{P} array (see Equation 3.17). We deepen in this methodology in chapter 5 when we applied TPS to color correction, for QR Code extraction we discarded to use it because we often want the extracted key features to match exactly their position in the recovered image. Nevertheless, as it was not checked it should be addressed in some future work.

Third, in this work we demonstrated that TPS can be used to map the surface where the QR Code is posed, no matter how adversarial was that surface – if it is continuous and derivable –. The TPS framework needs a huge quantity of landmark points to compute the TPS correction, *the more, the better*. We extracted these landmarks with classical feature extractors (contour detection, pattern matching, etc.), but one could use neural networks to solve that problem. For example, Shi et al. [118] presented an interesting solution which also involved TPS, they trained a neural network to discover the optimal landmarks for a given image with a text, in order to rectify it using a TPS method, and later on, apply a text recognition network to recover the text. Other authors, like Li et al. [126] have presented recent work using the popular general-purpose recognition neural network ‘YOLOv3’ [127] to locate the corners of ArUco codes [19].

Finally, our method could be applied to other 2D barcodes, such as DataMatrix, Aztec Code or MaxiCode. The main blocker to implement our methodology to such machine-readable patterns is the feature extraction. For example, QR Codes implement a variety of patterns, as detailed in chapter 2: finder, alignment and timing patterns. For example, DataMatrix codes only present timing patterns [46], but this handicap might be avoided using better extractors that use the Hough transform to recover the full grid of the machine-readable pattern not only the key features [128].

Chapter 4. Back-compatible Color QR Codes

As we have previously introduced, the popularization of digital cameras enabled an easier access to photography devices to the people. Nowadays, modern smartphones have onboard digital cameras that can feature good color reproduction for imaging uses. However, when actual colorimetry is needed, the smartphone camera sensor does not suffice, needing auxiliary ad hoc tools to evaluate color and guarantee image consistency among datasets [129].

As we have introduced in chapter 2, a traditional approach to achieve a general purpose color calibration is the use of color correction charts, introduced by C.S. McCamy et. al. in 1976 [13], combined with color correction techniques. It is safe to say that, in most of these post-capture color correction techniques, increasing the number and quality of the color references offers a systematic path towards better color calibration, we pursue this topic further in next chapter 5.

In 2018, we presented a first implementation of a machine-readable pattern (see Figure 4.1), based on the image recognizable structures of the QR Codes, that integrated a color changing indicator (sensitive to gases related to bad odor) and a set of color references (to measure that color indication) [29]. In 2020, we reported a more refined solution allocating hundreds of colors into another machine-readable pattern, suitable to measure multiple gas sensors by means of color changes alongside with the reference colors inside a pseudo QR Code pattern [30]. In both solutions, the QR Code finder, the timing and the alignment patterns (detailed in chapter 2) were present and used to find, locate and sample the gas sensitive pixels and the reference colors, but all the digital information was removed. These were, therefore, ad hoc solutions that lacked the advantages of combining a compact colorimetric readout and calibration pattern with the digital data available in a QR Code. These solutions are presented from the standpoint of view of colorimetric sensors in chapter 6.

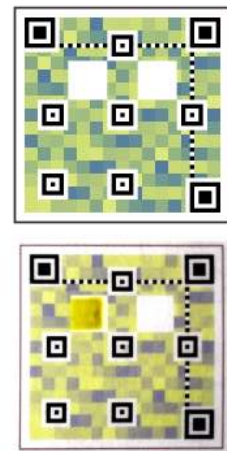


Figure 4.1: A machine-readable pattern to allocate an ammonia sensor. Top: the designed pattern, with two spaces to print a colorimetric sensor. Bottom: the captured version of the pattern with a printed colorimetric dye in one slot. Notice this pattern resembles a QR Code, but it does not contain any data.

4.1 Proposal

Linking the colorimetric problem to a set of digital information opens the door to many potential uses related to automation. For example, the digital data could store a unique ID to identify the actual color calibration references used in the image, or other color-measurement properties e.g. by pointing at a data storage location. When used, for example, in smart packaging, this enables the identification of each package individually, gathering much more refined and granular information.

In this chapter, we propose a solution for that by placing altogether digital information and color references without breaking the QR Code standard in a back-compatible Color QR Code implementation for colorimetric applications. Our solution modifies the above-presented default QR Code encoding process (see Figure 2.6), to enhance the QR Code to embed colors in a back-compatible way (see Figure 4.2).

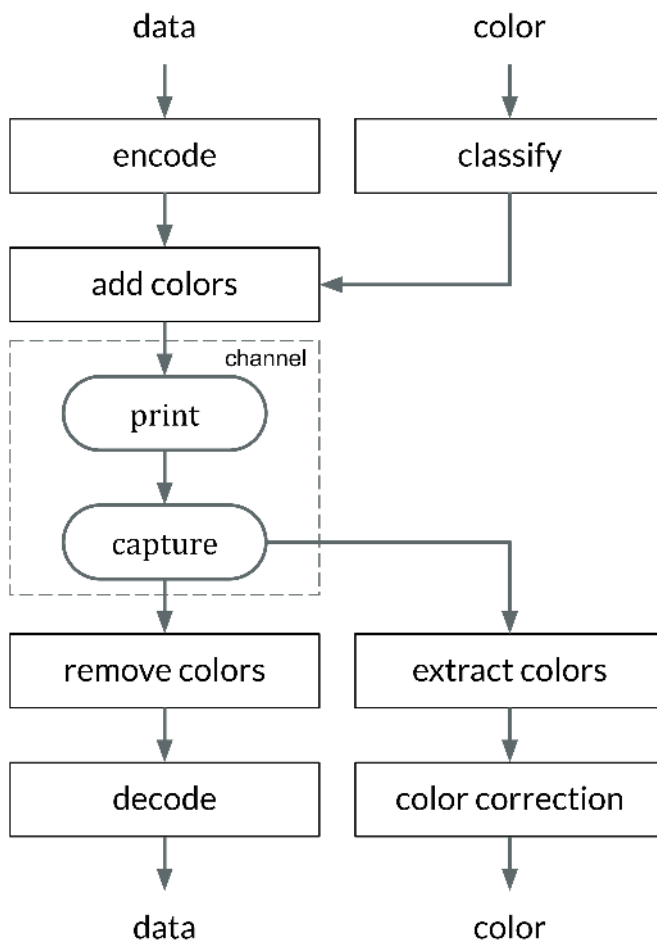


Figure 4.2: Block diagram for a back-compatible encoding-decoding process of a QR Code which features the embedding of a color layer for colorimetric applications. The process can be seen as a global encoding process (digital encode and color encode), followed by a channel (print and capture) and a global decoding process (extract colors and decode digital information). This process is back-compatible with state of the art scanners which remove colors and achieve the decoding of the data and compatible with new decoders which can benefit from color interrogation. The back-compatibility is achieved by following certain rules in the color encoding process (i.e. use the same threshold when placing the colors than when removing them).

This solution is inspired by, but not directly based on, previous Color QR Codes proposals that aim at enhancing the data storage capacity of a QR Code by replacing black and white binary pixels by color ones (see Figure 4.3) [57; 130; 131; 132]. Those approaches offer non-back-compatible barcodes that cannot be decoded with standard readers. Instead, our work offers a design fully compatible with conventional QR Codes. Evidently, without a specialized reader, the color calibration process cannot be carried out either, but back-compatibility assures that any standard decoder will be able to extract the digital data to, e.g. point at the appropriate reader software to carry out the color correction in full. From the point of view of the usability, back-compatibility is key to enable a seamless deployment of this new approach to color calibration, using only the resources already available in smartphones (i.e. the camera and a standard QR decoder).

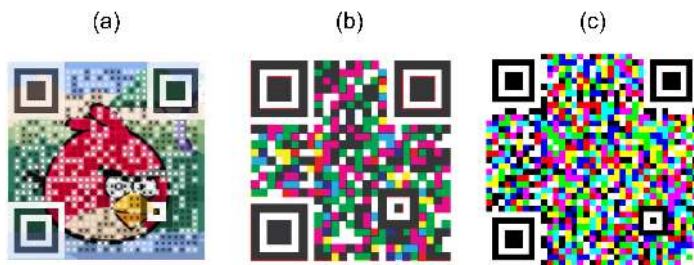


Figure 4.3: Previous state-of-the-art QR Code variants that implement colors in some fashion. (a) A QR Code which is able to back-compatible embed an image. (b) A RGB implementation of QR Codes where 3 different QR Codes are packed in each RGB channel, each channel is back-compatible, although the resulting image is not. (c) A High Capacity Color Barcode, a re-implementation of a QR Code standard using colors, which is not back-compatible with QR Codes.

4.1.1 Color as a source of noise

Before being able to formulate our proposal, it is necessary to study how the additions of color affects the QR Code as carrier in our proposed communication framework (see Figure 4.2). As QR Codes are equipped with and error correction blocks, we can think of color as a source of noise to be corrected with those correction blocks. Deliberate image modifications, like the insertion of a logo, or the inclusion of a color reference chart like we do here, can be regarded as additional noise to the channels. As such, the noise related to this tampering of pixels can be characterized with well-known metrics like the signal-to-noise ratio (SNR) and the bit error ratio (BER).

Let's exemplify this with a QR Code that encodes a website URL (see Figure 4.4.a.). First, this barcode is generated and resized (Figure 4.4.b.) to fit a logo inside (Figure 4.4.c.). The scanning process (Figure 4.2) follows a sequence of sampling –to detect the where QR Code is– (Figure 4.4.d.), desaturation –turning the color image into a grayscale image– (Figure 4.4.e.) and thresholding –to binarize the image– (Figure 4.4.f.). The original binary barcode (Figure 4.4.a.) and the captured one (Figure 4.4.f.) will be clearly different, and here is where the error correction plays a key role to retrieve the correct encoded message -the URL in this example-.

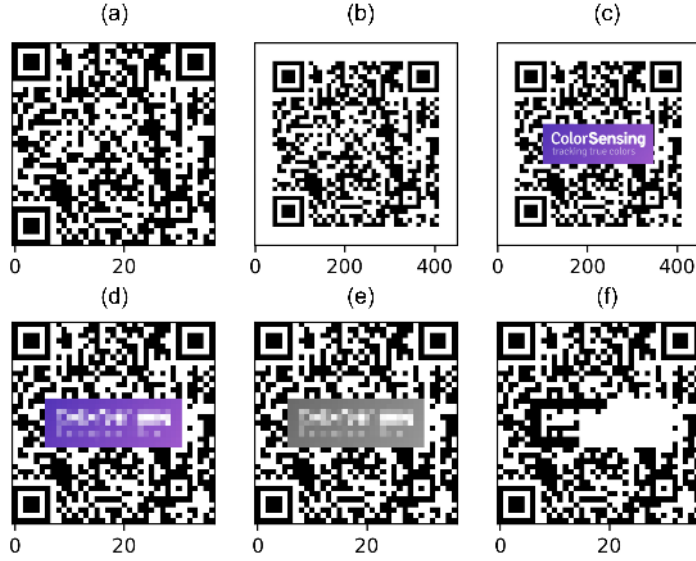


Figure 4.4: A QR Code is overlaid with a logo, which accumulates error due to the presence of the logo. (a) The QR Code is encoded. (b) The code is resized to accommodate the logo. (c) The logo is placed on top of the QR Code. (d) The code is “captured” and down-sampled again. (e) The sampled image is passed to grayscale. (f) The image is binarized, the apparent QR Code differs from the original QR Code (a).

We usually represent signal-to-noise ratio (SNR) from the stand point of view of signal processing. Thus SNR is the ratio between ‘*signal power*’ and ‘*noise power*’. Usually, as signals are evaluated over time this ratio is presented as an root mean square (RMS) average:

$$\text{SNR} = \frac{P_{\text{RMS,signal}}}{P_{\text{RMS,noise}}} \quad (4.1)$$

where $P_{\text{RMS,signal}}$ and $P_{\text{RMS,noise}}$ are the average power of the signal and the noise, respectively. Which in turn is equal to:

$$\text{SNR} = \left(\frac{A_{\text{RMS,signal}}}{A_{\text{RMS,noise}}} \right)^2 \quad (4.2)$$

where $A_{\text{RMS,signal}}$ and $A_{\text{RMS,noise}}$ are the root mean square (RMS) amplitude of the signal and the noise. A RMS of a discrete x variable can be written as:

$$x_{\text{RMS}} = \sqrt{\frac{1}{n} (x_1^2 + x_2^2 + \dots + x_n^2)} \quad (4.3)$$

Then, using this RMS expression and having into account grayscale images can be defined as two-dimensional discrete variables, we can rewrite SNR as follows:

$$\text{SNR} = \frac{\sum_0^n \sum_0^m (A_{\text{gray}}(i, j))^2}{\sum_0^n \sum_0^m (A_{\text{gray}}(i, j) - C_{\text{gray}}(i, j))^2} \quad (4.4)$$

where $A_{\text{gray}} \in [0, 1]^{n \times m}$ are the pixels of the QR Code original image (Figure 4.5.a), which act as a ‘*signal image*’, $C_{\text{gray}} \in [0, 1]^{n \times m}$

are the pixels of the QR Code with the logo in a normalized grayscale (Figure 4.5.b), the difference between both images acts as the ‘noise image’ (Figure 4.5.c), and the ratio between their variances is the SNR. Finally, the SNR values can be expressed in decibels using the standard definition:

$$\text{SNR}_{\text{dB}} = 10 \log_{10}(\text{SNR}). \quad (4.5)$$

The bit error ratio (BER) is defined as the probability to receive an error when reading a set of bits or, in other words, the mean probability to obtain a 0 when decoding a 1 and to obtain a 1 when decoding a 0:

$$\text{BER} = \frac{E(N)}{N} \quad (4.6)$$

where N is the total amount of bits received, and $E(N)$ the errors counted in the N bits. In our case, this translates into the mean probability to obtain a black pixel when decoding a white pixel, and to obtain a white one when decoding a black one. A reformulated BER expression for our binary images is as follows:

$$\text{BER} = \frac{\sum_0^n \sum_0^m |A_{\text{bin}}(i, j) - C_{\text{bin}}(i, j)|}{N} \quad (4.7)$$

where $A_{\text{bin}} \in \{0, 1\}^{n \times m}$ is the binarized version of $A_{\text{gray}} \in [0, 1]^{n \times m}$ (Figure 4.5.d), $C_{\text{bin}} \in \{0, 1\}^{n \times m}$ is the binarized version of $C_{\text{gray}} \in [0, 1]^{n \times m}$ (Figure 4.5.e) and $N = n \cdot m$ are the total pixels in the image. The pixels contributing to the BER are shown in Figure 4.5.f.

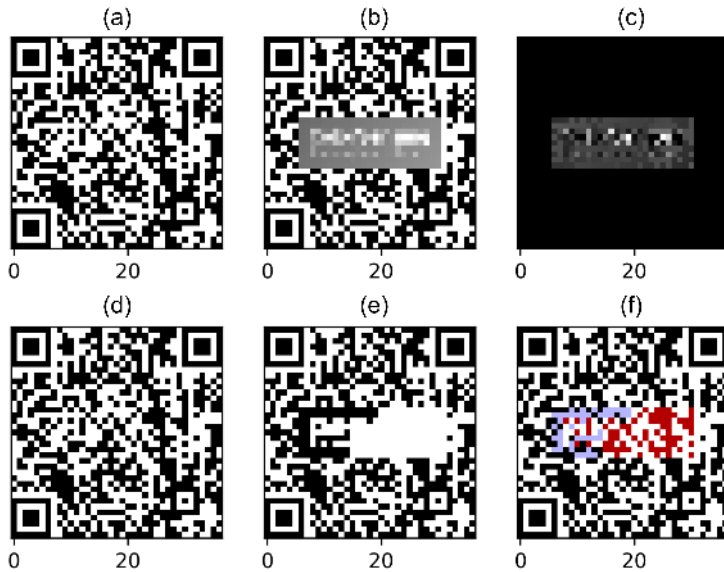


Figure 4.5: A QR Code with a logo is created and read, which accumulates error due to the presence of the logo. (a) The original QR Code encoded. (b) The captured sampled grayscale QR Code. (c) The power difference between (a) and (b). (d) The original grayscale QR Code encoded is binarized, which it is represented exactly as (a). (e) The captured sampled grayscale image from (b) is binarized. (f) The difference between (d) and (e) is shown: light blue pixels correspond to white pixels turned into black by the logo, and dark blue pixels correspond to black pixels turned into white by the logo.

As a summary, Table 4.1 shows the results for the computation of the SNR and BER figures for Figure 4.4 images. As we can see, adding a logo to the pattern represents a noise source that reduces the SNR to 10.53 dB, further noise sources (printing, capture, etc.) will add more noise thus reducing the SNR more. BER metric shows us the impact of the logo when recovering the digital bits, as we have mentioned before this quantity is directly related to the error correction level needed to encode the QR Code. In this example, with a BER of 8.54%, the poorest error correction level (L, 7%) would not suffice to ensure safe readout of the barcode.

Measure	Acronym	Value
Signal-to-Noise ratio	SNR	10.53 dB
Bit error ratio	BER	8.54 %

Table 4.1: The values for the SNR and BER are computed for the QR Code with a logo from Figure 4.4. The SNR is computed using grayscale images. The BER is computed using binary images (see Figure 4.4).

4.1.2 Back-compatibility proposal

We want to achieve back-compatibility with the QR Code standard. This means that we must still be able to recover the encoded data message from the colored QR Code using a standard readout process (capturing, sampling, desaturating and thresholding).

To make it possible we must place these colors inside the barcode avoiding the protected key areas that ensure its readability. In the rest of the available positions, the substitution of black and white pixels with colors can be regarded as a source of noise added to the digital data pattern. We propose here a method to reduce the total amount of noise and miss-classifications introduced in the QR Code when encoding colors, that is based on the affinity of those colors to black and white (i.e. to which color it resembles the most). To that end, we classify the colors of the palette to be embedded in two groups: *pseudo-black* and *pseudo-white* colors.

Initially, let $G'_{rgb} \in [0, 255]^{l \times 3}$ be a set of colors with size l we want to embed in a QR Code. Then, let us start with the definition of the main steps of our proposal to encode these colors inside a QR Code:

1. **Normalization**, the 8-bit color channels (RGB) are mapped to a normalized color representation:

$$f_{normalize} : [0, 255]^{l \times 3} \rightarrow [0, 1]^{l \times 3} \quad (4.8)$$

2. **Desaturation**, the color channels (RGB) are then mapped into a monochromatic grayscale channel (L):

$$f_{grayscale} : [0, 1]^{l \times 3} \rightarrow [0, 1]^l \quad (4.9)$$

3. **Binarization**, the monochromatic grayscale channel (L) is converted to a monochromatic binary channel (B):

$$f_{threshold} : [0, 1]^l \rightarrow \{0, 1\}^l \quad (4.10)$$

4. **Colorization**, the binary values of the palette colors represent the affinity to black (zero) and white (one) and can be used to create a mapping between the position in the color palette list and the position inside the QR Code matrix (a binary image). This mapping will also depend on the geometry of the QR Code matrix (where are the black and white pixels placed) and an additional matrix that protects the key zones of the QR Code (a mask which defines the key zones):

$$f_{mapping} : \{0,1\}^l \times \{0,1\}^{n \times m} \times \{0,1\}^{n \times m} \rightarrow \{0, \dots, l+1\}^{n \times m} \quad (4.11)$$

Once the mapping is computed, a function is defined to finally colorize the QR Code, which renders an RGB image of the QR Code with embedded colors:

$$f_{colorize} : \{0,1\}^{n \times m} \times [0,1]^{l \times 3} \times \{0, \dots, l+1\}^{n \times m} \rightarrow [0,1]^{n \times m \times 3} \quad (4.12)$$

Subsequently, to create the pseudo-black and pseudo-white colors subsets, we must define the implementation of these functions. These definitions are arbitrary, i.e. it is possible to compute a grayscale version of a color image in different ways. Our proposed implementation is intended to resemble the QR Code readout process:

1. **Normalization**, $f_{normalize}$ will be a function that transforms a 24-bit color image (RGB) to a normalized color representation. We used a linear rescaling factor for this:

$$G_{rgb}(k, c) = f_{normalize}(G'_{rgb}) = \frac{1}{255} G'_{rgb}(k, c) \quad (4.13)$$

where $G'_{rgb} \in [0, 255]^{l \times 3}$ is a list of colors with a 24-bit RGB color depth and $G_{rgb} \in [0, 1]^{l \times 3}$ is the normalized RGB version of these colors.

2. **Desaturation**, $f_{grayscale}$ will be a function that transforms the color channels (RGB) to a monochromatic grayscale channel. We used an arithmetic average of the RGB pixel channels:

$$G_{gray}(k) = f_{grayscale}(G_{rgb}) = \frac{1}{3} \sum_{c=0}^3 G_{rgb}(k, c) \quad (4.14)$$

where $G_{rgb} \in [0, 1]^{l \times 3}$ is the normalized RGB color palette and $G_{gray} \in [0, 1]^l$ is the grayscale version of this color palette.

3. **Binarization**, $f_{threshold}$ will be a function that converts the monochromatic grayscale channel (L) to a binary channel (B). We used a simple threshold function with a thresholding value of 0.5:

$$G_{bin}(k) = f_{threshold}(G_{gray}) = \begin{cases} 0 & G_{gray}(k) \leq 0.5 \\ 1 & G_{gray}(k) > 0.5 \end{cases} \quad (4.15)$$

where $G_{gray} \in [0, 1]^l$ is the grayscale version of the color palette and $G_{bin} \in \{0, 1\}^l$ is its binary version, which describes the affinity to black (0) and white (1) colors.

4. **Colorization**, f_{color} will be a function that will render a RGB image from the QR Code binary image and the palette colors by using a certain mapping. We used this function to implement it:

$$C_{rgb}(i, j, k) = f_{color}(A_{bin}, G_{rgb}, M) = \begin{cases} A_{bin}(i, j) & M(i, j) = 0 \\ G_{rgb}(p - 1, k) & M(i, j) = p > 0 \end{cases} \quad (4.16)$$

where $A_{bin} \in \{0, 1\}^{n \times m}$ is the original QR Code binary image, $G_{rgb} \in [0, 1]^{l \times 3}$ is a color palette to be embedded in the image, $C_{rgb} \in [0, 1]^{n \times m \times 3}$ is the colorized QR Code image, and $M \in \{0, \dots, t\}^{n \times m}$ is an array mapping containing the destination of each one of the colors of the palette into the 2D positions within the image. We propose to use G_{bin} (Equation 4.15) to create M , this mapping will also depend on the geometry of the QR Code image (where are the black and white pixels placed) and an additional matrix that protects the key zones of the QR Code (a mask which defines the key zones), this mapping will be $f_{mapping}$, it has the general form:

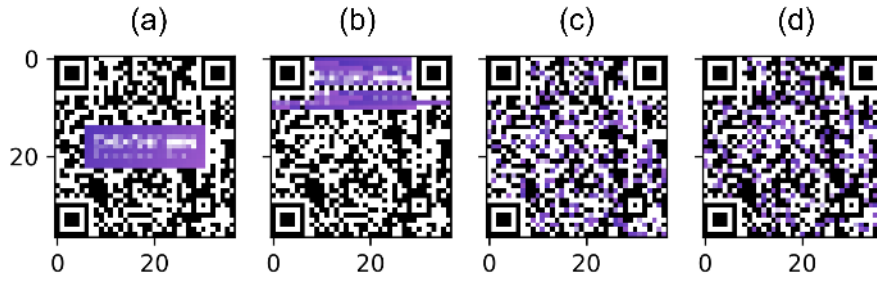
$$M = f_{mapping}(G_{bin}, A_{bin}, Z) \quad (4.17)$$

where $M \in \{0, \dots, t\}^{n \times m}$ is the array mapping, $G_{bin} \in \{0, 1\}^l$ is the affinity to black or white of each color in the palette, $A_{bin} \in \{0, 1\}^{n \times m}$ is the original QR Code binary image and $Z \in \{0, 1\}^{n \times m}$ is a mask that protects the QR Code key patterns to be overwritten by the palette. One possible implementation of $f_{mapping}$ (Equation 4.17) is shown in algorithm 1, where the colors of the palette are mapped to positions of the QR Code based on their affinity to black and white. For each one of these two classes, the particular assignment of a color to one of the many possible pixels of the class (either black or white) is fully arbitrary and allows for further design decisions. In this implementation of the mapping, we choose to assign the colors in random positions within the class. In other applications, interested e.g. in preserving a certain color order, additional mapping criteria can be used as shown below. Anyhow, preserving the assignment to the black or white classes based on the color affinity is key for back-compatibility.

Algorithm 1: Creation of the mask for grayscale insertion method

Input: $G_{bin} \in \{0,1\}^l$, $A_{bin} \in \{0,1\}^{n \times m}$, and $Z \in \{0,1\}^{n \times m}$
Output: $M \in \{0, \dots, l+1\}^{n \times m}$

- 1 $W_{color} \leftarrow []$
- 2 $B_{color} \leftarrow []$
- 3 **for** $k = 0, \dots, l$ **do**
- 4 **if** $G_{bin}(k) == 1$ **then**
- 5 Append k to W_{color}
- 6 **else**
- 7 Append k to B_{color}
- 8 $p \leftarrow \text{lenght}(W_{color})$
- 9 $q \leftarrow \text{lenght}(B_{color})$
- 10 $W_{pos} \leftarrow []$
- 11 $B_{pos} \leftarrow []$
- 12 **for** $i = 0, \dots, n$ **do**
- 13 **for** $j = 0, \dots, m$ **do**
- 14 **if** $Z(i, j) == 1$ **then**
- 15 **if** $A_{bin}(i, j) == 1$ **then**
- 16 Append (i, j) to W_{pos}
- 17 **else**
- 18 Append (i, j) to B_{pos}
- 19 $W'_{pos} \leftarrow$ Select p random values of W_{pos}
- 20 $B'_{pos} \leftarrow$ Select q random values of B_{pos}
- 21 $M \leftarrow \{0\}_{i,j} \quad \forall i \in \{0, \dots, n\}$ and $j \in \{0, \dots, m\}$
- 22 **for** $k = 0, \dots, p$ **do**
- 23 Append $M(W'_{pos}(k)) \leftarrow W_{color}(k) + 1$
- 24 **for** $k = 0, \dots, q$ **do**
- 25 Append $M(B'_{pos}(k)) \leftarrow B_{color}(k) + 1$
- 26 **return** M



Moreover, to illustrate how different placement mappings affect the readout process, we will consider 4 different situations, where $f_{mapping}$ plays different roles, and we will compute their SNR and BER metrics:

- **Logo.** When a logo-like pattern is encoded, G_{rgb} will be the colors of the logo and M_{logo} will be a mapping that preserves the logo image, overlaid on top of the original QR Code image (Figure 4.6.a).
- **Sorted.** We are going to use the colors of the logo (thus G_{rgb} will be the same as before), but we are going to place them on top of the QR Code, sorting them as they appear in the color list. M_{sorted} will establish that the first color goes to the first available position inside A_{gray} pixels, etc. (Figure 4.6.b).
- **Random.** Again we use the same colors of the logo (G_{rgb} remains the same) but now M_{random} defines a random mapping of the palette into the available positions of A_{gray} (Figure 4.6.c).
- **Grayscale.** Our proposed method. Same as before, but the now random assignment of M_{gray} respects the rule that pseudo-white colors are only assigned to white pixels of A_{gray} , and pseudo-black only to the black ones, as described in algorithm 1 (Figure 4.6.d).

Measure	Logo	Sorted	Random	Grayscale
SNR	10.53 dB	10.27 dB	10.35 dB	12.23 dB
BER	8.55 %	8.33 %	8.62 %	0.00 %

Finally, Table 4.2 shows the SNR and BER figures for the four mappings (exemplified in the images of Figure 4.6). Using the grayscale approach to encode colors by their resemblance to black and white colors leads to much lower noise levels. Since the original data of the QR Code can be seen as a random distribution of white and black pixels, M_{sorted} and M_{random} mappings yield similar results to M_{logo} , encoding the logo itself. Meanwhile, M_{gray} mapping shows us a 0% BER, and an almost 2dB SNR increase. This suggests that our proposal is an effective way to embed colors into QR Code in a back-compatible manner (see Figure 4.2), as it is demonstrated in the following sections.

Figure 4.6: The color information from the ColorSensing logo is distributed using different criteria, each one of these distributions compute different measures of SNR and BER, although the total amount of colors is the same, the way they are distributed affects the signal quality. (a) The original QR Code with the logo. (b) The logo colors are sorted at the top of the QR Code. (c) The logo colors are randomly distributed among the QR Code. (d) The logo colors are distributed by using a threshold criterion among blacks and white colors.

Table 4.2: Values of SNR and BER computed for each criteria in Figure 4.6. Using the logo as it is, the sorted criteria and random criteria yield to similar results. However, the use of a simple grayscale threshold criteria slightly increases the SNR and hugely depletes the BER, showing a good result for encoding colors in a back-compatible way.

4.2 Experimental details

Experiments were designed to test our proposed method, we carried out 3 different experiments where QR Codes were filled with colors and then transmitted through different channels. In all experiments, we calculated the SNR and BER as a measure of the signal quality of each QR Code once transmitted through different channels. Also, we checked the direct readability by using a QR Code scanner before and after going through the channels. Table 4.3 contains a summary of each experiment designed. A detailed explanation of the experimental variables is provided below.

All experiments	Values	Size
Color substitution (%)	1, 5, 10, 15, 20, 30, 40, 50, 60, 70, 80, 100	12
Colorized zone	EC, D, EC&D	3
Colorizing method	Random, Grayscale	2
Experiment 1	Values	Size
Digital IDs	from 000 to 999	1000
QR version	5, 6, 7, 8, 9	5
Channels	Empty, Image augmentation	1 + 1
Experiment 2	Values	Size
Digital IDs	000	1
QR version	5, 6, 7, 8, 9	5
Channels	Empty,	
Image augmentation	1 + 1000	
Experiment 3	Values	Size
Digital IDs	000	1
QR version	5	1
Channels	Empty, Colorimetry setup	1 + 25

Table 4.3: Summary of parameter values for each experiment designed. All experiments share common parameters, at least each experiment has 72 different QR Codes that will be generated using as reference the multiplication of the shared parameters. Experiment 1 generates 360.000 different QR Codes.

4.2.1 Color generation and substitution

We choose our random color palette G_{rgb} for the experiments to be representative of the RGB space. Nevertheless, G_{rgb} should be random in a way that it is uniformly random in the grayscale space L. But if we define three uniform random RGB channels as our generator, we will fail to accomplish a grayscale uniform random channel. This is due to the fact that when computing the L space as a mean of the RGB channels, we are creating a so-called Irwin-Hall uniform sum distribution [133] (see Figure 4.7.b.). In order to avoid this, we propose to first generate the L channel as a uniform random variable, then generate RGB channels which will produce these generated L channel values (see Figure 4.7.b.).

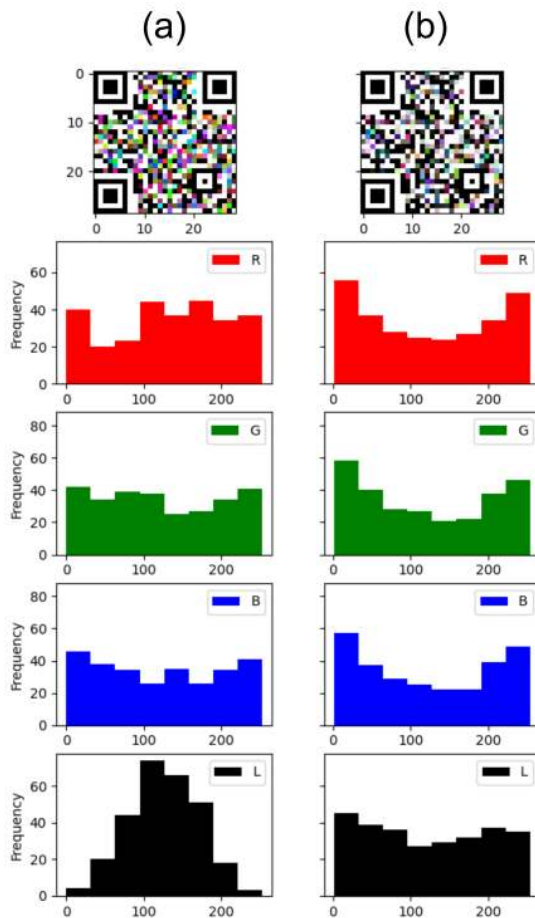


Figure 4.7: Histogram comparison between uniform randomly generated RGB channels. (a) which yields to a non-uniform grayscale -L- and uniform randomly generated grayscale -L-. (b) with derived pseudo-uniform RGB channels.

During the different experiments, we will be filling QR Codes with a palette of random colors G_{rgb} . The color substitution factor ranged from only 1% of the available pixel positions in a QR Code replaced with colors up to 100% (see Figure 4.8). Evidently, each QR Code version offers different numbers of pixels and thus positions available for color substitution.



Figure 4.8: The same QR Code is populated with different amounts of colors. (a) 1% of the pixels are substituted using a random placement method (yellow arrows show the colored pixels). (b) 100% of the pixels are substituted using a random placement method.

4.2.2 Placing colors inside the QR Code

Our back-compatibility proposal starts avoiding the substitution of colors in key protected areas of the QR Code. This can be implemented with a Z mask (see algorithm 1). In our experiments, we used 3 masks (see Figure 4.9):

- I $Z_{EC\&D}$, that excludes only the key protected areas and allows covering with colors all the error correction and data regions (see details in chapter 2),
- II Z_{EC} , that only allows embedding colors in the error correction,
- III Z_D , with colors only in the data.

Once we have restricted ourselves to these Z masks, we will embed the colors following a M mapping. We propose to use M_{random} and M_{gray} presented before (see Figure 4.6.c. and Figure 4.6.d.).



Figure 4.9: The same QR Code is populated in different areas with 80% of colors for each area. (a) the whole QR Code is populated (EC&D). (b) Only the error correction area is populated (EC). c. Only the data area is populated.

4.2.3 QR Code versions and digital IDs

The encoded data and the version of the QR Code will shape the actual geometry of the barcode, thus it will determine the A_{gray} pixels. To generate the barcodes, we choose as payload data a URL with a unique identifier such as <https://color-sensing.com/#000>, where the numbers after '#' range from 000 to 999 to make the barcodes different from each other. Also, the QR Code selected versions ranged from 5 to 9, to test and exemplify the most relevant computer vision pattern variations defined in the QR Code standard. For all of these barcodes, we used the highest level of error correction of QR Code standard: the H level, which provides a 30% of error correction.

4.2.4 Channels

The use of QR Code in real world conditions imply additional sources of error, like differences in printing, different placements, ambient light effects, effects of the camera and data processing, etc. All these factors can be regarded as sources of noise in a transmission channel.

We considered 3 different channels for the experiments:

- **Empty.** A channel where there is no color alteration due to the channel. It was used as a reference, to measure the noise level induced by the colorization process (see Figure 4.10.a).
- **Image augmentation.** With a data augmentation library [121] we generated images that mimic different printing processes and exposure to different light conditions. With this tool we also applied Gaussian blur distortions, crosstalk interferences between the RGB channels and changed contrast conditions. (see Figure 4.10.b).
- **Colorimetry setup.** We actually printed the QR Codes and captured them with a fixed camera (Raspberry Pi 3 with a Raspberry Pi Camera v2) [134] under different illumination-controlled conditions (Phillips Hue Light strip) [135]. The camera was configured to take consistent images. The light strip was configured to change its illumination conditions with two subsets of illumination conditions: white light (9 color temperatures from 2500K to 6500K) and colored light (15 different colors sampling evenly the CIE_xy space) (see Figure 4.10.c).

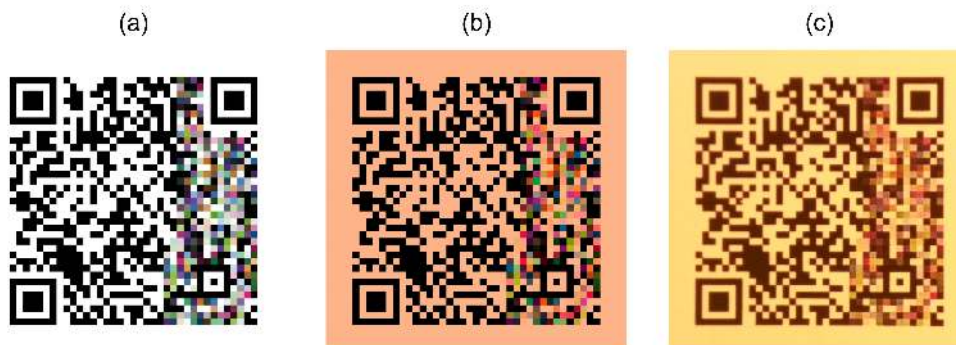


Figure 4.10: The same QR Code with data and the same amount of colors (80% of the data area) is exposed to different channels. (a) The image passed-through an empty channel. (b) The image passed-through an augmentation channel which resembles a warm light scene. (c) The image passed-through a real environment channel, actually printed and captured in a scene with a lamp at 2500K (warm light).

4.3 Results

4.3.1 Embedding colors in QRs codes: empty channel

Let us start with the results of Experiment 1, where 360.000 different color QR Codes were encoded (see Table 4.3). Then, SNR and BER were computed against an empty channel (only the color placement was taken into account as a source of noise). Results show only data from those QR Codes where colors were placed using the $Z_{EC\&D}$ mask (see details in subsection 4.2.3), reducing our dataset to 120.000 QR Codes. Figure 4.11 shows aggregated results of the SNR and BER as a function of the color substitution ratio, for M_{random} and M_{gray} mappings data is averaged for all QR Code versions (5, 6, 7, 8 and 9) and for all 1000 different digital IDs. These results indicate that SNR and BER are independent of the QR Code versions and the QR Code digital data, since the standard deviation of these figures (shadow area in Figure 4.11) that average different versions and digital IDs are very small. Only the BER for M_{random} shows a narrow deviation. Of course, all these deviations increased when noise was added (see further results).

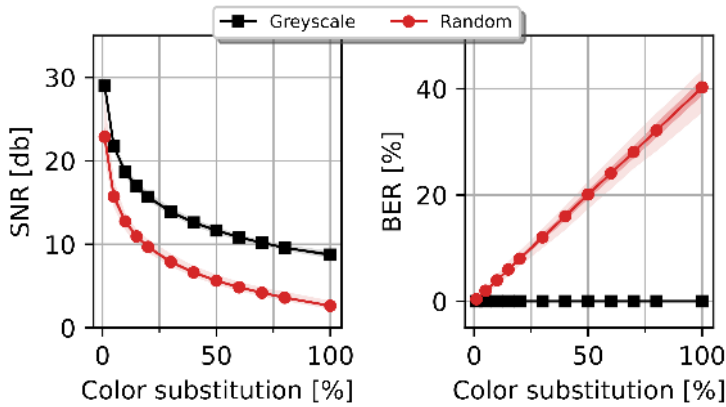


Figure 4.11: SNR and BER results for Experiment 1 before sending the QR Codes to any channel, only taking into account the QR Codes where all the area has been used (EC&D). Lines and points show average data, light shadows show the min and max values, and heavy shadows show the standard deviation for each color substitution ratio. Left: SNR results for Greyscale (squares, black) and Random (dots, red) methods. Right: BER results for Greyscale (squares, black) and Random (dots, red) methods.

Regarding the SNR, it decreases for both M_{random} and M_{gray} when the total amount of colors increases. We found that our M_{gray} proposal (affinity towards black and white) is 6 dB better than M_{random} , regardless of the quantity of colors embedded, the data, or the version of the QR Code. This means that our proposal to place colors based on their grayscale value is 4 times less noisy than a random method.

Concerning the BER, results show that, before including the effects of a real noisy channel, our placement method leads to a perfect BER score (0%). Instead, with a random substitution, and even in an ideal channel, BER increases linearly and reaches up to a 40% of BER. Taking into account the QR Code resemblance to a pseudo-random pattern, the maximum BER in this scenario is 50%. This slightly better result can be attributed to the fact that we are not tampering with the key protected areas of the QR Code (finder, alignment, ...).

4.3.2 Image augmentation channel

Results from Experiment 1 showed that the SNR and BER results are independent from the data encoded in the QR Code. Based on this finding, we reduced the amount of different IDs encoded to only one per QR Code version and increased the number of image augmentation channels to 1000. This was the key idea of Experiment 2, and by doing this we achieved the same statistics of a total 360.000 results, from 3.600 QR Codes sent through 1.000 different channels. Focusing again only on the QR Codes that are color embedded using the whole zone ($Z_{EC\&D}$) we ended up with 120.000 results to calculate the corresponding SNR and BER (see Figure 4.12).

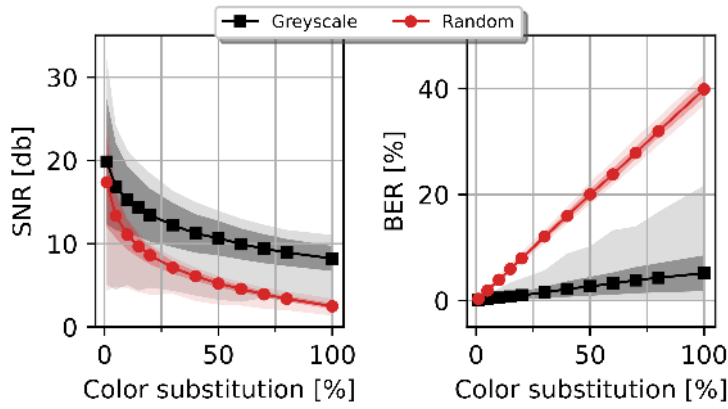


Figure 4.12: SNR and BER results for Experiment 2 after sending the QR Codes to an image augmentation channel, only taking into account the QR Codes where all the area has been used (EC&D). Lines and points show average data, light shadows show the min and max values, and heavy shadows show the standard deviation for each color substitution ratio. Left: SNR results for Greyscale (squares, black) and Random (dots, red) methods. Right: BER results for Greyscale (squares, black) and Random (dots, red) methods.

Regarding the SNR, it worsened in comparison with Experiment 1, because now the image augmentation channel is adding noise (see details in subsection 4.2.4). The 6 dB difference between M_{random} and M_{gray} remains for higher color substitution. This can be explained because the noise generated by the color placement is larger than the noise generated by the channel when increasing the amount of colors.

Concerning the BER, it increased up to an average value of about 7% for M_{gray} method due to the influence of a noisy channel. In the most extreme cases (channel with the lowest SNR and for the maximum color substitution ratio), BER values do not exceed 20%. Instead, the augmentation channel does not seem to increase the BER for M_{random} ; essentially because it is already close to the theoretical maximum.

We have also observed (see Figure 4.13) the impact of the channel noise on the SNR and BER figures of M_{random} and M_{gray} are mostly independent of the QR Code version. Therefore, we can expect that the level of resilience to noise offered by one or another mapping will remain, independently of the data to encode or the QR Code version needed. That is the reason why we removed the QR Code version from the set of variables to explore in the Experiment 3.

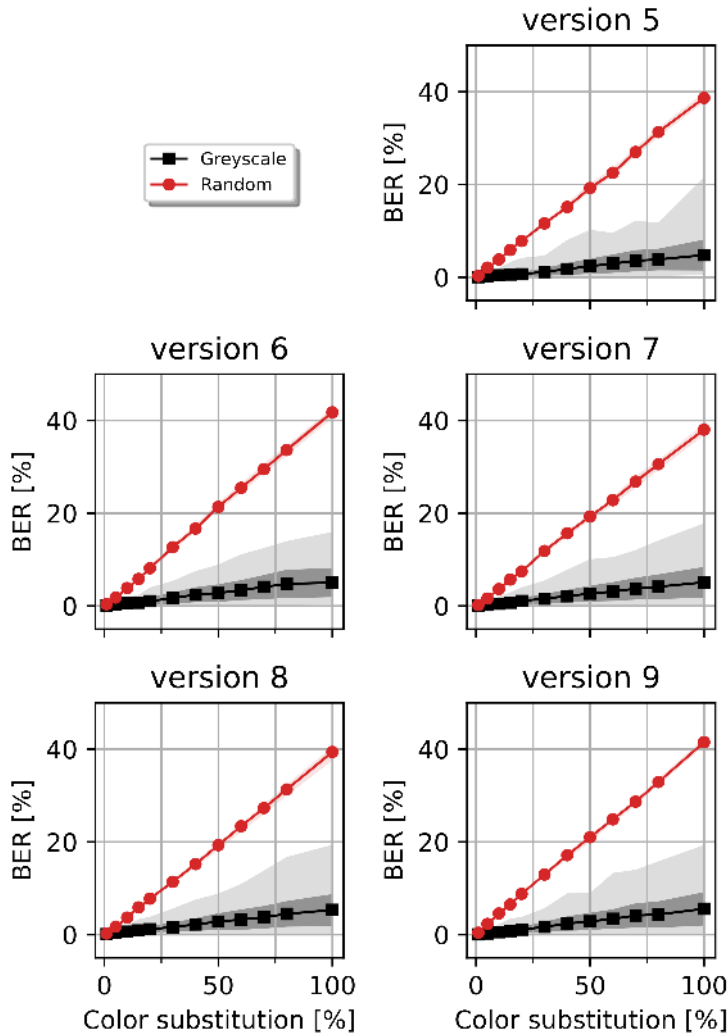


Figure 4.13: SNR results for Experiment 2, splitted by QR Code version, after sending the QR Codes to an image augmentation channel, only taking into account the QR Codes where all the area has been used (EC&D). SNR results are shown for Greyscale (squares, black) and Random (dots, red) methods. Lines and points show average data, light shadows show the min and max values, and heavy shadows show the standard deviation for each color substitution ratio.

4.3.3 Colorimetry setup as channel

Experiment 3 consisted of only one QR Code v5 (1 ID, 1 version) being colored in 72 different ways (12 color insertion ratios, 2 color placement mappings $-M_{random}$ and M_{gray} — and 3 different zones to embed colors $-Z_{EC\&D}$, Z_{EC} and Z_D —), then printed and exposed to a colorimetry setup with a total of 25 different color illumination conditions captured with a digital camera. We performed this experiment as a way to check if the proposed method and the results obtained with the image augmentation channel held in more severe, and real, capturing conditions. This experiment led to a dataset of 1.800 images acquired from the real world. The calculations of the SNR and the BER were based on those images with colors placed with the $Z_{EC\&D}$ mask, reducing our dataset to 600 results (see Figure 4.14).

Regarding the SNR, as our real channel was quite noisy, averaged values sank more than 10 dB, for all the color substitution ratio and for M_{random} and M_{gray} . Here, the huge advantage of 6dB observed

before for M_{gray} was not so evident, since the channel was now the main source of noise. This should serve to illustrate that our proposed method starts with an initial advantage in ideal conditions with respect to the random mapping method, which can diminish due to the channel noise but will always perform better.

Regarding the BER, for M_{gray} , the BER values did not increase relative to the image augmentation channel, both distributions overlap in the range of 7-10% of BER. For M_{random} , the linear maximum behaviour up to a BER of 40% is also shown in this situation. As shown in further sections, although noise levels from both methods are similar in practical applications, the difference in how they are translated into BER determines the grayscale mapping better performing.

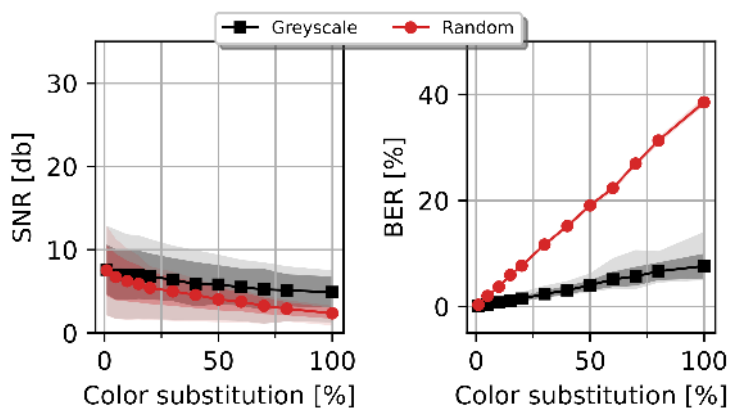


Figure 4.14: SNR and BER results for Experiment 3 after sending the QR Codes to a real channel (printing and capturing the QR Code in a colorimetry setup), only taking into account the QR Codes where all the area has been used (EC&D). Lines and points show average data, light shadows show the min and max values, and heavy shadows show the standard deviation for each color substitution ratio Left: SNR results for Greyscale (squares, black) and Random (dots, red) methods. Right: BER results for Greyscale (squares, black) and Random (dots, red) methods.

4.3.4 Readability

Up to this point, results show how embedding colors in the QR Codes might increase the probability of encountering bit errors when decoding those QR Codes. Results also indicate that our back-compatible method can reduce the average probability of encountering a bit error from 40% to 7-10%, enabling proper back-compatible QR Code scan using the error correction levels included in the standard that can tolerate this amount of error (levels Q and H). This is a necessary but not sufficient demonstration of back-compatibility.

We must also be sure that the new method offers QR Codes fully readable with conventional decoders. To assess this readability, we checked the integrity of the data of all the QRs in our experiments using ZBar, a well-established barcode scanner often used in the literature [56; 122]. We calculated the success ratio at each color substitution ratio as the amount of successfully decoded QR Codes by ZBar divided by the total amount of QR Codes processed. Also, we analyzed separately the results obtained when embedding colors in the 3 different zones ($Z_{EC\&D}$, Z_{EC} and Z_D masks), in order to identify further relevant behaviours.

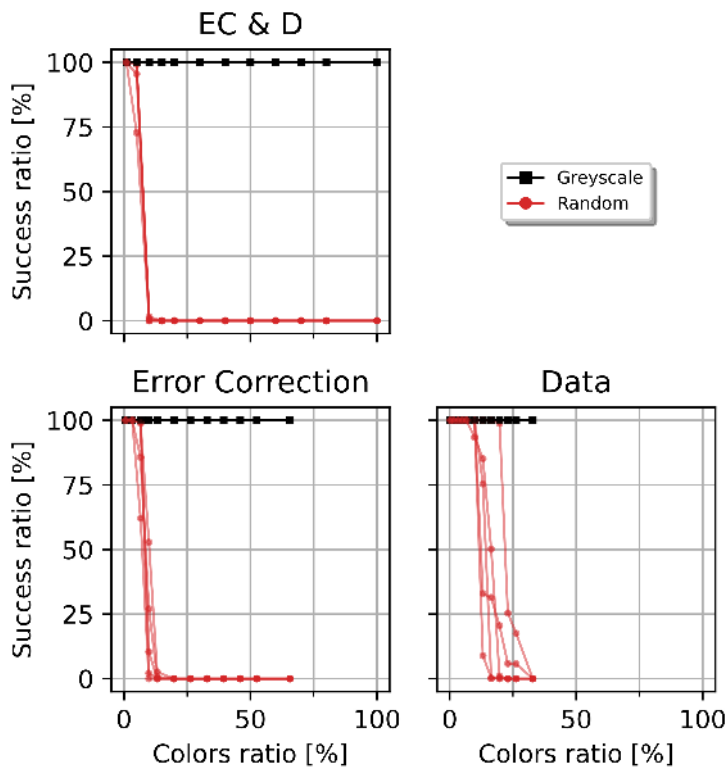


Figure 4.15: Success ratio of decoded QR Codes before passing through a channel among different embedding zones (EC&D, Error Correction and Data), for each color mapping method (greyscale and random) for all QR Code versions. Each curve represents a QR Code version, there are up to 5 curves for each method, Greyscale (squares, black) and Random (dots, red).

On the one hand, readability results of the QR Codes of Experiment 1 (channel without noise) are shown in Figure 4.15. M_{gray} , the proposed mapping, scores a perfect readability, no matter the insertion zone. This is because M_{gray} does not actually add BER

when colors are inserted. Instead, M_{random} , the random method, is extremely sensitive to color insertion, and the readability success rate decays rapidly as the number of inserted colors increases. As seen in Experiment 1, the Data zone (Z_D) seems the most promising to embed the largest fraction of colors.

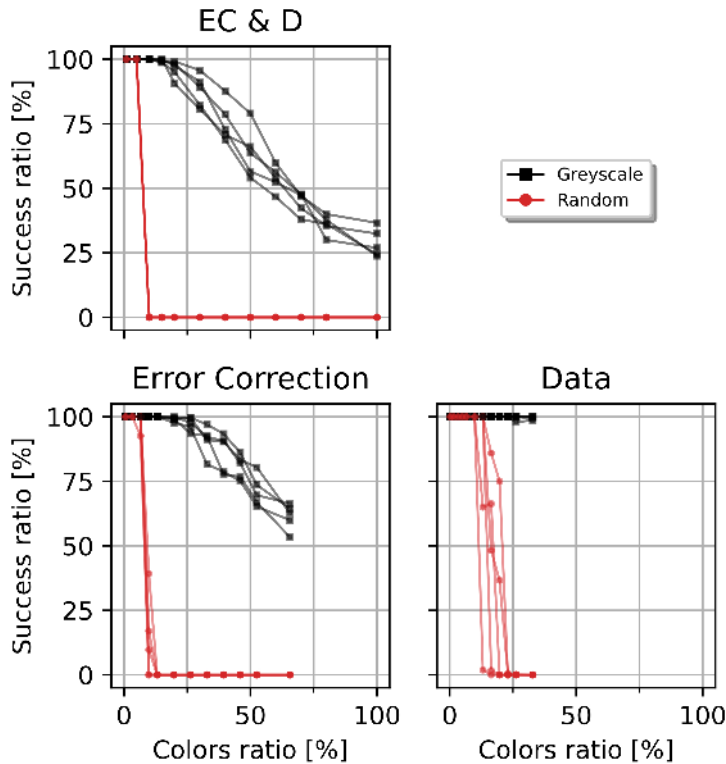


Figure 4.16: Success ratio of decoded QR Codes after passing through an image augmentation channel among different embedding zones (EC&D, Error Correction and Data), for each color mapping method (greyscale and random) for all QR Code versions. Each curve represents a QR Code version, there are up to 5 curves for each method, Greyscale (squares, black) and Random (dots, red).

On the other hand, results after passing through the noisy channels of Experiment 2 and Experiment 3 are shown in Figure 4.16 and Figure 4.17, respectively. Clearly, the noise of the channel also affects the readability of M_{gray} mapping, but the codes built this way are much more resilient and can allocate a much larger fraction of colors without failing. Even more: if colors are only placed in the Data (Z_D) encoding zone, grayscale mapped color QR Codes remain fully readable until all the available pixels of the zone are occupied.

To get a practical outcome of these results, one should translate the color substitution ratios into the actual amount of colors that these ratios mean when using different encoding zones in QR Codes with different versions. Table 4.4 summarizes these numbers grouped by encoding zone, QR Code version. Results compare the maximum number of colors that can be allocated using each one of the mapping methods (grayscale vs. random) with at least a 95% of readability. Experience shows that beyond this 5% of failure, the user experience is severely damaged.

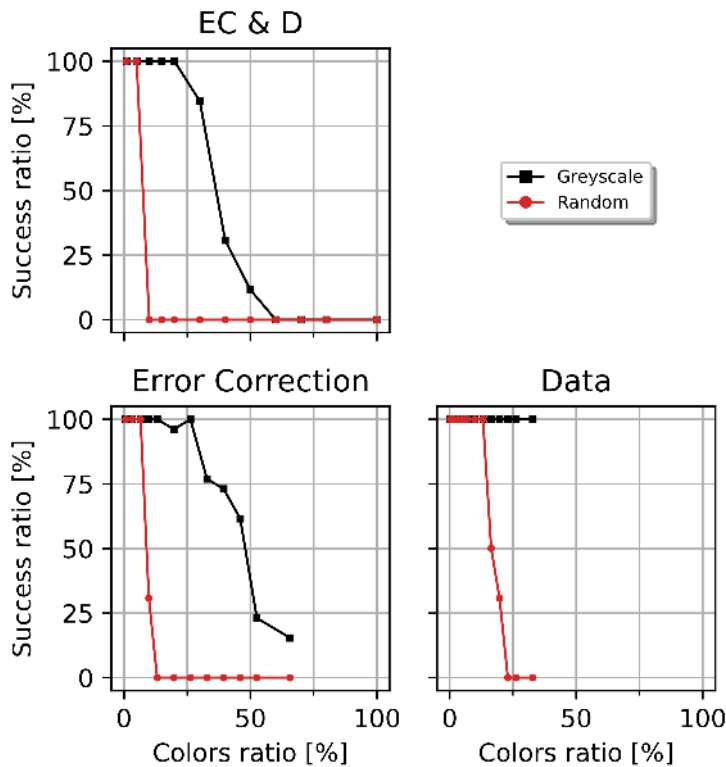


Figure 4.17: Success ratio of decoded QR Codes after passing through a real-life channel among different embedding zones (EC&D, Error Correction and Data), for each color mapping method, Greyscale (squares, black) and Random (dots, red), only for a QR Code of version 5.

Clearly, the M_{gray} mapping allows for allocating between 2x to 4x times more colors than a naive M_{random} approach. Interestingly, restricting the placement of colors to the data zone (Z_D) leads to a much larger number of colors, in spite of having less pixels available; being the error correction (Z_{EC}) the less convenient to tamper with. In the best possible combination (M_{gray} mapping, Z_D zone, v9 –largest version studied–) this proposal reaches an unprecedented number of colors that could be embedded close to 800. As a matter of fact, this could mean sampling a 3-dimensional color space of 24 bits resolution (i.e. sRGB) with 9^3 colors evenly distributed along each axis.

Needless to say, that such figures can be systematically increased with QR Codes of higher versions. To get a specific answer to the question of how many colors can be embedded as a function of the QR Code version in the best possible conditions -data zone (Z_D) with our grayscale mapping (M_{gray})-, we generated a specific dataset of QR Codes with versions running from v3 to v40, and checked their 95% readability in the conditions of Experiment 2 through 50 image augmentation channels (see Figure 4.18). Results indicate that thousands of colors are easily at reach, with a theoretical maximum of almost 10,000 colors with QR Codes v40. In real life, however, making these high version QR Codes readable with conventional cameras at reasonable distances means occupying quite a lot of space (about 5 inches for a QR Code v40). That size, though, is comparable to that of a ColorChecker pattern but giving access to thousands of colors instead of only tens.

QR Code version	EC&D		Error Correction		Data	
	Greyscale	Random	Greyscale	Random	Greyscale	Random
5	322	54	282	70	352	141
6	206	69	448	90	464	139
7	314	78	520	104	512	205
8	387	97	499	125	672	269
9	467	117	461	77	784	235

Table 4.4: Number of different colors that can be embedded inside a QR Code with a 95% success ratio during the decoding process for each insertion mask (EC&D, EC or D), for both color mapping methods (greyscale and random). In absolute terms, the mask corresponding with only the Data zone beats the other two, as expected the Grayscale method performs better than the Random one.

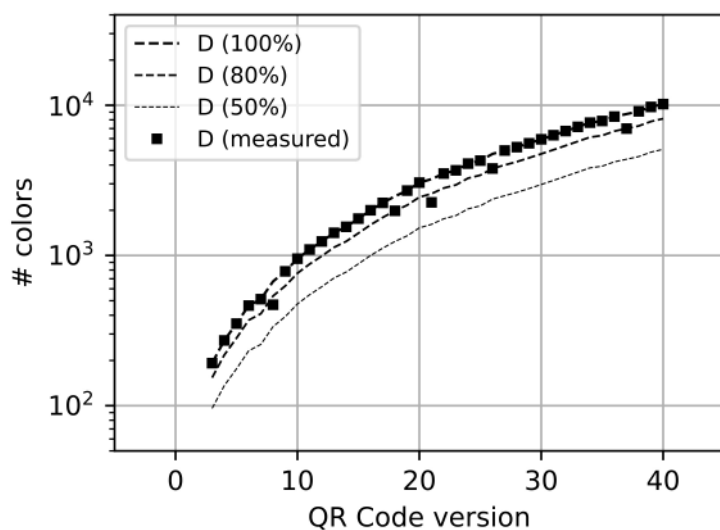


Figure 4.18: Number of colors that can be embedded in the D zone as a function of the QR Code version (from v3 to v40). Lines show the theoretical maximum number of colors, for different substitution ratios. Square dots show the maximum number of colors that could be embedded in a QR Code with a demonstrated readability above 95% in the conditions of Experiment 2. In contrast to the other QR Code zones, such high readabilities are obtained, even in 100% substitution ratio only in the D zone.

4.3.5 Example of use case

Finally, we illustrate how this approach can be applied to carry out actual color correction problems with full QR Code back-compatibility, using the 24 colors from the original ColorChecker [13] to create a barcode that contains them (see Figure 4.19). We created a compact color QR Code version 5 with H error correction level. According to our findings, this setup should let us embed 352 colors in the data zone (Z_D) zone without risking readability. In this example, this allowed us to embed up to 10 replicas of the 24 color references, offering plenty of redundancy to detect variation of the color calibrations across the image or to improve the correction itself. Table 5 shows the main quantitative results obtained with this colored QR Code, submitted to the conditions of Experiment 2, with one empty channel and 120.000 image augmentation channels.

Encoding		
Digital ID	000	
Version	5	
Error correction level	H	
Unique colors	24	colors
Total embedded colors	240	colors
Color substitution ratio	22	%
Empty channel		
SNR	12.68	dB
BER	0.0	%
Success ratio	100	%
Augmentation channels		
SNR	11 ± 2	dB
BER	2.7 ± 1.7	%
Success ratio	96	%

Table 4.5: Properties of the proposed QR Code with the ColorChecker colors embedded in it. Properties are related with different steps in the QR Code life-cycle, from encoding to decoding.



Figure 4.19: A color QR Code (version 5 with H error correction level) which contains 240 pixels that are coloured. This is implemented with our back-compatible method. These color pixels reproduce the 24 original ColorChecker colors with a redundancy of 10 pixels per color. Only 22% of the digital data pixels are used in this process, almost all the Data (D) zone is used to allocate the colors.

4.4 Conclusions

We have presented a method to pack a set of colors, useful for color calibration, in a QR Code in a fully back-compatible manner, this is preserving its conventional ability to store digital data. By doing this, we enhanced the state-of-the-art color charts with two main features: first, we did leverage the computer vision readout of the color references to the QR Code features; and two, *de facto* we reduced the size of the color charts to the usual size of a QR Code, one or two inches.

Also, we have demonstrated that the color capacity of the QR Codes constructed this way (up to a few thousand colors!) is orders of magnitude higher than that found in the traditional color charts, due to the image density and pattern recognition robustness of the QR Codes.

Moreover, compared to other colored QR Codes, our proposal, based on the grayscale affinity of the colors to white or black, leads to much lower signal alteration levels and thus much higher readability, than the found in more naive approaches like, e.g., random assignment methods, which represent the aesthetic QR Codes (printing a logo).

This work opens a way to explore further methods to embed color information in conventional 2D barcodes. Tuning how we defined our criteria of color embedding upon the affinity of colors to black and white would lead to more efficient embedding methods. We explored some of these ideas to seek those improved methods, and we expose them below.

First, the way in which we implemented the grayscale (a mean value of the RGB channels) is only one of the ways to compute a grayscale channel, i.e. one could use the *luma* definition a weighted mean based on the human eye vision:

$$f_{grayscale}(r, g, b) = 0.2126 \cdot r + 0.7152 \cdot g + 0.0722 \cdot b .$$

Or the *lightness* one:

$$f_{grayscale}(r, g, b) = \frac{1}{2} (\max(r, g, b) + \min(r, g, b)) .$$

These grayscale definitions are often part of colorspace definitions [11], such as CIE Lab, CIE Luv, HSL, etc. All these different grayscale will generate different color distributions, displacing colors between black and white regions of the QR Code.

Second, we defined a way to select the area inside the QR Code where to embed the colors (see algorithm 1), this could be improved also. For example, we could decide to implement $f_{threshold}$ in a more complex fashion. Let us imagine a certain set of colors G_{rgb} to encode in a certain QR Code, one could create more than two sets to define the black-white affinity, i.e. four sets, namely: *blackest* colors (0), *blackish* colors (1), *whitish* colors (2) and *whitest* colors (3):

$$f_{threshold}(G_{gray}) = \begin{cases} 0 & 0.00 < G_{gray}(k) \leq 0.25 \\ 1 & 0.25 < G_{gray}(k) \leq 0.50 \\ 2 & 0.50 < G_{gray}(k) \leq 0.75 \\ 3 & 0.75 < G_{gray}(k) \leq 1.00 \end{cases}$$

And accommodate algorithm 1 to this new output from $f_{grayscale}$ by assigning those colors with higher potential error (1,2) to the DATA zone and those with lower potential error (0,3) to the EC zone. Theoretically, this would outperform our current approach, as in this work we demonstrated that DATA zones are more resilient to error than EC zones, thus displacing away from EC critical colors would lead to a systematic increase of color capacity.

Third, many authors have contributed to create aesthetic QR Codes which embed trademarks and images with incredibly detailed results, we wanted to highlight some solutions that might be combined with our technology to embed even further colors or improve the control over the placement of the colors.

Halftone QR Codes were proposed by Chu et al. [56], they substituted QR Code modules for subsampled modules that contained white and black colors. These submodules presented a dithering pattern that followed the encoded image shape. One could use the dithering idea to embed also colors inside subsampled pixels.

QArt Codes were introduced by Cox [58], the proposal aims to force the QR Code encoding to block certain areas of the QR Code to be black or white no matter what the encoded data is (note this is only possible for some kinds of data encoding). One could use this feature to preserve dedicated areas for color embeddings, as a complement to algorithm 1. Note the need for a back-compatible criteria is still-present, the QArt Code only provides us the certainty if a module of the original QR Code is black or white, but they *must remain* black or white during the decoding process. This combination of technologies has a potential impact in reducing the cost of producing the Color QR Codes. as one could fix the position of the colored modules before encoding the data using our grayscale criteria, then print the QR Code using cost-effective printing technologies (rotogravure, flexography, etc.) and the black and white pixels using also cost-effective monochromatic printing technologies (laser printing), rather than use digital ink-jet printing to print the whole code.

Finally, we have assumed that our *back-compatible Color QR Codes* are meant to be for colorimetric measurement, as this is the preamble of our research.

Nevertheless, the above-presented results could be applied to encode data in the color of the QR Codes in a back-compatible manner. This means, include more digital information in the same space. Other authors have presented their approaches to this solution, none of them in a back-compatible way. Let us propose a couple of ideas to achieve these *digital back-compatible Color QR Codes*, based on other ideas to color encode data in QR Codes.

First, Blasinki et al. [130] introduced a way to multiplex 3 QR Codes in one, by encoding a QR Code in each CMY channel of the printed image. And then, when they recovered the QR Code, they applied a *color interference cancellation algorithm* to extract the QR Codes from the RGB captured image, they also discussed how to approach the color reproduction problem and manipulated further the remaining black patterns (finder, alignment and timing) to include color references.

All in all, this rendered non back-compatible Color QR Codes. Now, using our proposed method, one could turn this approach to back-compatibility again by simply do the following: keep the first QR Code to multiplex as the 'default' QR Code; then, take another 3 additional QR Codes and create a barcode following Blasinki et al. method; in turn, create a "pseudo-white" and "pseudo-black" version of this color QR Code; and finally, re-encode the default QR Code with the pseudo-colors in a back-compatible manner. Also, note this proposal is not restricted by the number of channels an image has, as we are exploiting intermediate values, not only the extreme ones. There should exist a limit yet to discover of how many QR Code can be multiplexed in this fashion.

Second, other authors like Berchtold et al. – JAB Code –[49] or Grillo et al. – HCCBC –[136] fled from the original QR Code standard to redefine entirely the data encoding process. The main caveat of their technological proposals is the lack of back-compatibility, as we have discussed before. One could combine both technologies to create more adoptable technology. Grillo et al. proposal seems the easiest way to go, as they kept the factor form of QR Codes. Theoretically, one could simply multiplex one HCCBC with one QR Code as described with the previous method and achieve a digital back-compatible Color QR Code.

Chapter 5. Image consistency using an improved TPS_{3D} method

Thin-plates splines (TPS) were introduced by Duchon in 1978 [137], reformulated by Meinguet in 1979 [138]. Later in 1989, TPS were popularized by Bookstein [115] due to their potential to fit data linked by an elastic deformation, especially when it comes to shape deformation.

So far, TPS have been used widely to solve problems like morphing transformations in 2D spaces. For example, Rohr et al. [139] and Crum et al. [140] used TPS to perform elastic registration of similar features in different images in a dataset. Or, Bazen et al. [116] used TPS to match fingerprints. Moreover, we have successfully used TPS to improve QR Code extraction in chapter 3.

TPS framework can be applied to color spaces. As explained in chapter 2, color spaces are three-dimensional spaces. TPS formulation covers this scenario for 3D deformations [137; 138; 115]. In fact, we can already find works that apply them to color technology. For example, Colatoni et al. [141] and Poljicak et al. [142] used TPS to characterize screen displays. Also, Sharma et al. [143] interpolated colors to be printed in commercial printers using TPS. Moreover, Menesatti et al. [15] proposed a new approach to color correct dataset for image consistency based upon TPS, and called this method *3D Thin-Plate Splines (TPS_{3D})*.

In this chapter, we focus on the implementation of thin-plate spline color corrections, specifically in the use of the TPS_{3D} method to perform color correction in image datasets directly in RGB spaces, while proposing an alternative radial basis function [144] to be used to compute the TPS, and introducing a smoothing factor to approximate the solution in order to reduce corner-case errors [145]. All in all, we illustrate here the advantages and limitations of the new TPS_{3D} methodology to color correct RGB images.

5.1 Proposal

Solving the image consistency problem, introduced in chapter 2, using the TPS_{3D} method requires the creation of an image dataset. The images on this dataset must contain some type of color chart. Also, the captured scenes in the images must be meaningful, and the color chart must be representative of those scenes. Here, we propose to use the widely-accepted Gehler’s ColorChecker dataset [146; 147], which contains 569 images with a 24 patch Macbeth ColorChecker placed in different scenes (see Figure 5.1). We do so, rather than creating our own dataset with the Back-compatible Color QR Codes proposed in chapter 4 because this is a standard dataset with a standard color chart. In chapter 6, we will combine both techniques into colorimetric sensors.

Moreover, we propose to apply a data augmentation technique to increase the size of the dataset by 100, to match in size other augmented image datasets that have appeared recently. We do not use those images as they do not always contain a ColorChecker [12].

Furthermore, we propose to benchmark our TPS_{3D} against its former implementation [15] and a range of alternative methods to correct the color in images, such as: white balance [44], affine [44], polynomial [27; 28] and root-polynomial [28] corrections. Benchmarking includes both quantitative color consistency and computational cost metrics [15; 148].

In this section we review the derivation of the above-mentioned color correction methods before introducing our improvements to the TPS_{3D} method. Notice the formulation will remind to the 2D projection formulation from chapter 3, however some differences have to be considered:

- as described in chapter 2 color corrections are **mappings between 3D spaces**,
- unlike projective transformations in 2D planes, we will be **only using affine terms** as basis, and
- our notation in this formulation avoids the use of homogeneous coordinates (p_0, p_1, p_2) in favor of more verbose notation using the **name of each color channel** (r, g, b) .



Figure 5.1: An example of a Gehler’s ColorChecker dataset image.

5.1.1 Linear corrections

In chapter 2 we defined color corrections as an application f between two RGB color spaces. If this application is to be linear, thus a *linear correction*, we can use a matrix product notation to define the correction [14; 44]:

$$\mathbf{s}' = f(\mathbf{s}) = \mathbf{M} \cdot \mathbf{s} \quad (5.1)$$

where \mathbf{M} is a 3×3 linear matrix that maps each color in the origin captured color space $\mathbf{s} = (r, g, b)$ to the corrected color space $\mathbf{s}' = (r', g', b')$ (see Figure 5.2). In order to solve this system of equations, we must substitute these vectors by matrices containing enough color landmarks (known pairs of colors in both spaces) to solve the system for \mathbf{M} :

$$\mathbf{M} \cdot \mathbf{P} = \mathbf{Q} \quad (5.2)$$

where \mathbf{Q} is matrix with \mathbf{s}' colors and \mathbf{P} is matrix with colors \mathbf{s} .

5.1.1.1 White-balance correction

White-balance is the simplest color transformation that can be applied to an RGB color. In the white-balance correction each channel of vector function f is independent:

$$\begin{aligned} r' &= f_r(r) = \frac{r_{max}}{r_{white}} \cdot r \\ g' &= f_g(g) = \frac{g_{max}}{g_{white}} \cdot g \\ b' &= f_b(b) = \frac{b_{max}}{b_{white}} \cdot b \end{aligned} \quad (5.3)$$

where $(r_{max}, g_{max}, b_{max})$ is the maximum value of each channel in the color corrected space, e.g. $(255, 255, 255)$ for 24-bit images; and $(r_{white}, g_{white}, b_{white})$ is the measured whitest color in the image (see Figure 5.3). This relation can be easily written as a matrix, and only needs one color reference to be solved (from Equation 5.2):

$$\begin{pmatrix} a_r & 0 & 0 \\ 0 & a_g & 0 \\ 0 & 0 & a_b \end{pmatrix} \cdot \begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} r' \\ g' \\ b' \end{pmatrix} \quad (5.4)$$

where a_k are the weight contributions of Equation 5.3 for each k channel.

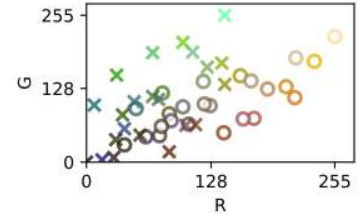


Figure 5.2: RGB colors of the ColorChecker of an image are projected in the *red-green* plane. The colors are replicated: (\circ) show the original colors of the ColorChecker and (\times) show their augmented version, as in their captured values.

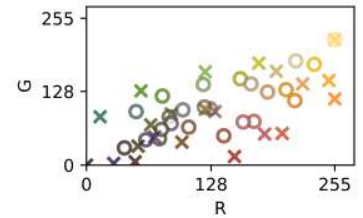


Figure 5.3: RGB colors of the ColorChecker of an image are projected in the *red-green* plane. The colors are replicated: (\circ) show the original colors of the ColorChecker and (\times) show the corrected values of the augmented version shown in Figure 5.2 using a *white-balance correction*. The whitest point (upper right) is the only one that is properly corrected.

The white-balance correction can be improved by subtracting the black level of the image before applying the white-balance correction (see Figure 5.4). For example, shown for the red channel for simplicity, this improvement looks like:

$$r' = f_r(r) = r_{min} + \frac{r_{max} - r_{min}}{r_{white} - r_{black}} \cdot (r - r_{black}) \quad (5.5)$$

where r_{min} is the minimum value of the channel red possible in the color corrected space, e.g. 0 for 24-bit images; and r_{black} is the red value of the measured darkest color in the image. Equation 5.2 is still valid for this linear mapping, but f becomes a composed application ($f : \mathbb{R}^3 \rightarrow \mathbb{R}^4 \rightarrow \mathbb{R}^3$), where \mathbf{M} becomes a 3×4 matrix, and we need to expand the definition of the \mathbf{P} colors using a homogeneous coordinate:

$$\begin{pmatrix} a_r & 0 & 0 & t_r \\ 0 & a_g & 0 & t_g \\ 0 & 0 & a_b & t_b \end{pmatrix} \cdot \begin{pmatrix} r_1 & r_2 \\ g_1 & g_2 \\ b_1 & b_2 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} r'_1 & r'_2 \\ g'_1 & g'_2 \\ b'_1 & b'_2 \end{pmatrix} \quad (5.6)$$

where a_k are the affine contributions and t_k are the translation contributions for each k channel, and now two points are required to obtain the color correction weights.

5.1.1.2 Affine correction

White balance is only a particular solution of an affine correction. We can generalize Equation 5.3 for, e.g., the red channel to accept contributions from green and blue channels:

$$r' = f_r(r, g, b) = a_{r,r} \cdot r + a_{r,g} \cdot g + a_{r,b} \cdot b = \sum_k^{r,g,b} a_{r,k} k \quad (5.7)$$

this expression is connected with the full matrix implementation, with 9 unknown weights:

$$\begin{pmatrix} a_{r,r} & a_{r,g} & a_{r,b} \\ a_{g,r} & a_{g,g} & a_{g,b} \\ a_{b,r} & a_{b,g} & a_{b,b} \end{pmatrix} \cdot \begin{pmatrix} r_1 & r_2 & r_3 \\ g_1 & g_2 & g_3 \\ b_1 & b_2 & b_3 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} r'_1 & r'_2 & r'_3 \\ g'_1 & g'_2 & g'_3 \\ b'_1 & b'_2 & b'_3 \end{pmatrix} \quad (5.8)$$

where $a_{j,k}$ are the weights of the \mathbf{M} matrix, and we need 3 known colors references to solve the system (see Figure 5.5).

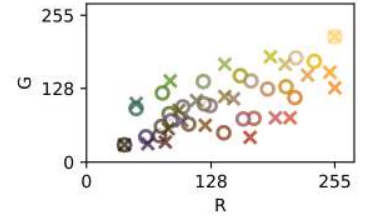


Figure 5.4: RGB colors of the ColorChecker of an image are projected in the *red-green* plane. The colors are replicated: (o) show the original colors of the ColorChecker and (x) show the corrected values of the augmented version shown in Figure 5.2 using a *white-balance with black-subtraction correction*. The whitest point (upper right) and the blackest point (lower left) are the only ones properly corrected.

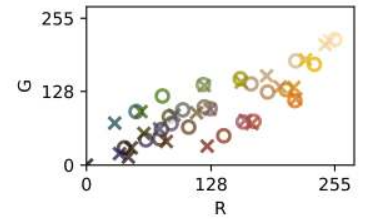


Figure 5.5: RGB colors of the ColorChecker of an image are projected in the *red-green* plane. The colors are replicated: (o) show the original colors of the ColorChecker and (x) show the corrected values of the augmented version shown in Figure 5.2 using an *affine correction*. We could choose to fix 3 points, but here we applied an approximated solver to the system, so any of the points is strictly matched.

In turn, white-balance with black-subtraction Equation 5.5 is a specific solution of an affine transformation, which handles translation and can be generalized as:

$$r' = f_r(r, g, b) = t_r + \sum_k^{r,g,b} a_{r,k} k \quad (5.9)$$

also tied up to its matrix representation:

$$\begin{pmatrix} a_{r,r} & a_{r,g} & a_{r,b} & t_r \\ a_{g,r} & a_{g,g} & a_{g,b} & t_g \\ a_{b,r} & a_{b,g} & a_{b,b} & t_b \end{pmatrix} \cdot \begin{pmatrix} r_1 & r_2 & r_3 & r_4 \\ g_1 & g_2 & g_3 & g_4 \\ b_1 & b_2 & b_3 & b_4 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} r'_1 & r'_2 & r'_3 & r'_4 \\ g'_1 & g'_2 & g'_3 & g'_4 \\ b'_1 & b'_2 & b'_3 & b'_4 \end{pmatrix} \quad (5.10)$$

where $a_{j,k}$ and t_k are the weights of the \mathbf{M} matrix, and we require 4 known colors to solve the system (see Figure 5.6).

5.1.2 Polynomial corrections

As we have seen with affine corrections, we can expand the definition of the measured color space matrix \mathbf{P} including additional terms to it. This is useful to compute non-linear corrections using a linear matrix implementation. Formally, this space expansion can be seen as f being now a composed application:

$$f : \mathbb{R}^3 \rightarrow \mathbb{R}^{3+N} \rightarrow \mathbb{R}^3 \quad (5.11)$$

where \mathbb{R}^{3+N} is an extended color space derived from the original color space \mathbb{R}^3 . We can write a generalization of Equation 5.9 for polynomial corrections as follows:

$$r' = f_r(r, g, b) = t_r + \sum_k^{r,g,b} a_{r,k} k + \sum_i^N w_{r,i} \Phi_i(r, g, b) \quad (5.12)$$

where $\Phi(r, g, b) = \{\Phi_i(r, g, b)\}$, $i = 1, \dots, N$ is a set of monomials, w_i are the weight contributions for each monomial and N is the length of the monomial set [28].

The monomials in the set $\Phi(r, g, b)$ will have a degree 2 or more, because we do not unify the affine parts as monomials, we do so to emphasize their contribution to the correction. Also, notice that N is arbitrary, and we can choose how we construct our polynomial expansions by tuning the monomial generator $\Phi_i(r, g, b)$.

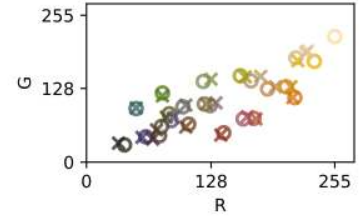


Figure 5.6: RGB colors of the ColorChecker of an image are projected in the *red-green* plane. The colors are replicated: (o) show the original colors of the ColorChecker and (x) show the corrected values of the augmented version shown in Figure 5.2 using an *affine correction with translation*. We could choose to fix 4 points, but here we applied an approximated solver to the system, so any of the points is strictly matched.

Despite that, N always relates to the number of vectors needed in Equation 5.2 to solve the system. \mathbf{M} takes the form of a $3 \times (4 + N)$ matrix, \mathbf{P} takes the form of a $(4 + N) \times (4 + N)$ matrix and \mathbf{Q} takes the form of a $3 \times (4 + N)$:

$$\begin{pmatrix} w_{r,N} & \cdots & w_{r,1} & a_{r,r} & a_{r,g} & a_{r,b} & t_r \\ w_{g,N} & \cdots & w_{g,1} & a_{g,r} & a_{g,g} & a_{g,b} & t_g \\ w_{b,N} & \cdots & w_{b,1} & a_{b,r} & a_{b,g} & a_{b,b} & t_b \end{pmatrix} \cdot \begin{pmatrix} \Phi_{N,0} & \Phi_{N,2} & \cdots & \Phi_{N,N+4} \\ \vdots & \vdots & \vdots & \vdots \\ \Phi_{1,1} & \Phi_{1,2} & \cdots & \Phi_{1,N+4} \\ r_1 & r_2 & \cdots & r_{N+4} \\ g_1 & g_2 & \cdots & g_{N+4} \\ b_1 & b_2 & \cdots & b_{N+4} \\ 1 & 1 & \cdots & 1 \end{pmatrix} = \begin{pmatrix} r'_1 & r'_2 & \cdots & r'_{N+4} \\ g'_1 & g'_1 & \cdots & g'_{N+4} \\ b'_1 & b'_1 & \cdots & b'_{N+4} \end{pmatrix} \quad (5.13)$$

5.1.2.1 Geometric polynomial correction

The simplest polynomial expansion of a color space occurs when $\Phi(r, g, b)$ generates a pure geometric set:

$$\Phi(r, g, b) = \{k^\alpha : 2 \leq \alpha \leq D\} \quad (5.14)$$

where $k \in \{r, g, b\}$ is any of the RGB channels, α is the degree of a given monomial of the set and D is the maximum degree we choose to form this set (see Figure 5.7). For example, for $D = 3$, it will produce the set:

$$\Phi_{D=3}(r, g, b) = \{r^2, g^2, b^2, r^3, g^3, b^3\} \quad (5.15)$$

Combining this expression with Equation 5.13, we can see we obtain a matrix that is directly related with the Vandermonde matrix [149], but for 3D data instead of 1D data.

5.1.2.2 Polynomial correction

Equation 5.14 can be generalized to take into account also cross-terms from any of the channels to create the monomial terms [27; 28]. So, we can write now:

$$\Phi(r, g, b) = \left\{ \prod_k^{rgb} k^{\alpha_k} : 2 \leq |\alpha| \leq D \right\} \quad (5.16)$$

where $|\alpha| = \sum_k^{rgb} \alpha_k$ is a metric, which is the sum of the degrees of each channel in the monomial, thus the degree of each monomial.

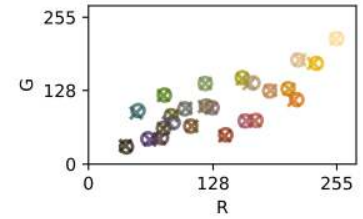


Figure 5.7: RGB colors of the ColorChecker of an image are projected in the *red-green* plane. The colors are replicated: (o) show the original colors of the ColorChecker and (x) show the corrected values of the augmented version shown in Figure 5.2 using a *geometric polynomial correction of degree 4*. Many of the points are almost matched due to the polynomial expansion.

Following with the example where $D = 3$, now we obtain an expanded set:

$$\Phi_{D=3}(r, g, b) = \left\{ r^2, g^2, b^2, rg, gb, br, r^3, g^3, b^3, rg^2, gb^2, br^2, gr^2, bg^2, rb^2, rgb \right\} \quad (5.17)$$

5.1.2.3 Root-polynomial correction

Finally, a root-polynomial correction is defined modifying Equation 5.16 to introduce the $|\alpha|$ -th root to each monomial [28]:

$$\Phi(r, g, b) = \left\{ \prod_k^{rgb} k^{|\alpha|} : 2 \leq |\alpha| \leq D \right\} \quad (5.18)$$

So, this reduces the amount of terms of each set for a given degree D . Then, our example with $D = 3$ becomes reduced to:

$$\Phi_{D=3}(r, g, b) = \left\{ \sqrt{rg}, \sqrt{gb}, \sqrt{br}, \sqrt[3]{rg^2}, \sqrt[3]{gb^2}, \sqrt[3]{br^2}, \sqrt[3]{gr^2}, \sqrt[3]{bg^2}, \sqrt[3]{rb^2}, \sqrt[3]{rgb} \right\} \quad (5.19)$$

Notice that all the terms present in a Vandermonde expansion have now disappeared, as they are now the same terms of the affine transformation due to the root application $\{\sqrt[3]{r^3}, \sqrt[3]{g^3}, \sqrt[3]{b^3}\} = \{\sqrt{r^2}, \sqrt{g^2}, \sqrt{b^2}\} = \{r, g, b\}$, and the only remaining terms are the roots of the cross-products.

5.1.3 Thin-plate spline correction

As an alternative to the former approaches, we can use thin-plate spline as the basis of the expansion to the color space in \mathbf{P} [15]:

$$r' = f_r(r, g, b) = t_r + \sum_k^{r,g,b} a_{r,k} k + \sum_i^N w_{r,i} h_i(r, g, b) \quad (5.20)$$

where w_i are the weight contributions for each spline contributions, and $h_i(r, g, b)$ are kernels of h in the N known colors. We will follow the same formulation described in chapter 3. A more detailed definition of h_i functions can be found there. Also, notice this expression is really similar to Equation 5.12 of polynomial corrections, the main difference is the fact that the number of N spline contributions equals to the number of color references (see Figure 5.8).

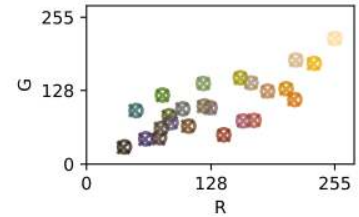


Figure 5.8: RGB colors of the ColorChecker of an image are projected in the *red-green* plane. The colors are replicated: (o) show the original colors of the ColorChecker and (x) show the corrected values of the augmented version shown in Figure 5.2 using a *thin-plate spline correction*. All the points are strictly matched by the TPS definition.

Equation 5.2 becomes now:

$$\begin{pmatrix} w_{r,1} & \cdots & w_{r,N} & a_{r,r} & a_{r,g} & a_{r,b} & t_r \\ w_{g,1} & \cdots & w_{g,N} & a_{g,r} & a_{g,g} & a_{g,b} & t_g \\ w_{b,1} & \cdots & w_{b,N} & a_{b,r} & a_{b,g} & a_{b,b} & t_b \end{pmatrix} \cdot \begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,N} \\ \vdots & \vdots & \vdots & \vdots \\ h_{N,1} & h_{N,2} & \cdots & h_{N,N} \\ r_1 & r_2 & \cdots & r_N \\ g_1 & g_2 & \cdots & g_N \\ b_1 & b_2 & \cdots & b_N \\ 1 & 1 & \cdots & 1 \end{pmatrix} = \begin{pmatrix} r'_1 & r'_2 & \cdots & r'_N \\ g'_1 & g'_1 & \cdots & g'_N \\ b'_1 & b'_1 & \cdots & b'_N \end{pmatrix} \quad (5.21)$$

This system is unbalanced, as we have N colors vectors in \mathbf{P} and \mathbf{Q} . In other corrections, we used four additional color references to solve the system, but here each new color is used to compute an additional spline, unbalancing the system again. Alternatively, the TPS formulation imposes two additional conditions [115]: the sum of $w_{j,k}$ coefficients is to be 0, and their cross-product with the \mathbf{P} colors as well. As a consequence of such conditions, spline contributions tend to 0 at infinity, while affine contributions prevail. This makes our system of equations solvable, and it can be expressed as an additional matrix product:

$$\begin{pmatrix} w_{r,1} & \cdots & w_{r,N} \\ w_{g,1} & \cdots & w_{g,N} \\ w_{b,1} & \cdots & w_{b,N} \end{pmatrix} \cdot \begin{pmatrix} r_1 & \cdots & r_N \\ g_1 & \cdots & g_N \\ b_1 & \cdots & b_N \\ 1 & \cdots & 1 \end{pmatrix}^T = 0 \quad (5.22)$$

5.1.3.1 Polynomial radial basis functions

The RBF used to compute splines remains open to multiple definitions. The thin-plate approach to compute those splines implies using solutions of the biharmonic equation [115]:

$$\Delta^2 U = 0 \quad (5.23)$$

that minimize the bending energy functional described by many authors, thus resembling the spline solution to the trajectory followed by an n -dimensional elastic plate. These solutions are the polynomial radial basis functions and a general solution is provided for n -dimensional data as [145; 144; 139]:

$$h_c(\mathbf{s}) = U(\mathbf{s}, \mathbf{c}) = \begin{cases} \|\mathbf{s} - \mathbf{c}\|^{2k-n} \ln \|\mathbf{s} - \mathbf{c}\| & 2k - n \text{ is even} \\ \|\mathbf{s} - \mathbf{c}\|^{2k-n} & \text{otherwise} \end{cases} \quad (5.24)$$

where n is the number of dimensions, k is the order of the functional, \mathbf{s} and \mathbf{c} are the data points where the spline is computed and $\|\cdot\|$ is a metric.

For a bending energy functional (the metal thin-plate approach) $k = 2$ and $n = 2$ (2D data), we obtain the usual thin-plate spline RBF [115]:

$$h_c(\mathbf{s}) = \|\mathbf{s} - \mathbf{c}\|^2 \ln \|\mathbf{s} - \mathbf{c}\| \quad (5.25)$$

But for $k = 2$ and $n = 3$ (3D data) we obtain [115]:

$$h_c(\mathbf{s}) = \|\mathbf{s} - \mathbf{c}\| \quad (5.26)$$

It is unclear why in the TPS3D to color correct images, Menesatti et al. [15] used the definition for 2D data (Equation 5.25), rather than the actual 3D definition (Equation 5.26) which according to the literature should yield to more accurate results. We will investigate here the impact of this change in the formal definition of the TPS3D.

So far, we have not defined a metric $\|\cdot\|$ to solve the TPS contributions. We will follow Menesatti et al. and use the euclidean metric of the RGB space. We will also name this metric Δ_{RGB} , as it is commonly known in colorimetry literature [15]:

$$\|\mathbf{s} - \mathbf{c}\| = \Delta_{RGB}(\mathbf{s}, \mathbf{c}) = \sqrt{(r_s - r_c)^2 + (g_s - g_c)^2 + (b_s - b_c)^2} \quad (5.27)$$

5.1.3.2 Smoothing the thin-plate spline correction

Approximating the TPS corrections is a well-known technique [139; 145]. Specifically, this is performed in ill-conditioned scenarios where data is noisy or saturated, and strict interpolation between data points, leads to important error artifacts. We propose now adding a smoothing factor to the TPS3D, to improve color correction in ill-conditioned situations.

We approximated the TPS by adding a smoothing factor to the spline contributions, which reduces the spline contributions in favor of the affine ones (see Figure 5.9). Taking Equation 5.20, we will introduce a smooth factor only for those color references where the center of the spline was those references themselves:

$$r'_j = f_r(r_j, g_j, b_j) = t_r + \sum_k^{r_j, g_j, b_j} a_{r,k} k + \sum_i^N (w_{r,i} h_i(r_j, g_j, b_j) + \lambda \delta_{ij}) \quad (5.28)$$

where λ is the smoothing factor, and δ_{ij} is a Kronecker delta.

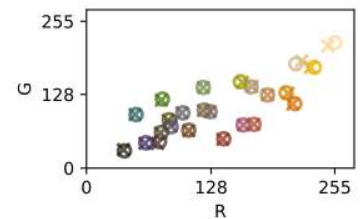


Figure 5.9: RGB colors of the ColorChecker of an image are projected in the *red-green* plane. The colors are replicated: (o) show the original colors of the ColorChecker and (x) show the corrected values of the augmented version shown in Figure 5.2 using a *smoothed thin-plate spline correction*. Not all the points are strictly matched now, as we relaxed the the TPS definition.

Notice that in the previous TPS definition the spline contributions of a reference color to the same reference color were 0 under the euclidean metric we chose. Also, notice the matrix product of Equation 5.21 is still valid, as we have only affected the diagonal of the upper part of the \mathbf{P} matrix. Thus,

$$\mathbf{P}_{smooth} = \mathbf{P} + \begin{bmatrix} \lambda \mathbf{I} \\ \mathbf{O}(4, N) \end{bmatrix} = \begin{pmatrix} h_{1,1} + \lambda & h_{1,2} & \dots & h_{1,N} \\ h_{2,1} & h_{2,2} + \lambda & \dots & h_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ h_{N,1} & h_{N,2} & \dots & h_{N,N} + \lambda \\ r_1 & r_2 & \dots & r_N \\ g_1 & g_2 & \dots & g_N \\ b_1 & b_2 & \dots & b_N \\ 1 & 1 & \dots & 1 \end{pmatrix} \quad (5.29)$$

where \mathbf{P} is the matrix of color references and their TPS expansion, \mathbf{I} is the identity matrix and $\mathbf{O}(4, N)$ is a matrix with os of size $4 \times N$.

5.2 Experimental details

So far, we have reviewed the state-of-the-art methods to color correct images to achieve consistent datasets using color references as fixed points in color spaces to compute color corrections. Also, we have proposed two updates to the TPS_{3D} method: using the suited RBF and smoothing the TPS contributions.

In Table 1, we show a summary of all the corrections that we studied in this work using the dataset described in the next section. First, a perfect correction (PERF) and a non-correction (NONE) scenario are present as **reference**. Notice that perfect correction will display here the quantization error after passing from 12-bit images to 8-bit images. Then several corrections have been implemented, that have been grouped by authorship of the methods and type of correction:

- **Affine (AFF):** white-balance (AFF₀), white-balance with black subtraction (AFF₁), affine (AFF₂), affine with translation (AFF₃).
- **Vandermonde (VAN):** four polynomial corrections from degree 2 to 5 (VAN₀, VAN₁, VAN₂ and VAN₃).
- **Cheung (CHE):** from Cheung et al. [27], four polynomial corrections with different terms: 5 (CHE₀), 7 (CHE₁), 8 (CHE₂) and 10 (CHE₃).
- **Finlayson (FIN):** from Finlayson et al. [28], two polynomial and two root-polynomial, of degrees 2 and 3 (FIN₀, FIN₁, FIN₂, FIN₃).
- **Thin-plate splines (TPS):** TPS_{3D} from Menesatti et al. [15] (TPS₀), our method using the proper RBF (TPS₁) and the same method with two smoothing values (TPS₂ and TPS₃).

Correction	Acronym	P extended color space
Perfect	PERF	(r, g, b)
No correction	NONE	(r, g, b)
White-balance	AFF0	(r, g, b)
White-balance w/ black subtraction	AFF1	$(1, r, g, b)$
Affine	AFF2	(r, g, b)
Affine w/ translation	AFF3	$(1, r, g, b)$
Vandermonde (degree=2)	VAN0	$(1, r, g, b, r^2, g^2, b^2)$
Vandermonde (degree=3)	VAN1	$(1, r, g, b, r^2, g^2, b^2, r^3, g^3, b^3)$
Vandermonde (degree=3)	VAN2	$(1, r, g, b, r^2, g^2, b^2, r^3, g^3, b^3, r^4, g^4, b^4)$
Vandermonde (degree=4)	VAN3	$(1, r, g, b, r^2, g^2, b^2, r^3, g^3, b^3, r^4, g^4, b^4, r^5, g^5, b^5)$
Cheung (terms=5)	CHE0	$(1, r, g, b, rgb)$
Cheung (terms=7)	CHE1	$(1, r, g, b, rg, rb, gb)$
Cheung (terms=8)	CHE2	$(1, r, g, b, rg, rb, gb, rgb)$
Cheung (terms=10)	CHE3	$(1, r, g, b, rg, rb, gb, r^2, g^2, b^2)$
Finlayson (degree=2)	FIN0	$(r, g, b, r^2, g^2, b^2, rg, rb, gb)$
Finlayson (degree=3)	FIN1	$(r, g, b, r^2, g^2, b^2, rg, rb, gb, r^3, g^3, b^3, rg^2, gb^2, rb^2, gr^2, bg^2, br^2, rgb)$
Finlayson root (degree=2)	FIN2	$(r, g, b, \sqrt{rg}, \sqrt{rb}, \sqrt{gb})$
Finlayson root (degree=3)	FIN3	$(r, g, b, \sqrt{rg}, \sqrt{rb}, \sqrt{gb}, \sqrt[3]{rg^2}, \sqrt[3]{gb^2}, \sqrt[3]{rb^2}, \sqrt[3]{gr^2}, \sqrt[3]{bg^2}, \sqrt[3]{br^2}, \sqrt[3]{rgb})$
Thin-plate splines (Manesatti)	TPS0	$(1, r, g, b, \Delta_1^2 \ln \Delta_1, \dots, \Delta_{24}^2 \ln \Delta_{24})$
Thin-plate splines (ours, smooth=0)	TPS1	$(1, r, g, b, \Delta_1, \dots, \Delta_{24})$
Thin-plate splines (ours, smooth=0.001)	TPS2	$(1, r, g, b, \Delta_1, \dots, \Delta_{24})$
Thin-plate splines (ours, smooth=0.1)	TPS3	$(1, r, g, b, \Delta_1, \dots, \Delta_{24})$

Table 5.1: All the color corrections performed in this work. The table shows the name of the correction, the tag used in this work to refer to the correction and the augmented definition for each vector of P , the color references or color to be corrected. In this table we use a reduced notation $\Delta_i = \Delta_{RGB}(\mathbf{s}_i, \mathbf{c})$ for simplicity.

5.2.1 Dataset and pipeline

As explained before, the usual approach to solve the image consistency problem is placing color references in a certain scene to later perform a color correction. There exists a widely spread usage of color charts, e.g. Macbeth ColorChecker of 24 colors [13]. Over the years, extensions of this ColorChecker have appeared, mostly presented by X-Rite, a Pantone company, or by Pantone itself, which introduced the Pantone Color Match Card[®] that features four Aruco patterns [19] to ease the pattern extraction when acquiring the colors of the chart.

Since in this chapter we do not propose improved versions of the charts themselves, we use an existing image dataset that contains images of the Macbeth ColorChecker of 24 colors in different scenes in order to evaluate our color correction with respect to image consistency; and benchmark it against other correction methods. The Gehler’s dataset is a widely used dataset with several versions, and there exists a deep discussion about how to use it. Despite the efforts of the dataset creators and other authors to enforce the use of the last “developed” dataset [147], here we use the RAW original version of the dataset [146], and we developed the images ourselves. We did so because we performed image augmentation over the dataset, as we want to control the developing process of the RAW images and also measuring the resulting augmented colors directly from the provided annotations in the original dataset (see Figure 5.10).

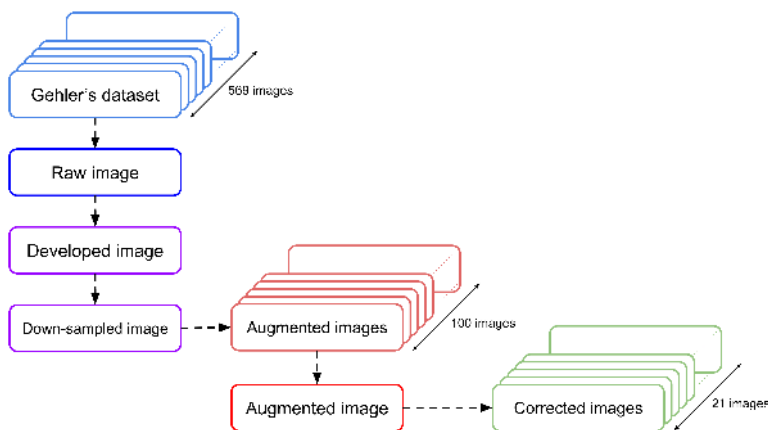


Figure 5.10: Our pipeline: for each Gehler’s dataset raw image (bayer) we develop an RGB image, which is already the half size of the original image, also this image is down-sampled to reduce its size 4 times. Then we augment this down-sampled image with 100 sample augmentation scenarios. For each augmented scenario we correct back before augmentation using 21 different correction methods described in Table 5.1.

The Gehler’s dataset comprises images from two cameras: a Canon EOS 1DS (86 images) and a Canon EOS 5D (483 images), both cameras producing raw images of 12-bit per channel ($\mathbb{I}N_{[0,4096]}^3$) with a RGG B Bayer pattern [150]. This means we have twice as many green pixels than red or blue pixels.

Images have been processed using Python [151], represented by numpy arrays [152; 86], and have been developed using imageio [92] and rawpy, the Python wrapper of raw binary, the utility used elsewhere to process the Gehler’s dataset [146; 147]. When developing the images, we implemented no interpolation, thus rendering images half the size of the raw image (see Table 5.2). These are our *ground-truth* images: the colors in these images are what we are trying to recover when performing the color corrections.

We chose to work with 8-bits per channel RGB images as is the most commonly developed pixel format present nowadays. First, we cast the developed dataset 12-bit images ($\mathbb{N}_{[0,4096]}^3$) to 8-bit resolution ($\mathbb{N}_{[0,255]}^3$). The difference between the cast images and the groundtruth images is the quantization error, due to the loss of color depth resolution. To speeded up the calculations without losing statistical significance in the results we down-sampled the images by a factor 4. The down-sampling factor is arbitrary and depends on the level of redundancy of the color distribution in our samples. We selected a down-sampling factor that did not alter the color histogram of the images of the dataset, see Figure 5.11. Table 5.2 shows the final image sizes for each camera on the Gehler’s dataset.

Camera	Raw image	Developed image	Down-sampled image
Canon EOS 1DS	(4064, 2704)	(2041, 1359)	(511, 340)
Canon EOS 5D	(4368, 2912)	(2193, 1460)	(549, 365)

Subsequently, we augmented the dataset using imgaug [93] (see Figure 5.12) that generated image replicas simulating different acquisition setup conditions. The augmentations were performed with random augmentations that modeled: linear contrast, gamma contrast and channel cross-talk. Geometrical distortions were omitted because this work is focused on a colorimetry problem.

Finally, we corrected each developed, down-sampled and augmented image using the color corrections listed in Table 5.1. These corrections were computed using color-normalized versions of those images ($\mathbb{R}_{[0,1]}^3$). White-balance corrections were implemented directly with simple array operations [86]; while affine, polynomial and root-polynomial corrections were applied as implemented in the [153]. We implemented our own version of the TPS with the corresponding RBFs, including support for smoothing, using a derivation of the scipy [86].

Table 5.2: Sizes in pixels (x, y) of the images along our pipeline. Notice raw pixels are natural pixels of the sensor, this means each pixel only represents one color (red, green or blue).

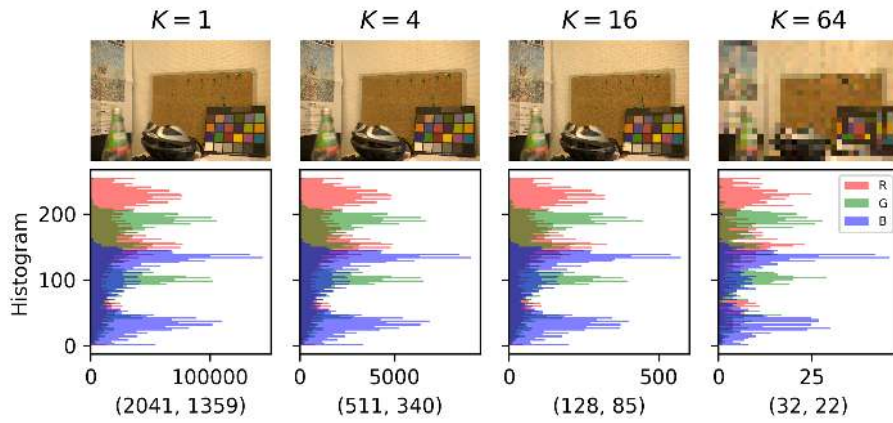


Figure 5.11: An image from Gehler’s dataset ($K=1$) is down-sampled with 3 factors ($K=4, 16, 64$), where K is the down-sampling factor. The figure also shows the histogram associated with each image and the size in pixels of the image. Down-sampled images by a factor 4 maintain the histogram representation, but further down-sampling alters the histogram.



Figure 5.12: Different examples of color augmentation using `imgaug` in Python. The upper-left image is the developed original image from the Gehlre’s dataset. The other images are augmentations of these image with variations in color, contrast and saturation.

5.2.2 Benchmark metrics

In order to benchmark the performance of all the correction methods, we implemented different metrics. First, a *within-distance* ($\overline{\Delta_{RGB,within}}$) as the mean distance of all and only the colors in the ColorChecker to their expected corrected values [15]:

$$\overline{\Delta_{RGB,within}} = \frac{\sum_{l=1}^L \Delta_{RGB}(s'_l, c'_l)}{L} \quad (5.30)$$

where s'_l is the corrected version of a certain ColorChecker captured color s_l , which has a ground-truth reference value of c'_l , and L is the number of reference colors in the ColorChecker (in our case $L = 24$). Alongside with this metric, a criterion was defined to detect *failed corrections*. We consider failed corrections those which failed to *reduce the within-distance* between the colors of the ColorChecker after the correction. Then, by comparing the $\overline{\Delta_{RGB,within}}$ of the corrected image and the image without correction (NONE):

$$\overline{\Delta_{RGB,within}} - \overline{\Delta_{RGB,within,NONE}} > 0. \quad (5.31)$$

Second, we defined a *pairwise-distance set* ($\Delta_{RGB,pairwise}$) as the set of the distances between all the colors in a ColorChecker in the same image:

$$\Delta_{RGB,pairwise} = \{\Delta_{RGB}(c'_l, c'_m) : l, m = 1, \dots, L\} \quad (5.32)$$

where c'_l and c'_m are colors of the ColorChecker in a given image. Also, we implemented another criterion to detect *ill-conditioned corrections*. Ill-conditioned corrections are those failed corrections in which colors have also collapsed into extreme RGB values (see Figure 5.16). By using the *minimum pairwise-distance* for a given color corrected image:

$$\min(\Delta_{RGB,pairwise}) < \delta, \quad (5.33)$$

where δ is a constant threshold which tends to zero. Note that somehow we were measuring here the opposite to the first criterion: we expected erroneous corrected colors to be pushed away from the original colors Equation 5.31. However, sometimes they also got shrunk into the borders of the RGB cube Equation 5.33, causing two or more colors to saturate into the same color. Also, notice that we did not define a *mean pairwise-distance*, $\overline{\Delta_{RGB,pairwise}}$, as it was useless to define a criterion around a variable which presented huge dispersion in ill-conditioned scenarios (e.g. colors pairs were at the same time close and far, grouped by clusters).

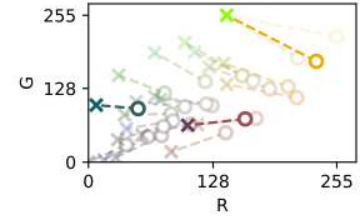


Figure 5.13: The metric $\overline{\Delta_{RGB,within}}$ is represented. RGB colors of the ColorChecker of an image are projected in the *red-green* plane. The colors are present as their ground-truth value (\circ) and their augmented copy (\times). Dashed lines across the plane show the $\Delta_{RGB,within}$ between each color pair. Cyan, magenta and yellow pairs are highlighted above the other ColorChecker colors.

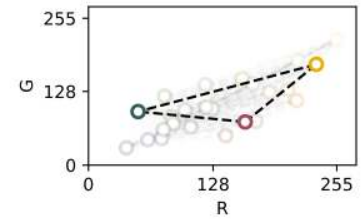


Figure 5.14: The set $\Delta_{RGB,pairwise}$ is represented. RGB colors of the ColorChecker of an image are projected in the *red-green* plane. The colors are present as their ground-truth value (\circ). Dashed lines across the plane show the $\Delta_{RGB,pairwise}$ between all the colors. The distances between cyan, magenta and yellow are highlighted above the other distances.

Third, we defined an *inter-distance* ($\overline{\Delta_{RGB,inter}}$) as the color distance between all the other colors in the corrected images with respect to their values in the ground-truth images (measured as the mean RGB distance of all the colors in the image but subtracting first the ColorChecker area as proposed by Hemrit et al. [147]):

$$\overline{\Delta_{RGB,inter}} = \frac{\sum_{m=1}^M \Delta_{RGB}(s'_m, c'_m)}{M} \quad (5.34)$$

where M is the total amount of pixels in the image other than those of the ColorChecker. This definition particularized the proposal of Menesatti et al., where in order to compute the $\overline{\Delta_{RGB,inter}}$ they used all the colors of another color chart instead of the actual image. Specifically, Menesatti et al. used the GretagMacbeth ColorChecker SG® with 140 color patches [15].

Finally, to compare the computational performance of the methods, we measured the *execution time* (\mathcal{T}) to compute each corrected image, \mathcal{T} was also measured for images with different sizes to study its linearity against the amount of pixels in an image in all corrections [148].

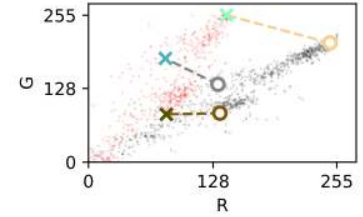


Figure 5.15: The metric $\overline{\Delta_{RGB,inter}}$ is represented. RGB colors of an entire image are projected in the *red-green* plane. The colors are present as their ground-truth value (black points, \circ) and their augmented copy (red points, \times). Then, three random colors are selected to show dashed lines across the plane to show the $\Delta_{RGB,inter}$ between each color pair.

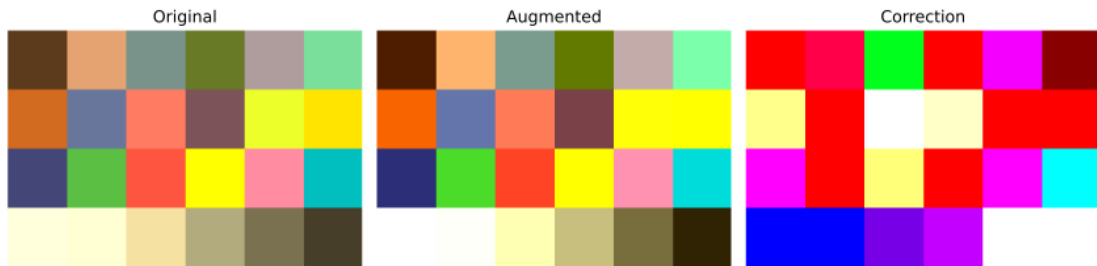
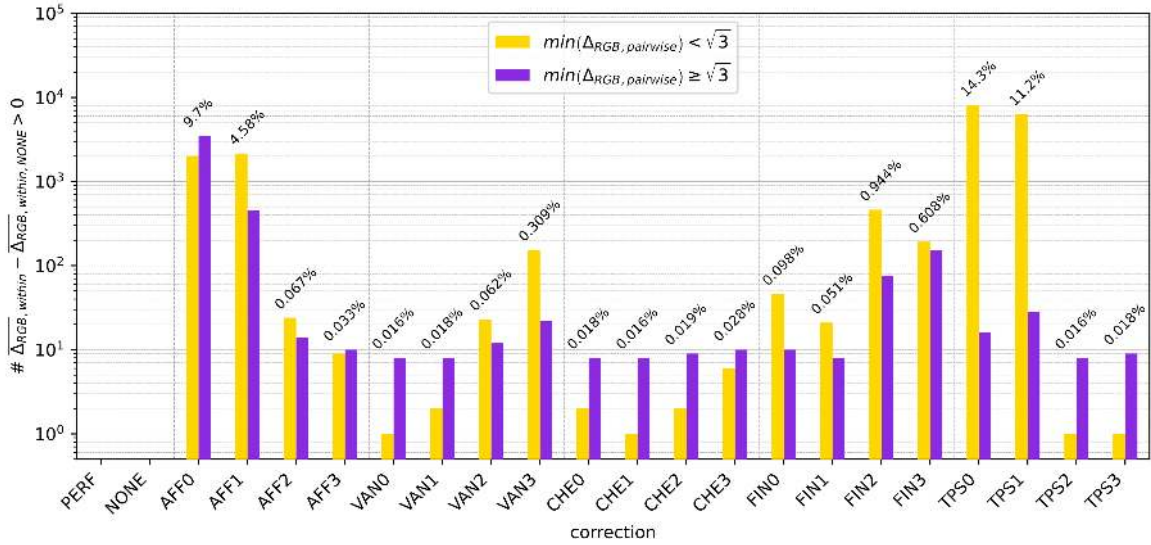


Figure 5.16: An example of a failed and ill-conditioned correction. The augmented image shows saturated colors: the yellowish colors and the whitish colors. The corrected image is computed with the TPSo method rendering an erroneous result.

5.3 Results

5.3.1 Detecting failed corrections

Let us start with the results of the detection of failed corrections for each color correction proposed. Here we used the defined criteria for $\overline{\Delta_{RGB,within}}$ (Equation 5.31) and $\Delta_{RGB,pairwise}$ (Equation 5.33) to discover failed and ill-conditioned corrections (see Figure 5.16).



First, we subtracted the $\overline{\Delta_{RGB,within}}$ measures to the other $\overline{\Delta_{RGB,within}}$ and compare this quantity to 0, following Equation 5.31. Those cases where this criterion were greater than 0 were counted as failed corrections.

Second, for those corrections marked as failed, the $\Delta_{RGB,pairwise}$ criteria (Equation 5.33) was applied to discovery ill-correction scenarios (such as Figure 5.16) in between failed corrections. The $\Delta_{RGB,pairwise}$ criteria was implemented using a $\delta = \sqrt{3}$, due to the fact this is the $\Delta_{RGB,pairwise}$ of two colors that dist one digit from each other in each channel (i.e. (0, 0, 0) and (1, 1, 1) for colors in the $\mathbb{N}_{[0,255]}^3$ space).

Finally, we also computed the relative % of failed color corrections referenced to the total of color corrections performed, this figure is relevant as we removed these cases from further analysis.

Figure 5.17 showed how resilient the studied color correction methods are to fail, let us see how well each group of correction has scored here:

- **AFF:** AFF0 and AFF1 scored poor results, 9.7% and 4.58% of failed corrections, respectively. On the contrary, AFF2 and AFF3 scored almost any failures, this responds to the fact that AFF2 and AFF3 were using all the available references, instead of one or two.

Figure 5.17: A count of the failed corrections for each correction method is shown. Failed corrections are selected if their $\overline{\Delta_{RGB,within}}$ computation is greater than the NONE correction. After this, the count is divided in ill-conditioned results or not. Ill-condition is assessed using the $\Delta_{RGB,pairwise}$ comparison to a minimum distance of $\Delta_{RGB} = \sqrt{3}$.

Also, AFF_1 reduces to a half the failed correction from AFF_0 , as AFF_3 reduces the AFF_1 ones, as they are the same corrections but incorporating the translation component to the corrections (Table 5.1).

- **VAN:** all four corrections scored less than a 1% of failed corrected images. Notice here that the degree of the polynomial expansion (from 2 to 5, VAN_0 to VAN_3) correlates with the amount of failed corrections. Specially for those scenarios who present ill-conditioned results, where $\min(\Delta_{RGB, pairwise}) < \sqrt{3}$.
- **CHE:** all four corrections scored less than 1% of failed corrected images. Results were very similar to VAN corrections. The correlation between the degree of the polynomial expansion (Table 5.1) and the failed corrections was also seen here (AFF_1 to AFF_3).
- **FIN:** all four correction scored less than 1% of failed corrected images. Root-polynomial corrections (FIN_1 and FIN_3) showed around the half of failed correction than their respective polynomial corrections (FIN_0 and FIN_2). But, all of them scored worst results than VAN_0 , VAN_1 and all four CHE. This might be linked with the fact FIN corrections did not implement the translation component to the expansion (Table 5.1), as AFF_0 and AFF_2 .
- **TPS:** TPS_1 and TPS_0 scored the worst results in Figure 5.17, 11.2% and 14.3% of failed cases, respectively. With a huge presence of ill-conditioned results. On the contrary, TPS_2 and TPS_3 scored in the top positions alongside with VAN_0 , VAN_1 , CHE_0 and CHE_2 . It was easy to conclude that our proposition to smooth the thin-plate contributions to the color correction had succeeded in terms of fixing ill-conditioned scenarios (such as Figure 5.16).

5.3.2 Color correction performance

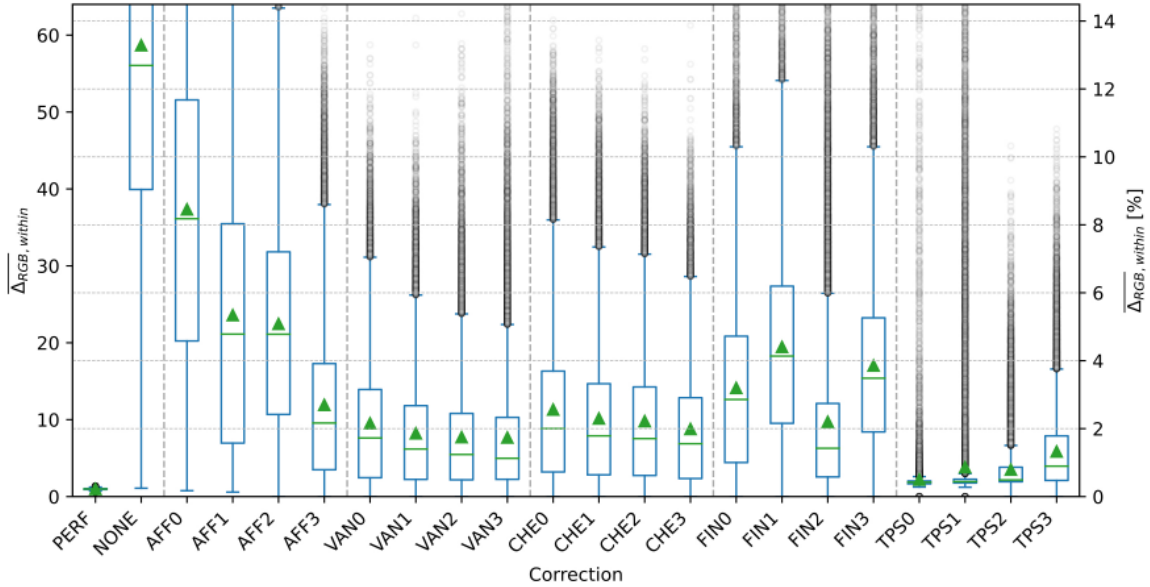
Once evaluated and cleaned the failed corrections from our results, we proceeded to evaluate how the proposed color corrections scored in terms of color correction performance. In other words, we evaluated how they minimize the median value of the *within-distances distributions* and the *inter-distances distributions*. Figure 5.18 and Figure 5.19.

We defined $\overline{\Delta_{RGB, within}}$ and $\overline{\Delta_{RGB, inter}}$ similar to Menesatti et al. [15], but it is also interesting to define these metrics with a percentage definition. The maximum distance in the RGB space is the distance $\Delta_{RGB}((0,0,0), (255,255,255)) = 255 \cdot \sqrt{3}$, following Equation 5.27. Thus,

$$\Delta_{RGB}[\%] = 100 \cdot \frac{\Delta_{RGB}}{255 \cdot \sqrt{3}} \quad (5.35)$$

Figure 5.18 and Figure 5.19 show both definitions.

5.3.2.1 Within-distances



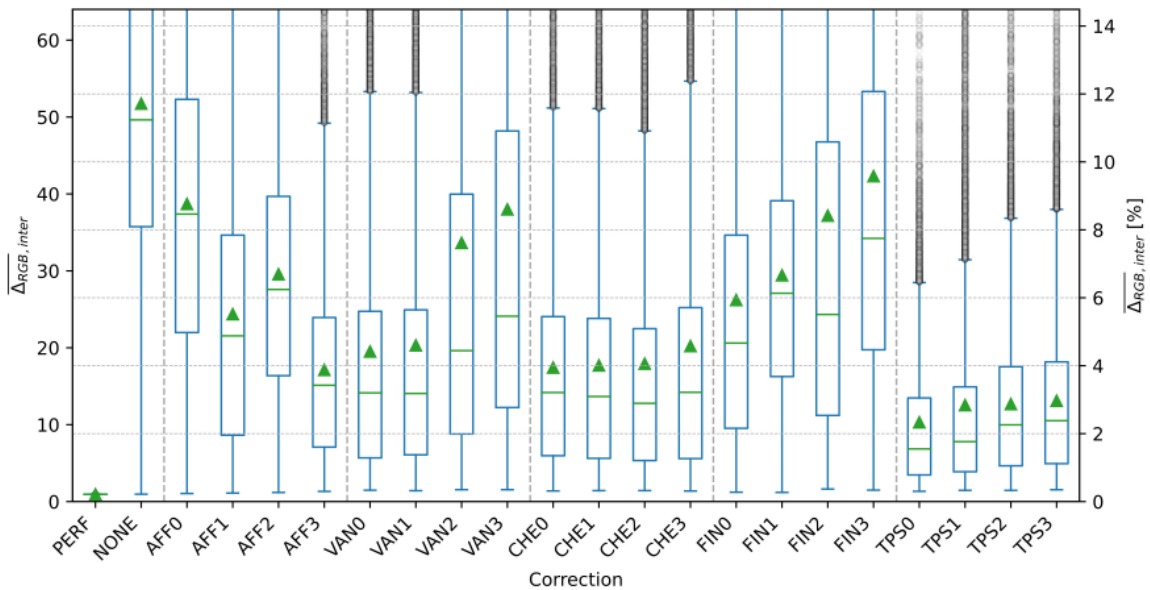
On one hand, let us see how well each group of correction has scored in the $\overline{\Delta_{RGB,within}}$ metric (see Figure 5.18):

- **AFF:** as expected AFF0, the white-balance correction, scored poorly. It was the worst correction, scoring a mean $\overline{\Delta_{RGB,within}}$ of more than 8%. This is due to the fact that only one color reference (white) was taken into account to compute this color correction. AFF1 and AFF2 scored a similar mean $\overline{\Delta_{RGB,within}}$ above 5%. AFF2, the most complete affine correction, scored the best result in this group with around a 3%. Also, the addition of a translation component, AFF1 and AFF3 (Table 5.1), reduced the $\overline{\Delta_{RGB,within}}$.
- **VAN:** all four VAN corrections scored a better mean and median $\overline{\Delta_{RGB,within}}$ than AFF3, all scoring around 2% or less. This was good news, here it can be seen that systematically increasing the degree of a polynomial expansion results in a better fitting of the RGB color space deformation. Despite this, results showed how this method seems to converge to a minimum median $\overline{\Delta_{RGB,within}}$ around 1%.
- **CHE:** all four CHE corrections scored a better mean and median $\overline{\Delta_{RGB,within}}$ than AFF3, but they scored slightly worst results than VAN corrections (3-1%). This result showed that adding cross-term contributions to the polynomial expansion (Table 5.1) did not improve the fitting of the RGB color space deformation.

Figure 5.18: The $\overline{\Delta_{RGB,within}}$ for each image in the dataset and for each augmentation is shown as a distribution against the color correction techniques. The means of the distributions are also present (\triangle). PERF correction is not zero and shows the quantization effect. NONE is a reference of not applying any correction at all. The rest of the corrections are grouped in: AFF, VAN, CHE, FIN and TPS corrections.

- **FIN:** surprisingly FIN corrections scored the worse results above all the polynomial corrections -VAN and CHE- (5-3%). This might be explained by the lack of translation components (Table 5.1). Also, FIN₁ and FIN₃, the root-polynomial, scored around 2% more $\overline{\Delta_{RGB,within}}$ than their respective polynomial corrections FIN₀ and FIN₂ (Table 5.1).
- **TPS:** all TPS correction scored the best results for this metric. TPS₀ and TPS₁ scored an incredible good result of less than 1% of mean $\overline{\Delta_{RGB,within}}$. Here it can be seen the outlying behavior of TPS₀ and TPS₁ was not fully solved before, as the mean $\overline{\Delta_{RGB,within}}$ of TPS₁ is outside the distribution box. Also, TPS₂ and TPS₃, which approximated the TPS method to the AFF₃ method scored also excellent results, better than VAN₃, which is the best polynomial correction. Moreover, we checked with these results that increasing the smooth factor in the TPS formulation (TPS₁ → TPS₂ → TPS₃), increased the $\overline{\Delta_{RGB,within}}$ as it smoothed the fitting RGB space color deformation.

5.3.2.2 Inter-distances



On the other hand, let us see how well each group of correction has scored in the $\overline{\Delta_{RGB,inter}}$ metric (see Figure 5.19):

- **AFF:** these corrections showed somehow expected results, as they scored similar $\overline{\Delta_{RGB,inter}}$ than $\overline{\Delta_{RGB,within}}$. $\overline{\Delta_{RGB,inter}}$ increased around 1-2% for all corrections, respectively to $\overline{\Delta_{RGB,within}}$. AFF₃ was the best of the AFF corrections, performing a mean and median inner-distance around 4%.

Figure 5.19: The $\overline{\Delta_{RGB,inter}}$ for each image in the dataset and for each augmentation is shown as a distribution against the color correction techniques. The means of the distributions are also present (Δ). PERF correction is not zero and shows the quantization effect. NONE is a reference of not applying any correction at all. The rest of the corrections are grouped in: AFF, VAN, CHE, FIN and TPS corrections.

- **VAN:** VAN₀ and VAN₁ presented similar results to AFF₃, mean $\overline{\Delta_{RGB,inter}}$ were around 4%, and the distribution matched AFF₃ distribution (mean, median, box and outliers). VAN₂ and VAN₃ presented worst results, their distributions got spread. VAN₂ and VAN₃ scored mean $\overline{\Delta_{RGB,inter}}$ around 8%. Despite the higher degree polynomial expansions systematically reduced the $\overline{\Delta_{RGB,within}}$, they increased the $\overline{\Delta_{RGB,inter}}$.
- **CHE:** all four corrections scored similar results to AFF₃, matching the AFF₃ distribution (mean, median, box and outliers). Thus, performing a mean and median inner-distance around 4%.
- **FIN:** all four corrections scored the worst results. FIN₀ and FIN₁ scored similar to AFF₁ and AFF₂. FIN₃ showed the worst correction of the overall data, above AFF₀ and VAN₃, with a median $\overline{\Delta_{RGB,within}}$ of almost 10%, and a mean of almost 8%. Once again, it was observed that root-polynomial presents worst results than their respective polynomial correction.
- **TPS:** all four correction scored the best results also for this metric. The effect of smoothing or not the TPS correction was reduced in this metric. All four corrections scored median and mean $\overline{\Delta_{RGB,inter}}$ around 2%.

All in all, TPS corrections proved to provide the best solution to color correct images in our dataset. The original Menesatti et al. [15] proposal (TPS₀) worked slightly better than our first proposal of using the recommended RBF for 3D spaces (TPS₁). The smoothed TPS proposals (TPS₂ and TPS₃) scored the subsequent best results for both metrics, $\overline{\Delta_{RGB,within}}$ and $\overline{\Delta_{RGB,inter}}$. VAN₃ proved to be a good competitor in the within-distance metric, on the contrary had one of the poor results in the $\overline{\Delta_{RGB,inter}}$ metric. AFF₃, VAN₀, VAN₁ and all CHE methods proved to be good competitors in the $\overline{\Delta_{RGB,inter}}$ metric, that is an interesting result as it opens the possibility to have fall-back methods if the TPS fails.

5.3.3 Execution time performance

Let us see how the proposed color correction methods scored in terms of execution time for each image corrected. As our dataset has images from two cameras, with different sizes, we decided to focus only in one camera to ensure results were not affected by the disparity in size. We chose to work with the larger subset of images: the Canon EOS 5D with 483 images. These images have 549×365 pixels = 200385 pixels \approx 0.2 Mpx (see Table 5.2), as we down-sampled them ($K = 4$) to speed up the global computation time of our pipeline (see Figure 5.10).

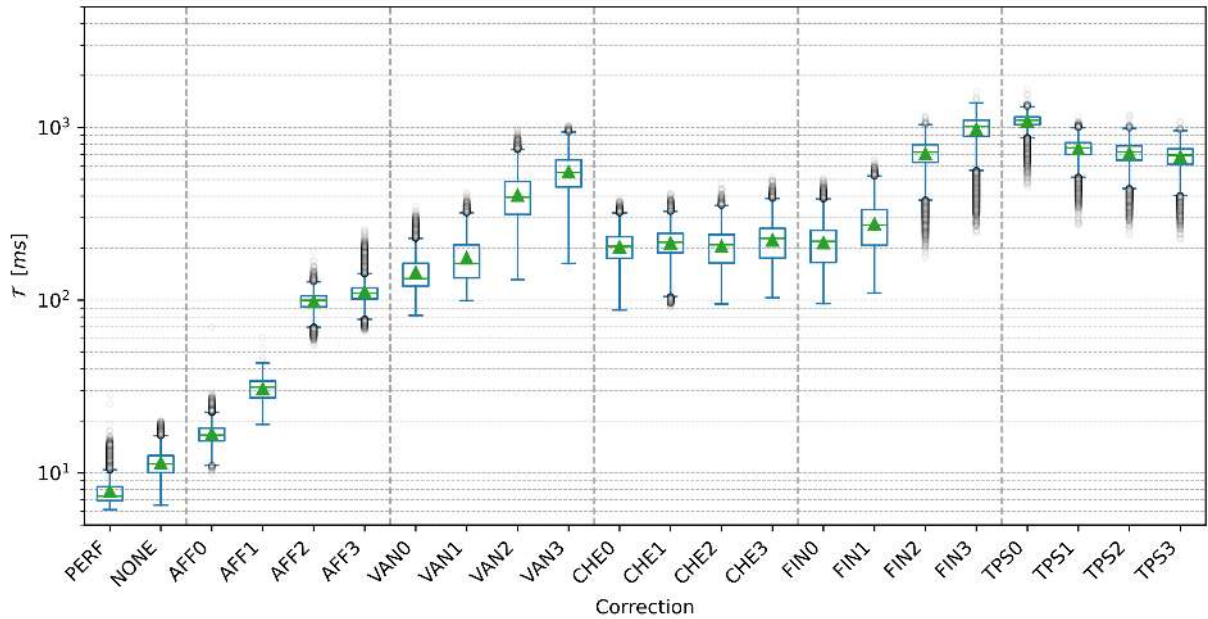


Figure 5.20 shows the results of the measured execution times. PERF execution time represents the minimal time to compute our pipeline, as the PERF method also went all the way computing the same pipeline, it just returns the perfect expected image in 8-bit representation. NONE did the same but returning the image without applying any correction. Let us see how the other methods scored in this benchmark:

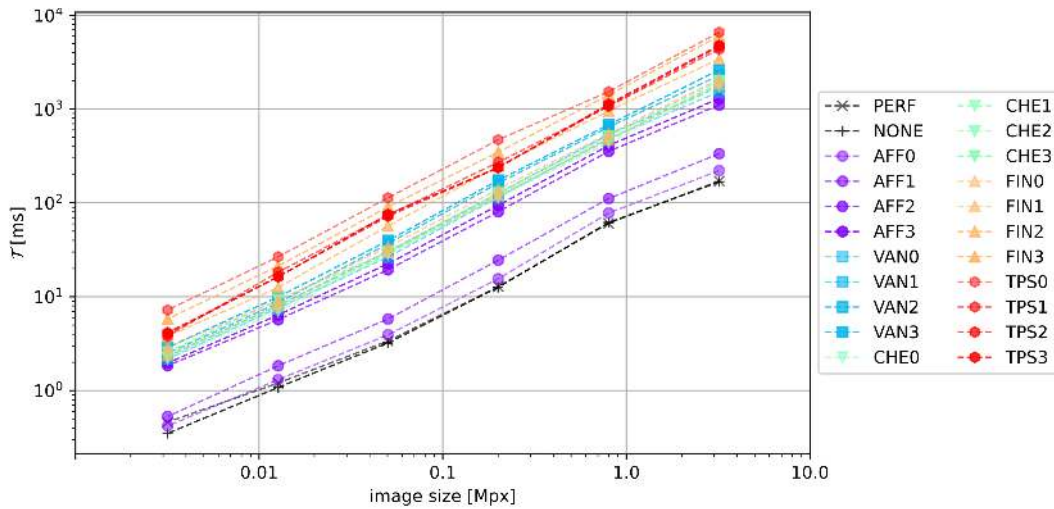
Figure 5.20: The execution time in milliseconds for each image in the dataset and for each augmentation is shown as a distribution against the color correction techniques. The means of the distributions are also present (Δ).

- **AFF:** all four corrections scored the best results in the benchmark, as expected, as they are the simpler corrections regarding implementation. AFF0 and AFF1 scored a mean \mathcal{T} per image around 20-30 ms. AFF2 and AFF3 scored around 100 ms.
- **VAN:** all four corrections were slower than AFF3, from a 100 ms to 600 ms. As AFF3 is a polynomial correction of order 1, and the subsequent corrections are VAN0 to VAN3, with degrees 2 to 5, respectively, we can observe an exponential relation between the polynomial degree of the expansion and \mathcal{T} .
- **CHE:** all four corrections scored similar results to VAN1, with a mean \mathcal{T} per image around 200 ms. Being around 10 times slower than AFF0 and 2 times slower than AFF2-AFF3.
- **FIN:** FIN0 scored very similar to the above-mentioned CHE methods. Despite this good result, the other FIN displayed slow mean \mathcal{T} per image. FIN3 takes around 1000 ms to compute per image. FIN methods presented again that adding superior degrees to the polynomial expansion add computational time to the correction. Also, adding complex operation to the pipeline, such as computing a root square, it also affects the computational cost of the solution.

- TPS:** TPS₀ scored very similar to FIN₃, and in fact scored above the mean 1000 ms timestamp, achieving the worst score for this metric. This makes sense, as TP₀ is the original TPS method, which implied the use of a more complex function (see Table 5.1). Our proposal TPS₁ reduced this computation time around 800 ms. Also, adding smooth to the TPS method seems to reduce slightly its mean \mathcal{T} , arriving down to 700 ms at TPS₃. TPS₃ is around 25 times slower than AFF₁ and 8 times slower than AFF₃.

All in all, results for AFF, VAN, CHE and FIN showed that increasing the degree of the polynomial expansion, increased the mean \mathcal{T} for each image. AFF corrections achieved the top scores as they are computationally simple. And, TPS scored poorly in this benchmark, as expected [148]. Also, the scores for VAN₂, VAN₃, FIN₂ and FIN₃ were also poor. We accomplished to improve slightly the TPS scores by introducing a change in the RBF and the smooth parameter.

Finally, as we computed the above-mentioned results using thumbnail images ($K = 4$, 549×365 pixels), we wanted to check the computational order of the presented methods against the size of the image.



To do so, we computed a reduced dataset of images containing 10 images from the dataset and recompute the same pipeline (see Figure 5.10) for different K down-sampling constants (see Figure 5.11): 1, 2, 4, 8, 16 and 32. Which render images of approximately: 3.2, 0.80, 0.2, 0.05, 0.012 and 0.003 megapixels, respectively.

Figure 5.21 shows the computed results, as we can see all corrections performed with a linear computational order $O(n)$, for the all the down-sampled versions of the images. The figure also shows the relation we found earlier between the different correction, i.e. TPS are almost two decades apart from AFF corrections. We consider these results to be useful as they could be used eventually as a design rule when designing color corrections pipelines.

Figure 5.21: The execution time in milliseconds against the image size for a reduced set of images of the dataset. Results show a linear behavior for all the corrections techniques. Corrections are grouped by color and marker, within the same group different transparencies have been applied to differentiate the corrections

5.4 Conclusions

In this chapter, we improved the work done by Menesatti et al. [15]. We successfully reproduced their findings about the TPS_{3D} method for color correction to achieve image consistency in datasets. It can be shown that our results match theirs not only qualitatively but also quantitatively. For this purpose, Table 5.3 shows a summary of the results above-presented for future comparison.

Also, we extended the study to other state-of-the-art methods, Gong et al. [44], Cheung et al. [27] and Finlayson et al. [28], that can be found in standard libraries [153]. The TPS_{3D} proved to be the best correction color method among the other in terms of color correction performance, both in $\overline{\Delta_{RGB,within}}$ and $\overline{\Delta_{RGB,inter}}$ metrics. Despite this, TPS_{3D} has a heavy implementation compared to simpler method such as AFF color corrections, resulting in \mathcal{T} per image 20 to 100 times superior to AFF color corrections.

Moreover, we proposed 2 criteria to detect failed corrections using the $\overline{\Delta_{RGB,within}}$ and $\Delta_{RGB,pairwise}$ metrics. These criteria discovered failed corrections over the dataset which heavily affected TPS_{3D}. Our proposal to approximate the TPS_{3D} formulation by a smoothing factor proved the right way to systemically remove those ill-conditioned scenarios.

Furthermore, we compared different RBF into the TPS_{3D} formulation. We did not prove a significance improvement in the color correction, although we did find that our proposed RBF would improve by a 30% the results regarding the \mathcal{T} per image.

Finally, we demonstrated the \mathcal{T} increases linearly with the image size for all the compared color corrections, enabling to take into account this variable when designing future color correction pipelines.

Regarding future work to improve this color correction framework, let us highlight some alternatives.

First, the systematic increase of color references should lead to a systematic improvement. This was not explored in the presented work in the chapter, as explained we preferred to use an established dataset which contained only images with the original ColorChecker with 24 color patches [13].

Thus, if creating a new dataset, one could add to the images modern color charts from X-Rite[®] which include up to 140 colors, as other authors did [15; 154; 155]. Or, one could use directly our proposal of chapter 4 to encode the same 140 color references in our proposed back-compatible QR Code.

We deepen into this idea of using our machine-readable patterns in chapter 6, where we used a Color QR Code to embed 125 colors of the RGB cube and use those colors with the above-presented color correction framework.

Second, we proposed this color correction framework as a solution to a general-purpose image consistency scenario. Often, colorimetric problems present themselves as a more reduced problem, i.e. we only need to seek for color correction in a certain subset of colors. If this is the case, instead of increasing the amount of correction colors we could reduce the colors to perform the color correction.

When doing so, we ought to select the color references which are *near our data points*, or at least *they are the most representative* version of our data within our correction colors. If not, the mapping will be poor in some parts of the data, as Donato et al. pointed out when discussing different approximation techniques for TPS mappings [125].

Third, there exist several authors that have explored different RBF that could be placed in the kernel definition of the TPS3D method, here we only discussed between two RBF which were solutions for 2D and 3D for the thin-plate spline solution. Theoretically, any RBF could be used [144], even more modern smooth bump functions [156; 157]

Correction	#1	#2	#3			#4	#5			#6	#7
	-	-	μ	σ	$\tilde{\mu}$	μ	μ	σ	$\tilde{\mu}$	μ	μ
PERF	0	0	0.99	0.12	0.99	0.223	0.945	0.019	0.949	0.214	8
NONE	0	0	59	27	56	13	52	23	50	12	11
AFF0	5519	2018	37	21	36	8	39	22	37	9	17
AFF1	2605	2152	24	18	21	5	24	19	22	6	30
AFF2	38	24	22	14	21	5.1	30	17	28	7	96
AFF3	19	9	12	10	10	2.7	17	12	15	3.9	110
VAN0	9	1	10	8	8	2.2	20	22	14	4	142
VAN1	10	2	8	7	6	1.9	20	23	14	5	172
VAN2	35	23	8	7	5	1.8	30	40	20	8	397
VAN3	176	154	8	8	5	1.7	40	40	20	9	542
CHE0	10	2	11	9	9	2.6	17	15	14	3.9	199
CHE1	9	1	10	9	8	2.3	18	18	14	4	210
CHE2	11	2	10	8	8	2.2	18	22	13	4	202
CHE3	16	6	9	7	7	2.0	20	25	14	5	219
FIN0	56	46	14	11	13	3.2	26	24	21	6	212
FIN1	29	21	19	12	18	4.4	29	18	27	7	271
FIN2	537	462	10	11	6	2.2	40	40	20	8	691
FIN3	346	193	17	11	15	3.9	42	34	34	10	958
TPS0	8133	8117	2	7	2	0.5	10	13	7	2.3	1067
TPS1	6359	6331	4	10	2	0.9	13	16	8	3	738
TPS2	9	1	3.5	3.3	2.2	0.8	13	11	10	2.9	697
TPS3	10	1	6	5	4	1.3	13	11	11	3.0	665

Table headers:

- #1: $\overline{\Delta_{RGB,within}} - \overline{\Delta_{RGB,within,NONE}} > 0$ [u.]
#2: $\min(\Delta_{RGB,pairwise}) < \sqrt{3}$ [-]
#3: $\overline{\Delta_{RGB,within}}$ [-]
#4: $\overline{\Delta_{RGB,within}}$ [%]
#5: $\overline{\Delta_{RGB,inter}}$ [-]
#6: $\overline{\Delta_{RGB,inter}}$ [%]
#7: \mathcal{T} [ms]

Table 5.3: A summary of the presented results. The summary includes metrics for each color correction for 7 different metrics (see left), the within-distances and inter-distances also include some statistical information such as the mean (μ), the standard deviation (σ) and the median ($\tilde{\mu}$). The median should be considered as the reference figure in those metrics as their distributions are quite asymmetric.

Chapter 6. Application: Colorimetric indicators

In previous chapters, we presented the need to achieve image consistency in datasets, and how this need relates with the capacity to perform quantitative color measurements over those datasets. Also, we discussed how this need is relevant in several industries. In this chapter, we are going to focus on analytical chemistry [4], specifically in environmental sensing.

Environmental sensing a wide field of research, for example, one could tackle the problem using very-low power electronic sensors [158]. Or, one could use *colorimetric indicators*. Colorimetric indicators are present in our daily life as humidity [39], temperature [40] or gas sensors [41; 42].

Usually, colorimetric indicators feature chemical reactions which act as a sensor or dosimeter for a certain substance or physical magnitude, a change on these magnitudes is then transduced into a change in the color of the chemical solution, i.e. a pH change induces a change in the chemical structure of a molecule which renders the color change (see Figure 6.1).

Moreover, colorimetric indicators are inexpensive and disposable, and simple to fabricate: i.e. printing them on top of a cellulose paper [31].

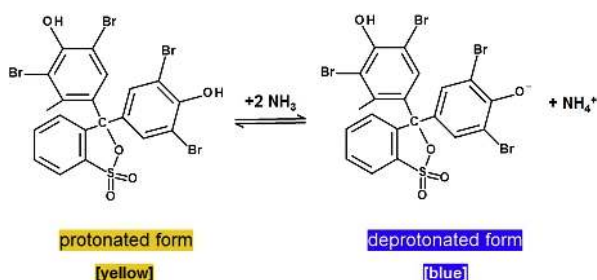


Figure 6.1: Reaction mechanism of the pH indicator bromocresol green (BCG, pH 3.8–5.4) for the detection of NH_3 . Increase of the NH_3 concentration leads to a proton release, detectable as a color change from yellow over green to blue.

In 2017, within a related research, we presented a solution to detect nitrogen dioxide (NO_2) in the environment using a colorimetric indicator. In that work, the colorimetric indicators were prepared soaking sterilized absorbent cellulose into the reactive ink. The results successfully conclude it was possible to measure air concentrations of NO_2 from 1 ppm to 300 ppm using a colorimetric indicator [159; 3] (see Figure 6.2).

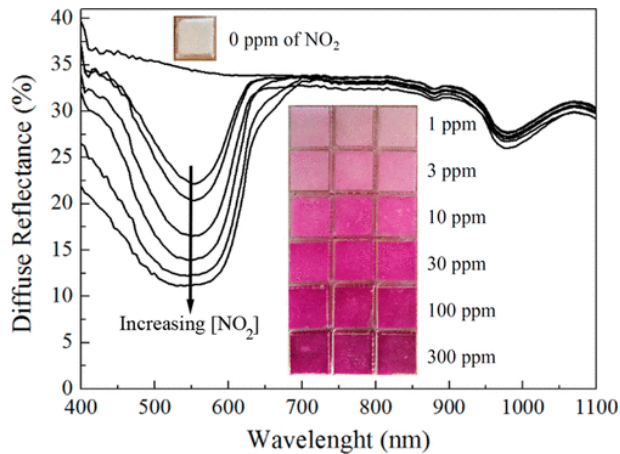


Figure 6.2: UV-VIS diffuse reflectance of the soaked pads with Griess-Saltzman reagent exposed to different NO_2 concentrations and the corresponding images of the colors developed (insets, 3 replicas per concentration).

Furthermore, in 2017, we presented a solution to detect ammonia (NH_3) in the environment with the use of colorimetric indicators. In this case, the colorimetric indicators were created dip-coating a glass substrate in a solution containing the reactive ink (see Figure 6.3). Results shown that it was possible to measure concentrations of NH_3 from 10 to 100 ppm [160].



Figure 6.3: Left, an ammonia (NH_3) colorimetric indicator has been dip-coated into a glass substrate, which exhibits a yellow color when exposed to synthetic air. Right, the same sensor is exposed to 100 ppm of NH_3 and it turns into purple.

In both works, we did not measure the color with digital cameras. On one hand, the NO_2 sensor was enclosed in a setup with a fixed one-pixel RGB sensor and several LED acting as a light source. In fact, we have continued to contribute to this line of research, enclosing colorimetric sensors in controlled compact and cost-effective fixed setups [45; 161; 162].

On the other hand, the NH_3 sensor was studied using standard spectrophotometry and sRGB color was computed from the measured spectra (see Figure 6.4). We pursued this line of research further in parallel to the development of this thesis [29; 163; 30].

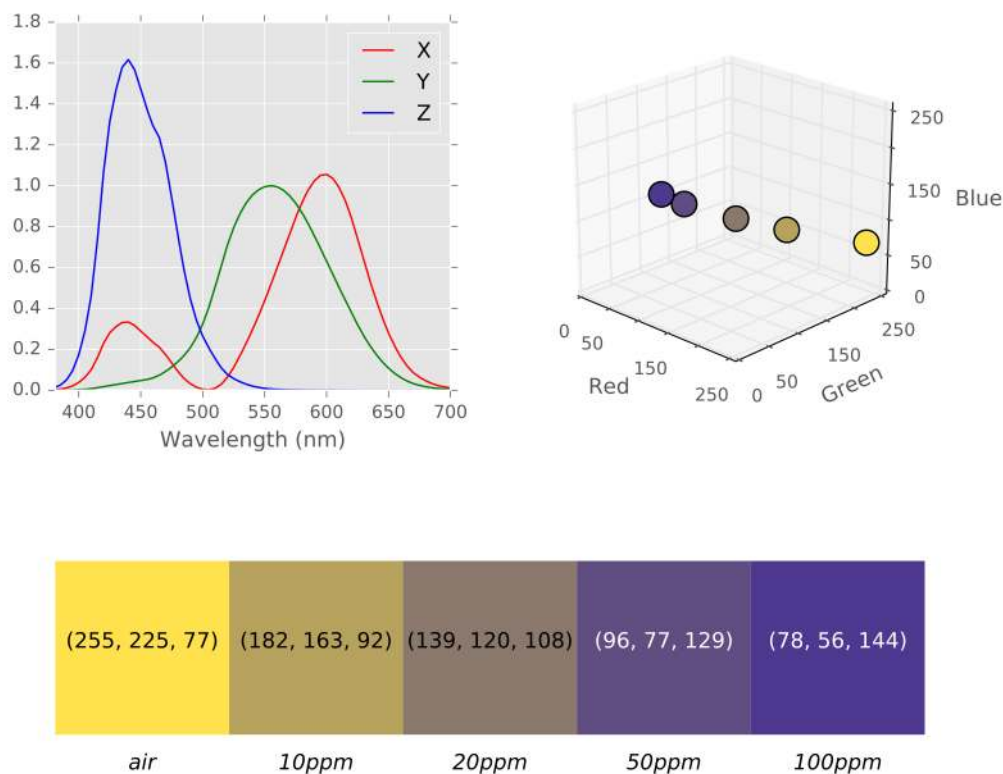


Figure 6.4: (a) Standard tristimulus $X(\lambda)$, $Y(\lambda)$, $Z(\lambda)$ curves of the human eye. (b) The integrated sRGB colors represented in the RGB cube. (c) The rendered sequence of RGB colors corresponding to the gas sensing spectra $c(\lambda)$.

As a wrap-up, in this chapter we present the different partial approaches to combine colorimetric indicators with our thesis proposal of Color QR Codes. The partial solutions were applied to different target gases, such as ammonia, (NH_3), hydrogen sulfide (H_2S), etc.

Finally, we present here a carbon dioxide (CO_2) sensor featuring a Color QR Code. The Color QR Code enables to: extract the sensor from any surface (chapter 3), embed inside or outside the QR Code the sensor ink (chapter 4) alongside with color references to perform a color correction using a whole framework of corrections (chapter 5).

6.1 Proposal

6.1.1 Early prototypes

In 2018, we presented a solution [29] to automate the readout of an environmental colorimetric indicator that was developed to detect NH_3 [160]. This solution preceded most of the research above-presented in this thesis.

The proposal was to design a machine-readable pattern resembling a QR Code, without any digital data, to allocate color references and two reserved areas in order to print the colorimetric ink. The whole process of design, fabrication and interrogation is described in Figure 6.5.

The machine-readable pattern would maintain the finder, alignment and timing patterns of QR Codes (more details in chapter 2). see Figure 6.6 shows an example of these machine-readable patterns designed to embed a NH_3 sensor.

The first downside of this proposal is the way the color references are generated. These colors derived from the measures of the ink color when exposed to different amounts of the target gas – NH_3 – (see Figure 6.7), and then classified into a subset of colors – e.g. 32 colors – (see Figure 6.8). Later, when the machine-readable pattern is printed the color might differ from the measured color. This is a perfect example of solving the problem of *color reproduction*. As we discussed in previous chapters, we preferred to solve the *image consistency* problem, i.e. place more color references than only those colors from the sensor.

The second downside was the way these color references were encoded in the QR Code-like pattern. As we cleared all the digital information area, we invalidated one of our goals: to achieve a back-compatible QR Code for colorimetric applications. Also, this proposal embedded the colors in 3×3 module blocks as we did not develop yet the proper methods to correctly perform a successful extraction in challenging surfaces without significant readout failures.

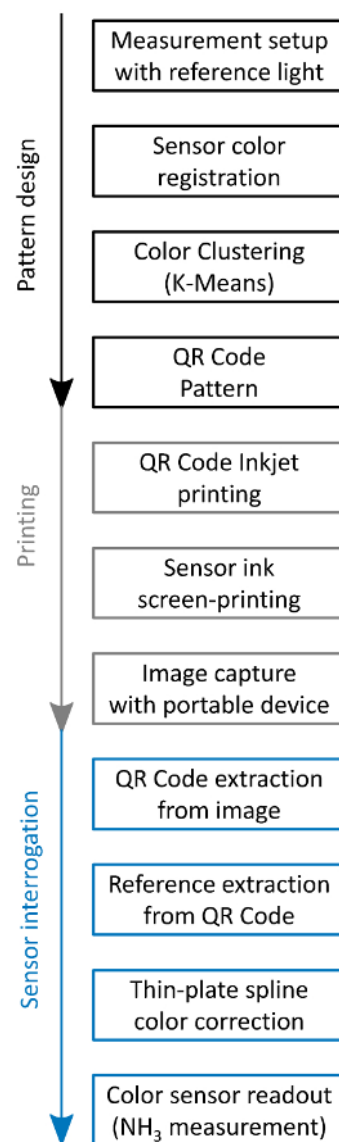
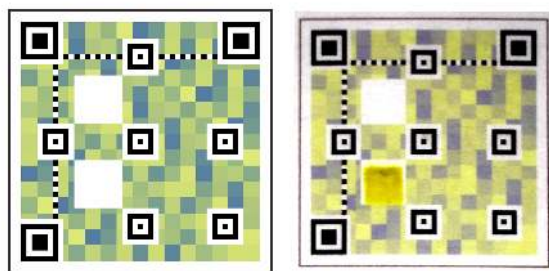


Figure 6.5: The proposed pipeline for creating machine-readable patterns proposed in 2018 [29].

Figure 6.6: A machine-readable pattern to allocate an ammonia sensor. Left: the designed pattern, with two spaces to print a colorimetric sensor. Right: the captured version of the pattern with a printed colorimetric dye in one slot. Notice this pattern resembles a QR Code, but it does not contain any data.

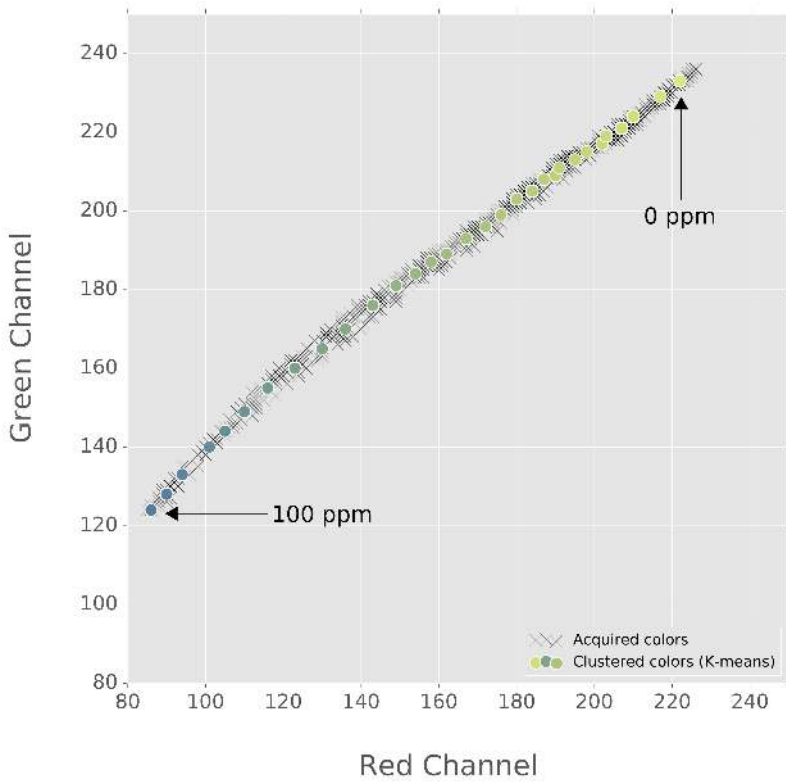


Figure 6.7: RGB 8-bit color data acquired from a colorimetric sensor captured with a digital camera at 5500K color temperature exposition, with the centers of 32 clusters generated by K-means clustering. Data is presented as a projection into the red-channel plane of the RGB space.

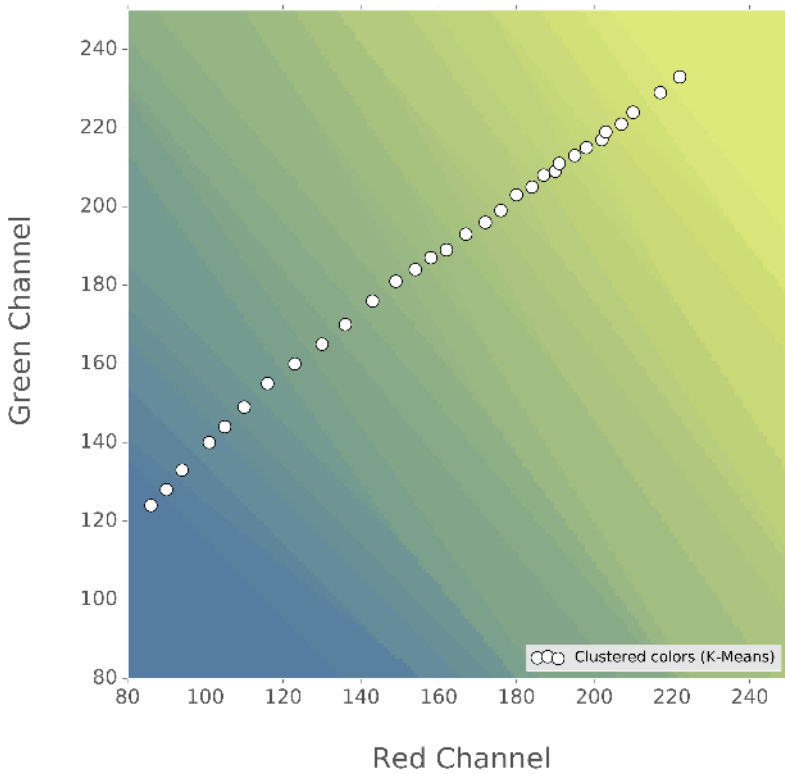


Figure 6.8: 32 clusters centers from Figure 6.7 data, and color clustering regions. Data is presented as a projection into the red-channel plane of the RGB space.

6.1.2 A machine-readable pattern for colorimetric indicators

In 2020, we introduced our improved proposal for a machine-readable pattern for colorimetric indicators [163; 30]. This approach maintained the use of a QR Code-like machine-readable pattern without digital data, only allocating the sensor ink, the color references and computer vision patterns to perform the readout.

However, as we improved our computer vision algorithms to capture QR Codes, we were able to add more complex color encoding to the pattern definition. Then, the number of embedded color references in the pattern was considerably increased, and with so the color correction method was improved (see Figure 6.9).

This proposal tackled many aspects of the sensor readout improving the former one:

- The factor form of a QR Code version 7 was maintained, preserving the alignment pattern array to ease the readout.
- An additional pattern to ease the location of the fourth corner was added.
- The reference colors were embedded as 1×1 modules, this is as the same size of a QR Code module.
- Two palettes of reference colors were embedded with a total of 245 colors: 125 colors from an RGB excursion, with 5 samples per channel; and 120 colors from an HSL excursion, with 6 samples for the H channel, 4 for the S channel and 5 for the L channel.
- A fabrication process which involved screen-printing instead of dip-coating or other techniques, improving the sensor reproduction.
- An improved version of the TPS_{3D} color correction which accounted for ill-conditioned scenarios and corrected them if possible with a fall-back mechanism into affine transformations.

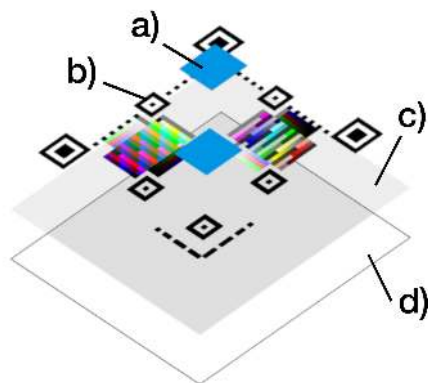
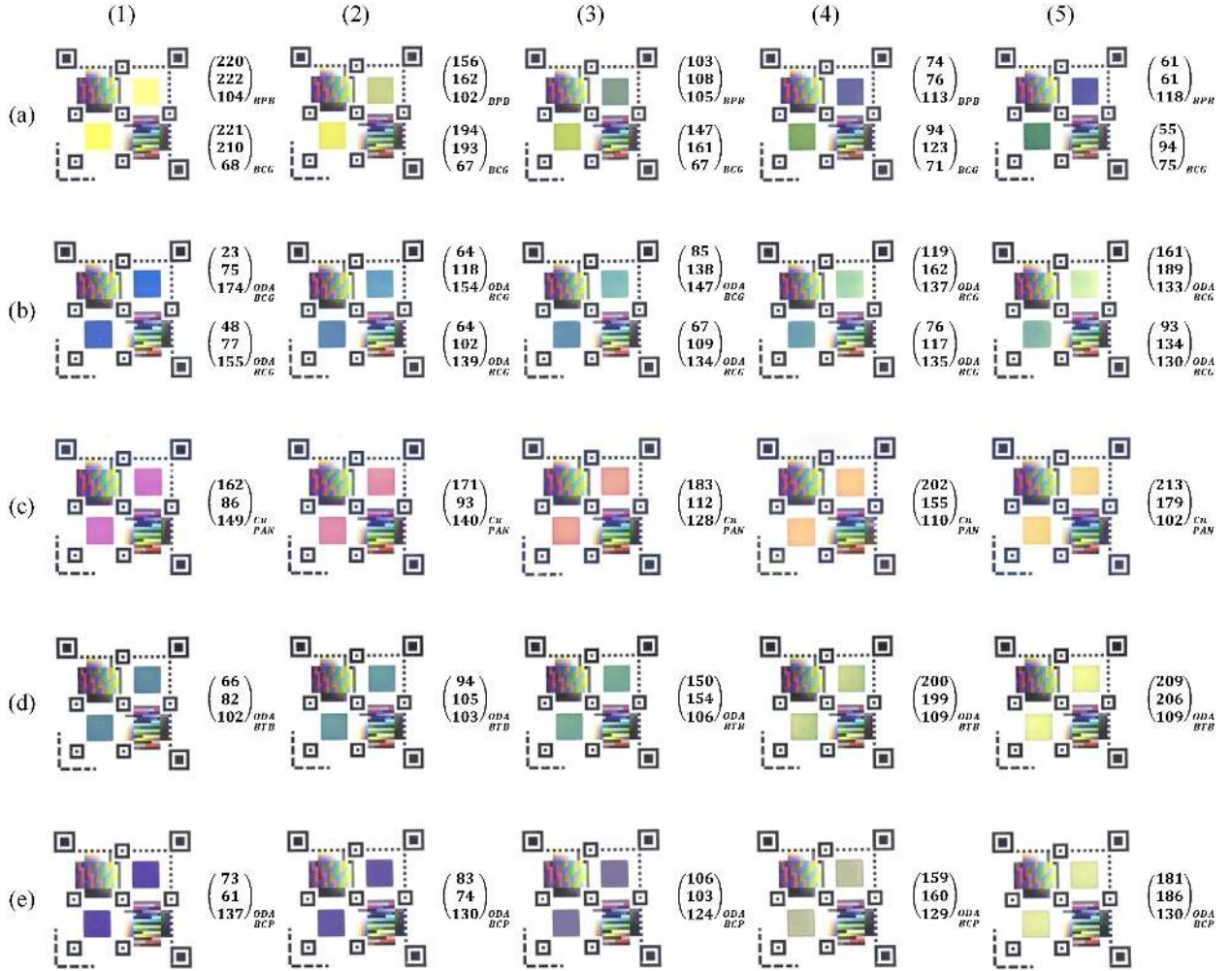


Figure 6.9: The layer structure of the machine-readable pattern for colorimetric indicators: a) the colorimetric indicator ink, b) the machine-readable pattern inks, c) the plastic substrate and d) white cardboard.



All in all, we successfully applied this proposal of machine-readable pattern which resembled a QR Code into several colorimetric indicators that targeted different environmental gases [30] (see Figure 6.10):

- **NH₃**: an ammonia sensor was presented as a result of combining two colorimetric indicators, *bromophenol blue* (BPB) and *bromocresol green* (BCG).
- **H₂S**: a hydrogen sulfide dosimeter was presented using a *copper complex of an azo dye* (Cu-PAN) as colorimetric indicator.
- **CH₂O**: three different dosimeter were presented to measure formaldehyde, based on three different colorimetric indicators. Once again, based on the *bromophenol blue* (BPB) and *bromocresol green* (BCG), with the addition of an extra compound, *octadecylamine* (ODA), to tune the original indicator to this target gas.

Figure 6.10: Five machine-readable patterns (a), (b), (c), (d) and (e) are exposed to different atmospheres (1), (2), (3), (4), (5), the value of the mean measured RGB color for each ink at each atmosphere is represented as a transposed vector. (a) a NH₃ sensor using the BPB and BGC indicators. (b) a CH₂O dosimeter using the BGC indicator. (c) a H₂S dosimeter using the Cu-PAN. (d) a CH₂O dosimeter using the BTB+ODA indicator. And, (e) a CH₂O dosimeter using the BCP+ODA indicator. The different 5 atmospheric conditions can be consulted in Engel et al. [30].

6.1.3 A Color QR Code for colorimetric indicators

Here, we present a Color QR Code for colorimetric indicators which features fully-functional back-compatibility, this means it can be read with any commercial QR Code scanner and a URL, or other desired message, is obtained (see Figure 6.11). The main specifications of these machine-readable patterns are:

- they use the full standard of the QR Code. Thus:
 - enabling extraction techniques similar to the ones presented in chapter 3, which leads to ease the location of colorimetric elements in the same scene, placed outside the QR Code (see Figure 6.12.a), or inside (see Figure 6.12.b),
 - plus, they can be designed in any desired version of the QR Code standard, the current proposal uses a version 3 instead of a version 7 (reducing the physical size of the sensor),
 - and they can encode any desired information, for example a URL with a variable ID, rendering almost infinite possibilities (see Figure 6.13)
- Also, they can embed hundreds of colors, following our technique to create back-compatible QR Codes, as we detailed in chapter 4. This implies:
 - they contain by default 125 RGB colors ($5 \times 5 \times 5$),
 - 100 of them are encoded in the DATA zone of the QR Code,
 - 25 of them are encoded in the EC zone of the QR Code,
 - and more extra space to allocate reactive inks, that always be computed in the design of the QR Code as error.

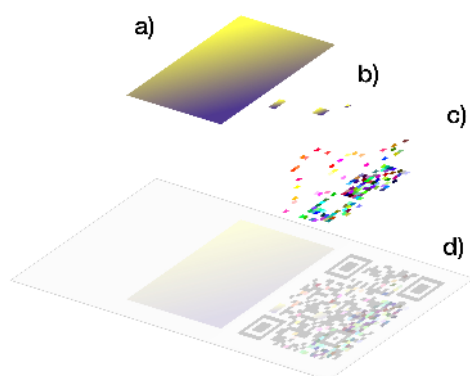


Figure 6.11: A back-compatible Color QR Code for colorimetric indicators. This QR Code will be read by commercial scanners, and it should display the URL: `c-s.is/#38RmtGVV6RQ5f`. The Color QR Code includes up to 125 reference colors and blank space to allocate a colorimetric indicator ink (above the lower finder pattern).

Figure 6.12: The structure of the Color QR Code from Figure 6.11 is detailed. a) and b) show possible sensor inks placements, a) shows a big sensor outside the QR Code, b) shows smaller factor forms (3×2 , 1×1 , ...) inside the QR Code. c) Shows the color references and how they are spread in the QR Code areas. Finally, d) shows the whole layout of the sensor with the Color QR Code.

- Moreover, as they can embed such quantity of colors (see Figure 6.12):
 - the palette of colors can be fine-tuned depending on the application to solve, as we explained in chapter 5 this leads a path to systematically enhance color corrections,
 - plus, redundancy can be added if the palette is reduced, for example, if the ColorChecker palette is suitable for a certain problem it can be embedded several times,
 - then, with these colors references all the available framework of color corrections presented in chapter 5 can be applied.

This proposal is a wrap up of the before studied technologies, which combines the practical use case of colorimetric sensors with our thesis proposal to use QR Codes to embed color references to act as a color chart.

In the subsequent sections, we present the results of using a Color QR Code, from the same batch as Figure 6.13, to measure and calibrate a CO₂ sensor, based on the *m-cresol purple* (mCP) and *phenol red* (PR) colorimetric indicators [164; 165; 166; 31]. The measurements were performed in a dedicated setup with artificial an atmosphere and different light conditions. The results show how different color correction techniques from our framework yielded to different calibration models results for the sensor.



Figure 6.13: 16 different Color QR Codes for colorimetric indicators with different encoded data that differs in an alphanumeric ID. The encoded reference colors in each QR Code is the same, however the position of the colors is distributed following the digital data in a back-compatible manner. Each Color QR Code has a reserved area (white) above the lower finder pattern to allocate a colorimetric ink.

6.2 Experimental details

6.2.1 Sensor fabrication

We had previously fabricated colorimetric indicators in several forms: soaked cellulose [159], dip-coating [160] or screen-printing [163]. The later method provides a more reliable fabrication method in terms of reproducibility. Also, as a printing method is the entry point to other printing techniques such as rotogravure or flexography, among other industrial printing technologies [10].

Then, we fabricated our current sensors using a *screen-printing* manual machine in our laboratory. The screens were created according to the designs before-mentioned in the previous section. The substrate was a *coated white cardboard of 350 gr*, the coating was *mate polypropylene*. And, the Color QR Codes were previously printed using ink-jet technology (see Figure 6.14).

Sensors were printed in batches including per each Color QR Code: a CO_2 sensor, based on the mCP+PR indicators; and a NH_3 sensor [30], based on the BPB color indicator. Here, we focused only in the CO_2 , which is the blueish sensor before exposing it to the CO_2 target concentrations (see Figure 6.15).

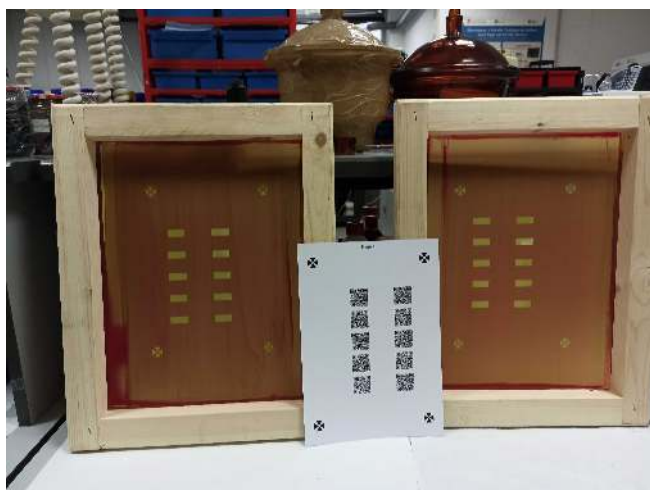


Figure 6.14: Two screens and one substrate sheet. Each screen can print one color indicator, and both can be combined into the same pattern. The substrate has DIN A4 measures, it also contains up to 10 Color QR Codes with an approximated size of 1 inch.

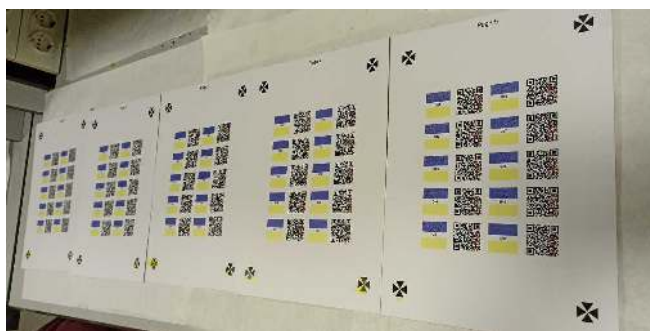
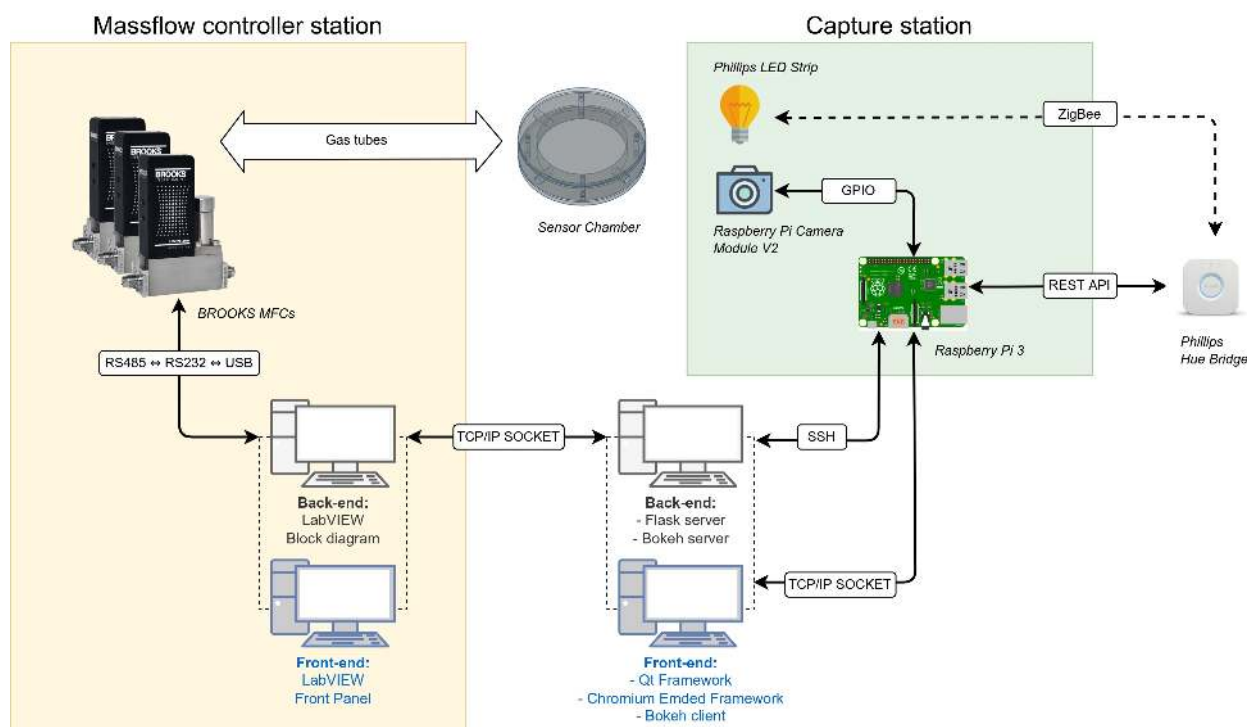


Figure 6.15: Several substrate sheets already printed, each sheet contains up to 10 CO_2 sensors and 10 NH_3 sensors.

6.2.2 Experimental setup



We designed and built our experimental setup from scratch. This experimental setup was used in the research of this thesis and related research, i.e. seeking for new colorimetric indicators. The setup consists of a complex system which responds to the necessity to capture colorimetric indicators targeting specific gases. Thus, the setup needed to solve not only the optical measurement, but also the management of target gas lines (see Figure 6.16).

The setup consisted of three main subsystems:

1. **Mass-flow control station:** a tiny desktop computer, a Lenovo ThinkCentre M93 Tiny, implemented the software to control up to five BROOKS 5850S mass-flow controllers [167], which fed a sensor chamber with the desired gas mix (see Figure 6.17). The control over the BROOKS devices was done in LabVIEW® [168].

Also, a LabVIEW front-end screen was implemented to enable user interaction in this subsystem. The tiny computer was equipped with a touchscreen to ease user interaction, but usually this station is set to automatic when using the setup to perform long-term data acquisition.

Moreover, LabVIEW used a serial communication protocol using the BROOKS official Windows DLL [169] with some hardware protocol adaptations performed (USB ↔ RS232 ↔ RS485).

Figure 6.16: Schema of our laboratory setup. The setup features 3 subsystems: a massflow controller station, a capture station and a user-access computer. The massflow controller station provides modified atmospheres to a chamber where the gas sensors are placed. The capture station can see the chamber through an optical window, and take time-lapses with a controlled light setting. Finally, the user computer presents a web page interface to the user to operate the system.

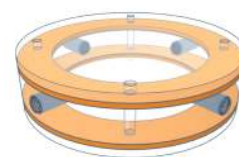


Figure 6.17: The 3D design of the circular sensor chamber. The chamber is transparent to enable optical readings, and it is sealed using rubber (orange). The chamber also has four threaded input/output holes.

2. **Capture station:** a single-board computer, a Raspberry Pi 3B+, implemented the software to control a digital camera (Raspberry Pi Camera v2) [134] and a strip of RGB LEDs from Phillips Hue [135] which acted as a variable light source.

Then, the control software of both the camera and the light strip was implemented in Python [151]. Specifically, the `picamera` module was used to drive the camera, and the `phue` one to drive the LED strip ¹.

3. **User station:** a desktop computer, from Hewlett-Packard, implemented the user control software to manage all the system. This software was implemented using Python again, but with a different stack in mind: `flask` was used as a back-end service [170], and `bokeh` was used to present plots in the front-end [171]. The front-end was based upon a web-based technology that uses the popular Chromium Embedded Framework to contain the main application [172].

6.2.2.1 Gas concentrations

The colorimetric indicator was exposed to a series of pulses of different controlled atmospheres. In total, it was exposed to 15 pulses of 100 minutes each pulse. Each pulse consisted of exposing the sensor 30 min to the target gas CO_2 , followed by an exposure of 70 min to a synthetic atmosphere without the target gas. This let the experiment last for a day.

Table 6.1 shows the expected concentration of the CO_2 pulses versus the measured and corrected against dilution laws, as indicated by the manufacturer [167]. The target gas was diluted using a synthetic atmosphere (21 % oxygen, 79% nitrogen). We configured the experiment to repeat 3 times the same pulse for 5 different target concentrations: 20%, 30%, 35%, 40% and 50%.

These concentrations were selected as the CO_2 indicator was designed to tackle the scenario of modified atmosphere packages (MAP) – 20% to 40% of CO_2 – [32]. Also, this is why, the synthetic atmosphere was partially exposed to a humidifier to achieve proper work conditions for the colorimetric indicator, those resembling a MAP containing some fresh food, i.e. meat, fish or vegetable.

Figure 6.18 shows a detailed view on the above-mentioned gas pulses, all the measures are shown for the target gas channel (CO_2), and both the dry and the humid synthetic atmospheres (SA).

¹ Note this part of the setup was also used in chapter 4 when exposing the Color QR Codes to the *colorimetry setup* channel.

Pulse	Expected [%]	Measured [%]
1	20.0	25.21 ± 0.00
2	20.0	25.22 ± 0.22
3	20.0	25.22 ± 0.22
4	30.0	36.62 ± 0.22
5	30.0	36.67 ± 0.31
6	30.0	36.67 ± 0.31
7	35.0	42.10 ± 0.40
8	35.0	42.30 ± 0.40
9	35.0	42.20 ± 0.40
10	40.0	46.90 ± 0.40
11	40.0	47.30 ± 0.40
12	40.0	47.40 ± 0.40
13	50.0	57.60 ± 0.50
14	50.0	57.60 ± 0.50
15	50.0	57.60 ± 0.50

Table 6.1: The expected and the measured gas concentration for each gas pulse is shown. The measured values were taken from the BROOKS instrumentation reading while applying a correction algorithm provided by the manufacturer [167].

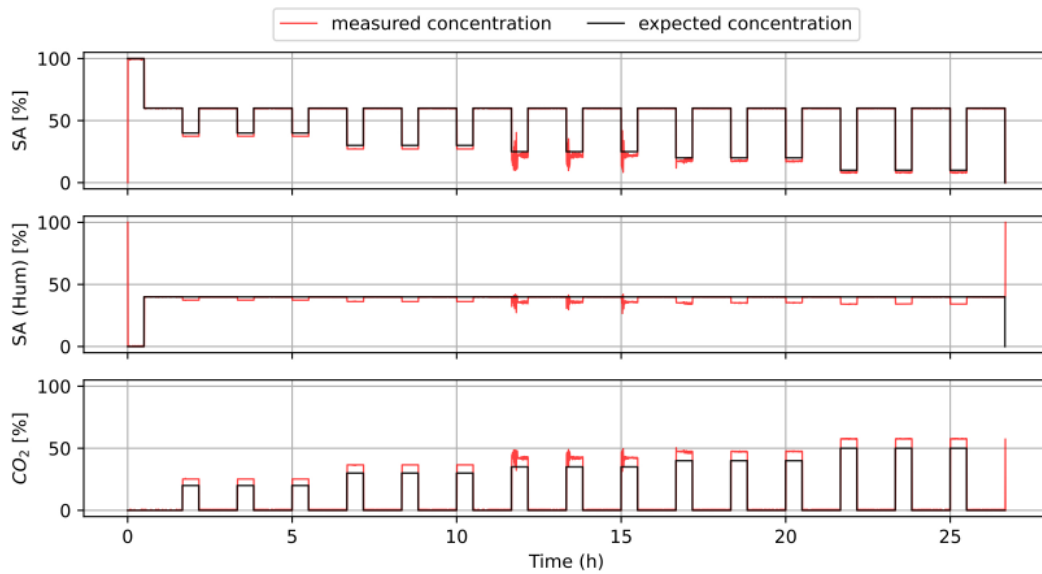


Figure 6.18: The expected (black) and the measured gas concentration (red) for each gas pulse is shown on a temporal axis along the experiment duration. The measured values were taken from the BROOKS instrumentation reading while applying a correction algorithm provided by the manufacturer [167].

6.2.2.2 Capture settings

The sensor was exposed to different light conditions of white light. This was achieved using the above-mentioned Phillips Hue light system. Color temperatures ranged from 2500K to 6500K, in steps of 500K (see Figure 6.19). These are a less aggressive light conditions than those we used in chapter 4 when studying the capacity of QR Codes in the colorimetry setup.

Also, the camera settings were fixed, without auto-exposition nor auto white-balance, to capture color consistent images through all the dataset. This ensures us we can extract the color reference tables during color correction only in the 9 first images.



Notice, that the number of different light illuminations (9) was kept low to preserve an adequate sampling rate of the sensor dynamic. As, our setup performs the captures in a synchronous way: an image is taken, then the color illumination changes, then another image is taken, etc. The global sampling rate was 1 FPS, which is the maximum frame rate a Raspberry Pi Camera can process at Full HD quality (1920×1088 pixels). Then, the actual frame rate for each illumination stream was $1/9$ FPS.

Figure 6.19: A printed sensor, featuring a Color QR Code and two colorimetric indicators is displayed inside the sensor chamber of our setup. Then is exposed to different light conditions. From left to right, the illumination changes following 3 color temperatures of white light: 2500K, 4500K and 6500K.

6.2.3 Expected response model

In previous works, we already studied the relation between the colorimetric response of an indicator with the presence of the target gas [30; 29; 160; 163]. The relation we found was:

$$S[\%] = m \log(c) + n \quad (6.1)$$

where S is the colorimetric response of the sensor, c is the concentration of the target gas in the atmosphere, m and n are the constants of a linear law. Then, the colorimetric response is linear with the logarithm of the gas concentration. m represents the sensitivity towards the logarithm of the gas concentration, and n the response at very low concentrations.

Also, we can recover the sensitivity as a function of the gas concentration using derivatives and use it to compute the error of the model for each concentration. To do so, we will use error propagation rules:

$$\left. \frac{\Delta S}{\Delta c} \right|_c = \frac{m}{c} \implies \Delta c|_c = \Delta S|_c \cdot \frac{c}{m} \quad (6.2)$$

where c is a given concentration recovered with the inverted Equation 6.1, m depends on each fitted model, $\Delta S|_c$ is the error of the measured signal response and $\Delta c|_c$ is the model error for this given concentration.

Finally, the signal color response $S[\%]$ is usually normalized following a metric. We defined this metric to resemble the normalization performed in an electronic gas sensor [30]. This kind of normalization divides the measured signal by the signal value assigned to zero gas concentration, this produces a metric that is not upper-bounded $[0, \infty)$, the more the initial value is to *zero resistance*, the greater the response. Let us adapt this normalization for a red channel of a colorimetric indicator:

$$S_r[\%] = 100 \cdot \frac{r(c) - r_0}{r_0 - r_{ref}} \quad (6.3)$$

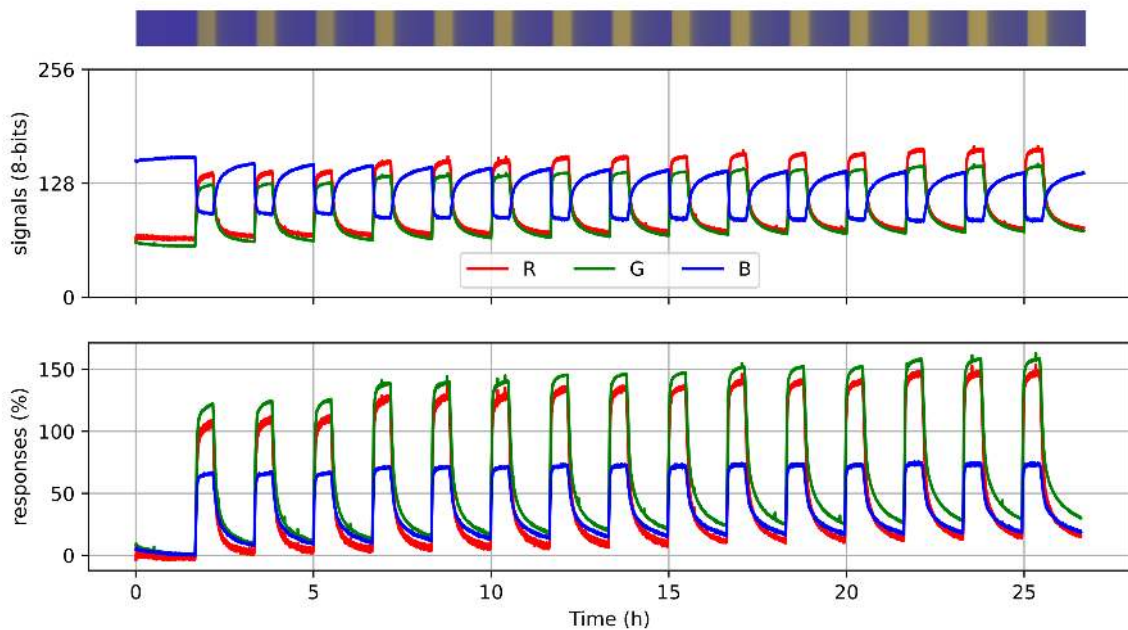
where S_r is the response in % of the red channel, c the concentration of the target gas in %, $r(c)$ the raw red sensor signal with an 8-bit resolution (0–255), r_0 the value of $r(c = 0\%)$ and r_{ref} an absolute color reference which acts as the *zero resistance* compared to electronic sensors, for our sensor the value is $(r_{ref}, g_{ref}, b_{ref}) = (0, 0, 255)$, as our measured blue channel signal decreases when the gas concentration increases [31].

6.3 Results

6.3.1 The color response

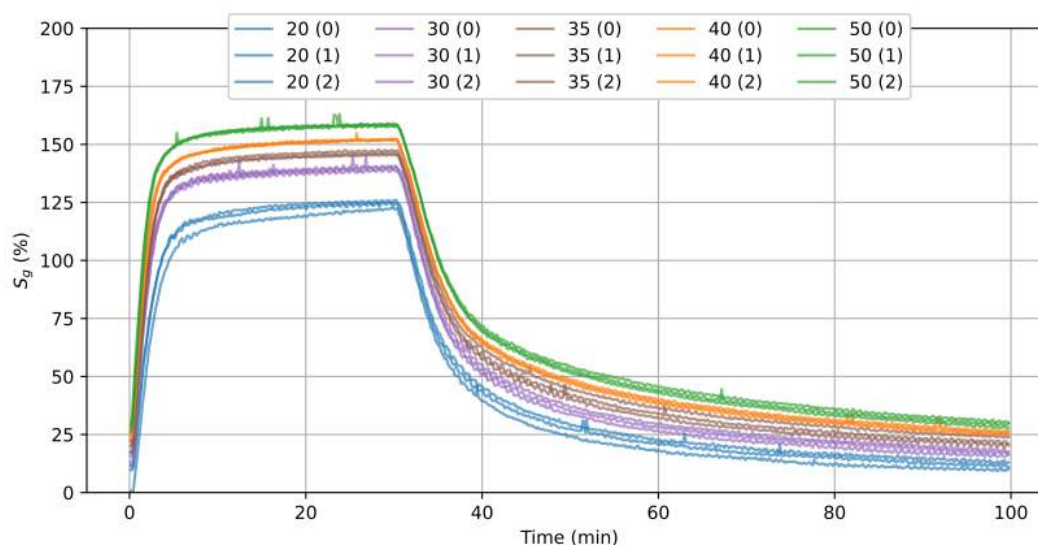
Let us start with the response of the colorimetric indicator under the different CO_2 atmospheres. In Figure 6.20 it is represented the obtained signals from the mean RGB channels for the colorimetric indicator for all the experiment captures in the D65 standard illuminant (6500K). In order to obtain these mean values, we created a computer vision algorithm to extract the region of interest. This algorithm was based upon the state-of-the-art, presented in chapter 2, and our above-presented work, in chapter 3.

First, with these results, we could already confirm that we correctly selected our absolute reference to compute the color response – $(r_{ref}, g_{ref}, b_{ref}) = (0, 0, 255)$ –. As the previous work suggested [31], the color indicator moves from a blueish color to a yellowish color with the appearance of CO_2 in the atmosphere.



Then, Figure 6.20 displays the computed responses from the sensor. The responses were computed following Equation 6.3. The results show the channel which achieved the best response was the **green channel**. The green channel presented a higher response and less noise. Followed by the red channel, which performed close to the green channel in response, but with a more accurate noise. The blue channel, performed with approximately the half of the response than the other channels at the lower concentration tested. And, its response saturated in the higher concentrations more rapidly than red and green channels.

Figure 6.20: From top to bottom: a representation of the color of the sensor over time for the D65 standard illuminant (6500K), it can be observed it changes from blueish colors to yellowish colors; the same colors as RGB channel signals; and the response (%) for all the RGB channels.

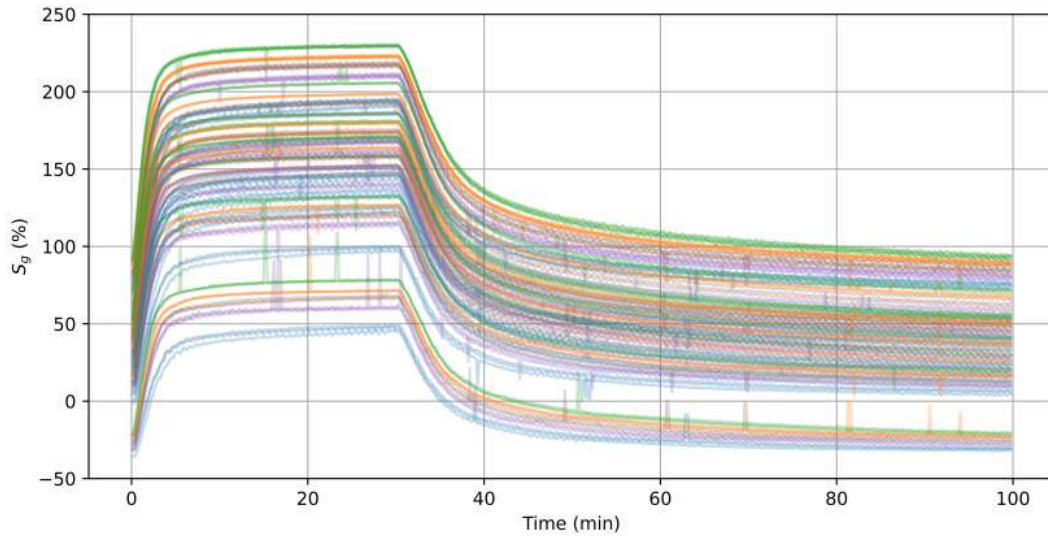


Moreover, in Figure 6.21 we present the previous results but now stacked as into the same time frame, the pulse duration of 100 minutes. This is interesting from the gasometric standpoint of view. Our colorimetric indicator presented:

- a fast response, achieving the 90% of the S_g (%) response in less than 5 minutes, for the highest concentration; and in less than 10 minutes, for the lower one,
- a reasonable maximum response above 150% for the higher target concentration of CO_2 50%,
- a slight saturation in the upper region of concentrations, this will be reflected in our fitted models,
- excellent reproducibility among the replicas of each pulse, except for one pulse in the first triplet (20% of target gas concentration),
- and, a drift in the lower concentration area, as pulses did not end in the same response they started.

All in all, Figure 6.21 presented our colorimetric indicator as a good choice to detect the concentration of CO_2 in modified atmosphere packaging (MAP). The caveats presented by the sensor: the saturation in the upper range (50%) and the drift in the lower range (0%) did not affect our further results as our models were targeting only to fit the data of the range 20% - 50% which applies to MAP.

Figure 6.21: The response of the green channel exposed to a D65 standard illuminant (6500K) for all the pulses are overlapped in the same time frame. This results in 15 pulses, each reference target gas concentration (20, 30, 35, 40, 50) has a pulse replica (0, 1, 2).



However, these results became meaningless when we exposed the colorimetric indicator to other light conditions than D65 standard illumination, as the apparent sensor response changed drastically, as expected. Figure 6.22 portrays how the different illuminations affected the measure of the response for all the above-mentioned 15 pulses.

Finally, it was the time to exploit the color correction framework studied in chapter 5. For each illumination, the 125 RGB colors placed inside the Color QR Code were extracted (see Figure 6.23). These colors were used to apply each one of the above-mentioned color correction techniques (see Table 5.1). All the images from each illumination were corrected (>10000 image/illumination).

The references of the D65 standard illumination were taken as the destiny color space of our correction techniques. Figure 6.24 shows how a TPS₃ improved the situation exposed in Figure 6.22 recovering a more suitable scenario to fit a colorimetric model to the data. In the text subsection, we focus in how we measured each pulse and fitting our proposed model to the different color corrections.

Figure 6.22: The response of the green channel exposed to nine illuminants (2500K to 6500K) for all the pulses are overlapped in the same time frame. This results in 135 apparent pulses, now each reference target gas concentration (20, 30, 35, 40, 50) has a pulse replica (0, 1, 2) for each illuminant. Legend is omitted to favor clearness, results should be compared to Figure 6.21.

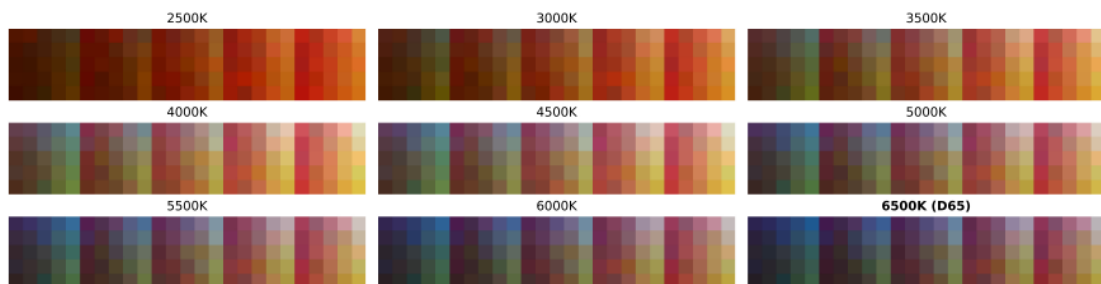
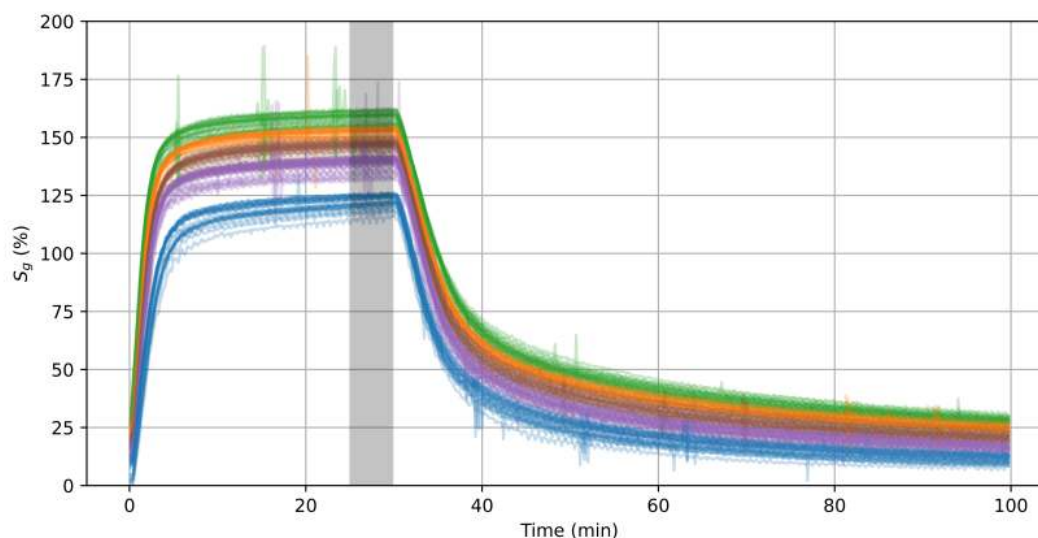


Figure 6.23: The captured color references from the Color QR Code in each illumination condition. D65 is the reference space.



6.3.2 Model fitting

After applying the color correction techniques, we prepared the data to be suitable to fit our proposed linear-logarithmic model (Equation 6.1), to do so we:

- measured a mean corrected response value from the data during the last five minutes of gas exposition. Figure 6.24 shows this time window shadowed;
- transformed these responses measures applying a logarithm,
- linked those responses to their respective gas concentration measures from Table 6.1;
- and, split the available data into train (75%) and validation subsets (25%).

Then we fed the train subsets, one for each available color correction in Table 5.1, to a linear model solver and obtained up to 22 different solutions, including the special NONE and PERF corrections described in chapter 5. The validation subsets were used after to evaluate the models.

Let us start with this two reference corrections, Figure 6.25 and shows the fitting for both corrections, and Figure 6.26 shows the validation results. Results indicated that **NONE** was the *worst case scenario*, thus without correction the measurement of the colorimetric indicator was impossible. And **PERF**, as expected, was the *best case scenario*.

Figure 6.24: The response of the green channel exposed to nine illuminants (2500K to 6500K), then corrected using the TPS₃ method (Table 5.1), for all the pulses are overlapped in the same time frame. This results in 135 apparent pulses, now each reference target gas concentration (20, 30, 35, 40, 50) has a pulse replica (0, 1, 2) for each illuminant. Legend is omitted to favor clearness, results should be compared to Figure 6.21. The shadowed area is the area corresponding to the 5 minutes windows used to integrate the response of the sensor for the model computation.

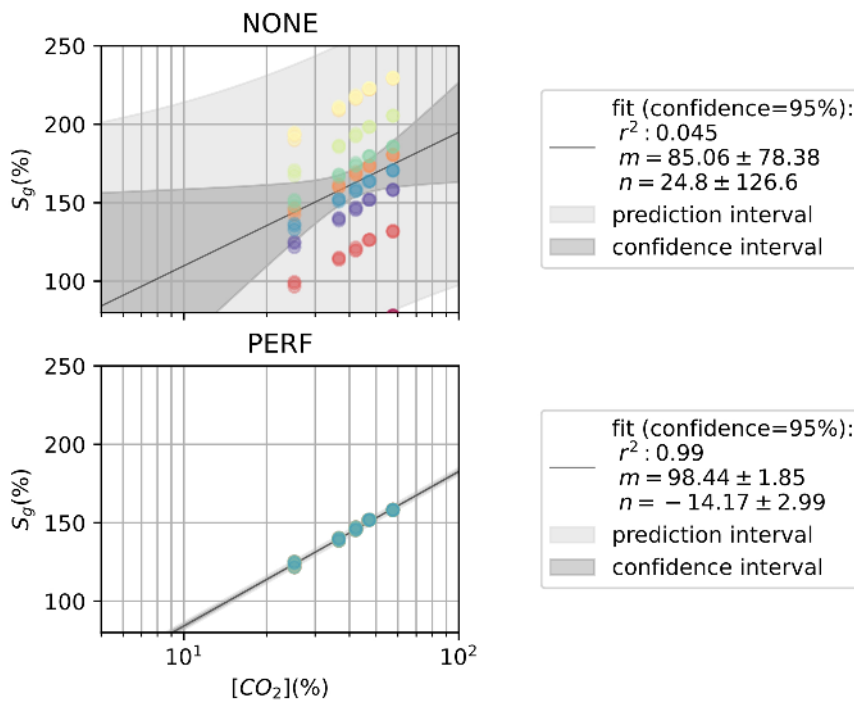


Figure 6.25: NONE and PERF fitted models, which represent the worst and the best case scenarios, respectively. The fitted model in NONE is the one performed without correcting any captured color. The PERF model is an artificial model in which each captured color has been mapped to its correspondent D65 color.

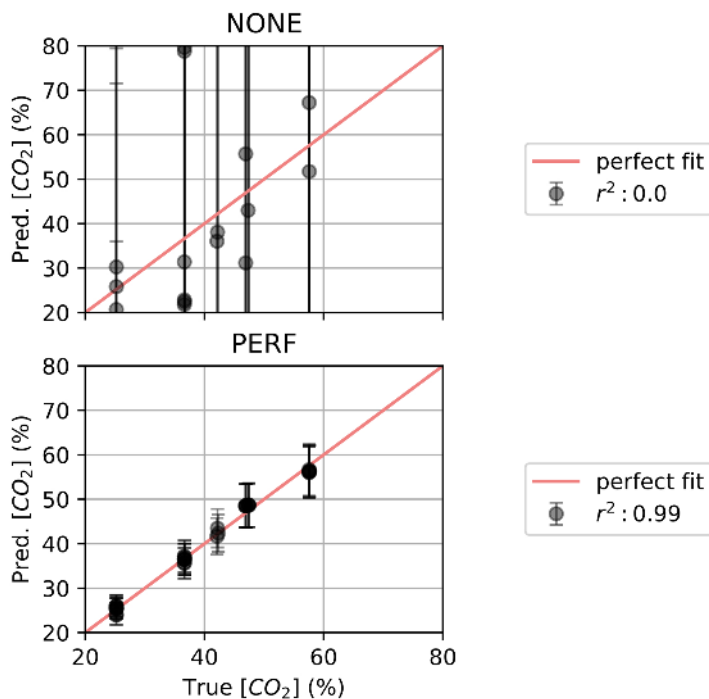


Figure 6.26: NONE and PERF regression for the validation data. The coefficient r^2 was computed for this data. This result confirms that NONE is the worst case scenario with a null r^2 , and PERF is the best score in the whole set of results.

Specifically, when we say PERF is the best case scenario we mean the following: as the PERF model is the model of acquiring the data in a fixed setup –with a fixed camera configuration, a fixed light conditions, etc.–, the problem which we aim to solve in this thesis, the image consistency problem, is not present in this data; which follows that the error seen in this model is the intrinsic error of the colorimetric indicator technology.

Then, the PERF results showed the good performance of the colorimetric indicator to sense CO_2 in the target range of gas concentrations, scoring both r^2 metrics for training (Figure 6.25) and validation (Figure 6.26) almost a perfect score. *This confirmed our model proposal.*

Let us compare now the subsequent color corrections (Table 5.1) with the before-mentioned extreme cases, the results are displayed from Figure 6.27 to Figure 6.36:

- **AFF:** AFF₀, AFF₁ and AFF₂ showed, in that order, the worst results in the dataset, both in training (see Figure 6.27) and validation (see Figure 6.28). Although the data was biased towards this group of corrections. The bias is explained with the fact that we only changed color temperature of white illuminants in our setup.

However, AFF₃ scored the best results above the whole other corrections groups, this is the correction which best approximated to the PERF solution.

This is explained by the bias, but also because a general affine correction has contributions not present in a simple white-balance (AFF₀, AFF₁) or the affine without translation (AFF₂).

- **VAN:** all corrections showed similar result to AFF₃, but neither of them accomplished to outperform it. From the standpoint of the training model all four models showed the same fitting, if error correction is taken into account (see Figure 6.29). From validation, the results showed also the same metric once again, good approaches to PERF (see Figure 6.30).
- **CHE:** all corrections showed a similar result to AFF₃ in training (see Figure 6.31), the same as all VAN corrections. Only CHE₀ showed a slightly worse metric in validation (see Figure 6.32), which is non-significant.
- **FIN:** FIN₀ presented similar results to AFF₃ both in training (see Figure 6.33) and validation (see Figure 6.34). However, FIN₁, its root-polynomial version, presented worse results, both in training and validation. FIN₂ and FIN₃, the order 3 versions of FIN₀ and FIN₁, also failed to correct the color responses properly to fit the data.

- **TPS:** all corrections showed good results, close to AFF₃. In training, all the corrections showed the same fitting, if error correction is taken into account (see Figure 6.35). As for validation, TPS₀ and TPS₁ achieved the best scores. (see Figure 6.36) Note here, TPS₀ and TPS₁ presented problems to ill-conditioned scenarios, this was solved using Equation 5.31 criterion to detect and clean those scenarios.

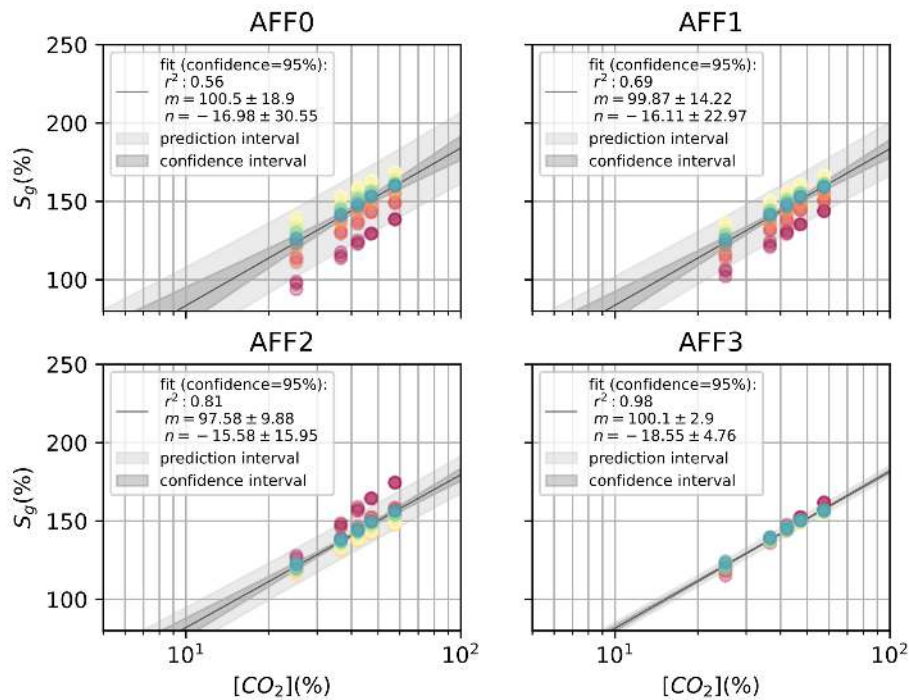


Figure 6.27: AFF₀, AFF₁, AFF₂ and AFF₃ fitted models for the green channel of the measured sensor data. AFF₀ to AFF₂ scored the worst results in the whole dataset. However, AFF₃ scored the best.

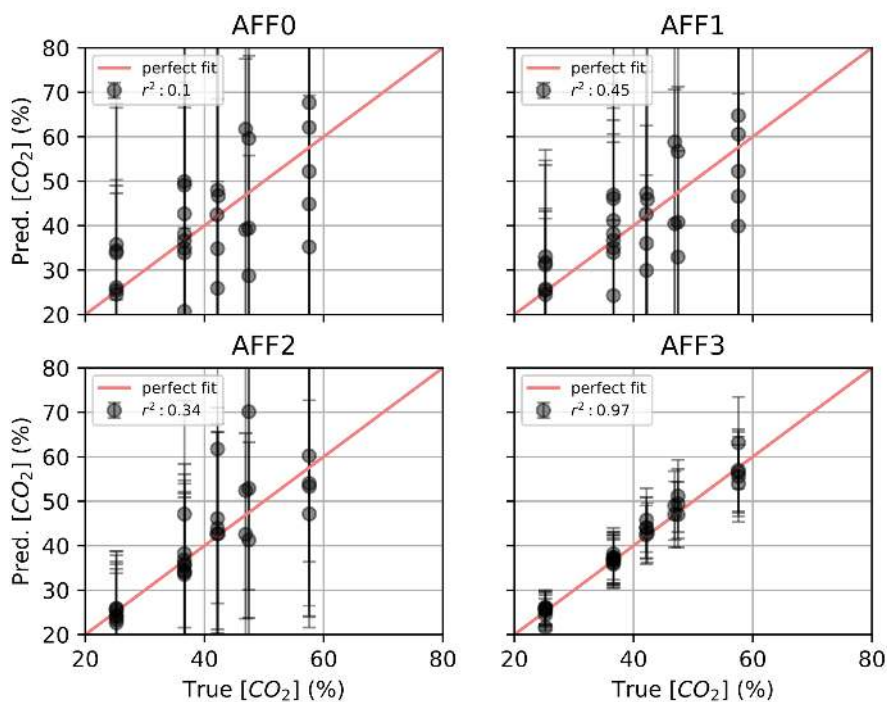


Figure 6.28: AFF₀, AFF₁, AFF₂ and AFF₃ validation regressions. Once again, AFF₀ to AFF₂ present bad results, where their r^2 shows these models are meaningless. On the other hand, AFF₃ presents a really close result to PERF.

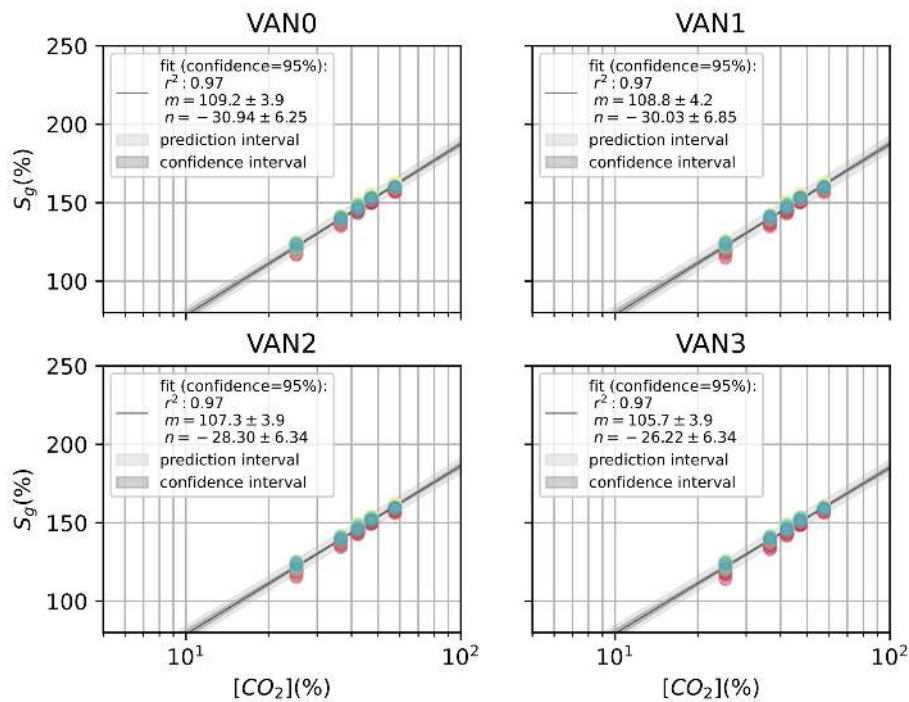


Figure 6.29: VAN0, VAN1, VAN2 and VAN3 fitted models for the green channel of the measured sensor data. All models scored slightly worse metrics than the AFF3 correction, despite this their training results are as good as the AFF3.

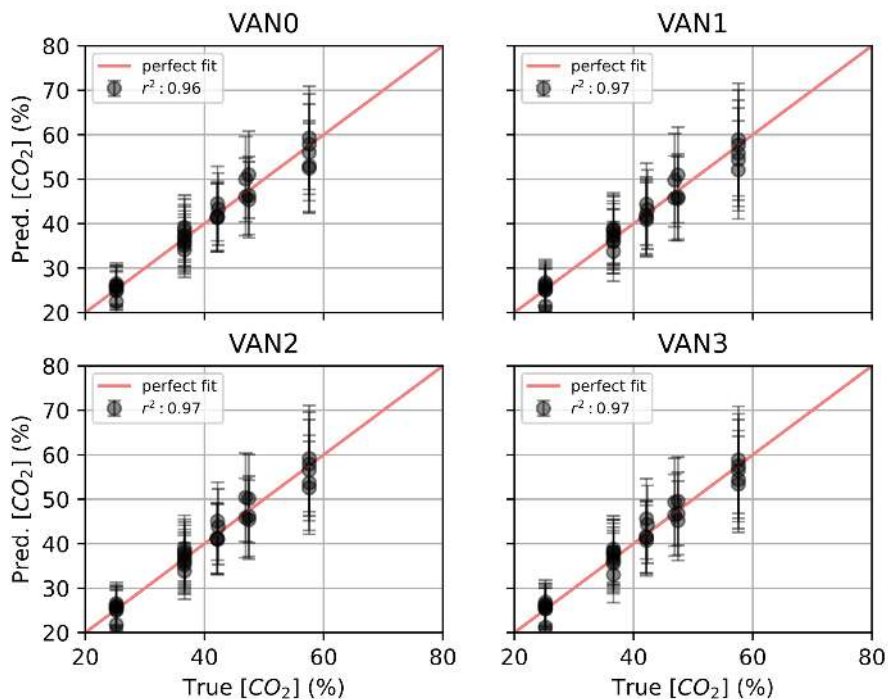


Figure 6.30: VAN0, VAN1, VAN2 and VAN3 validation regressions. All models scored good results ($r^2 > 0.95$) approximating to the PERF results.

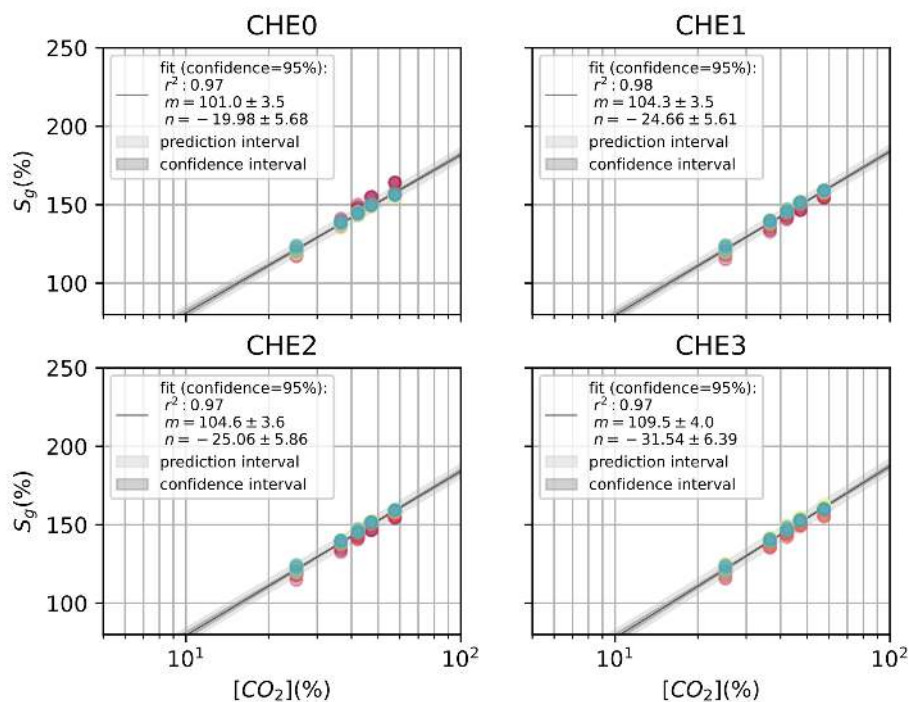


Figure 6.31: CHE0, CHE1, CHE2 and CHE3 fitted models for the green channel of the measured sensor data. All models scored slightly worse metrics than the AFF3 correction, despite this their training results are as good as the AFF3.

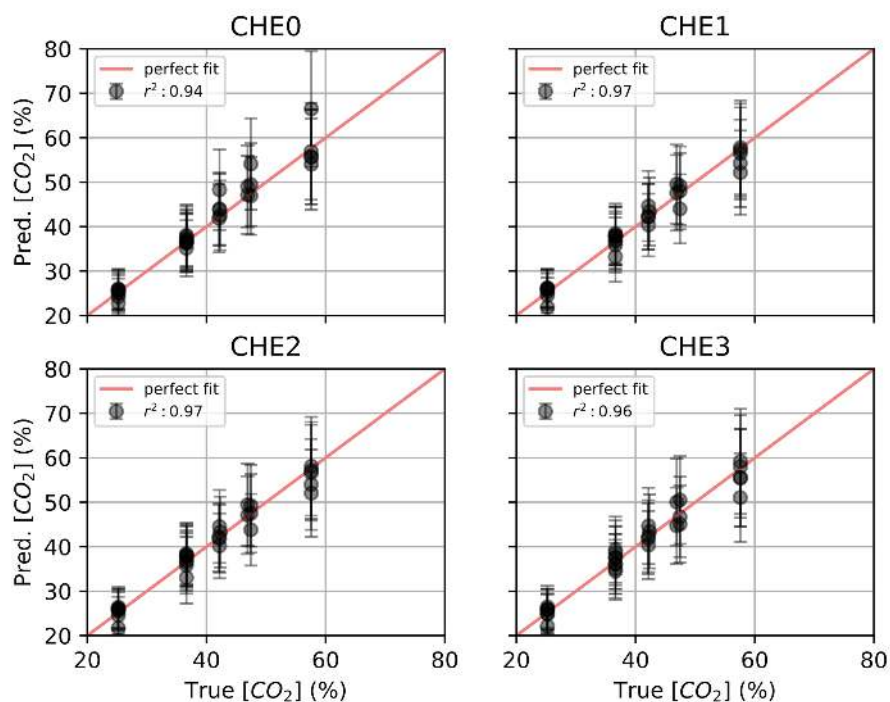


Figure 6.32: CHE0, CHE1, CHE2 and CHE3 validation regressions. All models scored good results ($r^2 > 0.95$), except CHE0 which scored 0.94, approximating to the PERF results.

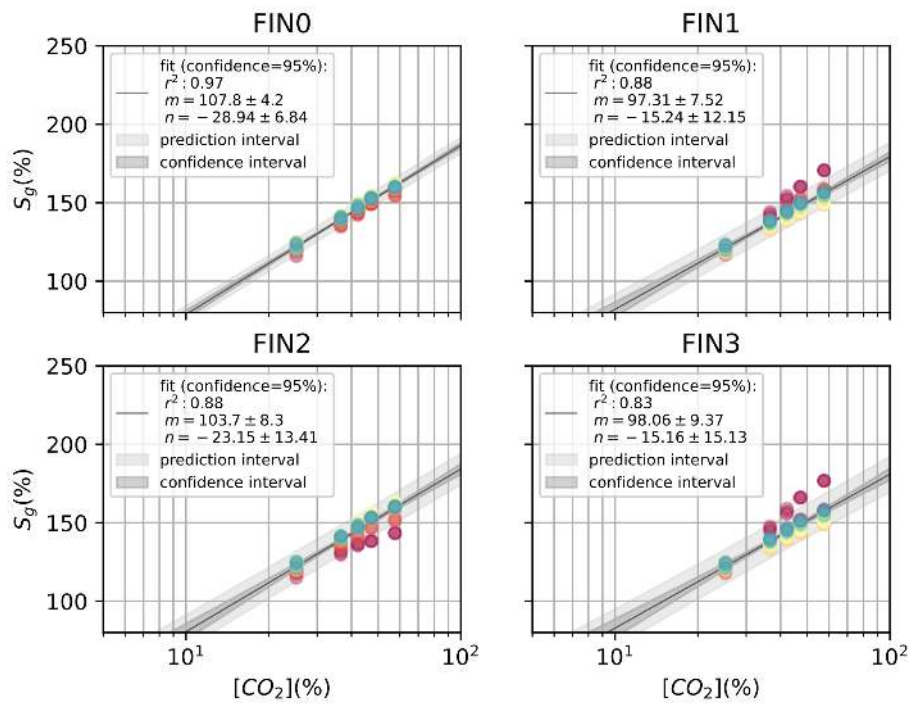


Figure 6.33: FIN0, FIN1, FIN2 and FIN3 fitted models for the green channel of the measured sensor data. Only FIN0 resembles to AFF3, the other three methods performed worse.

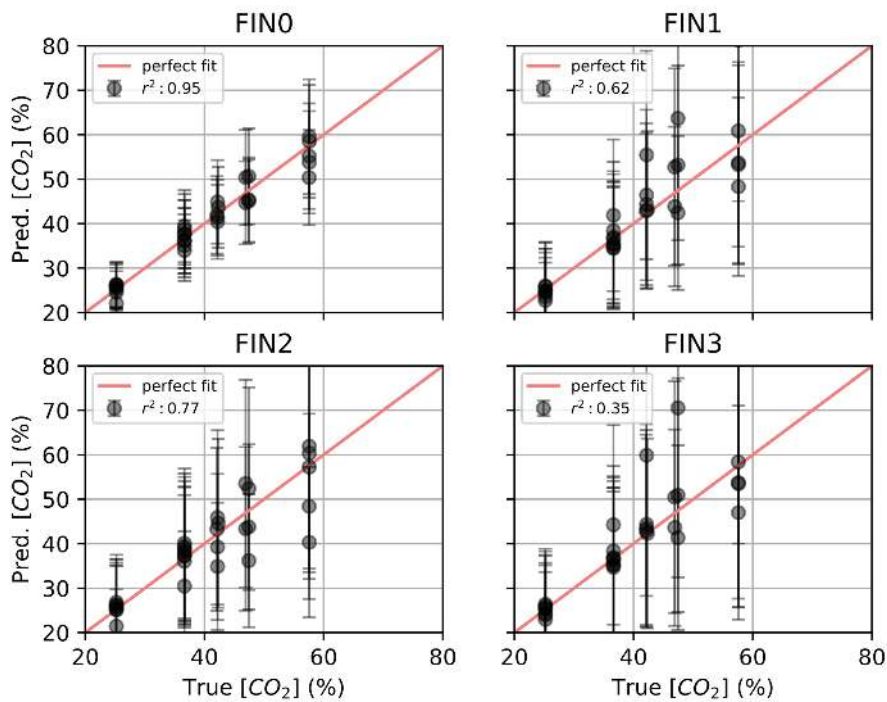


Figure 6.34: FIN0, FIN1, FIN2 and FIN3 validation regressions. Only FIN0 scores good results, compared to AFF3 ($r^2 = 0.95$). The other methods scored worse, resulting in meaningless models.

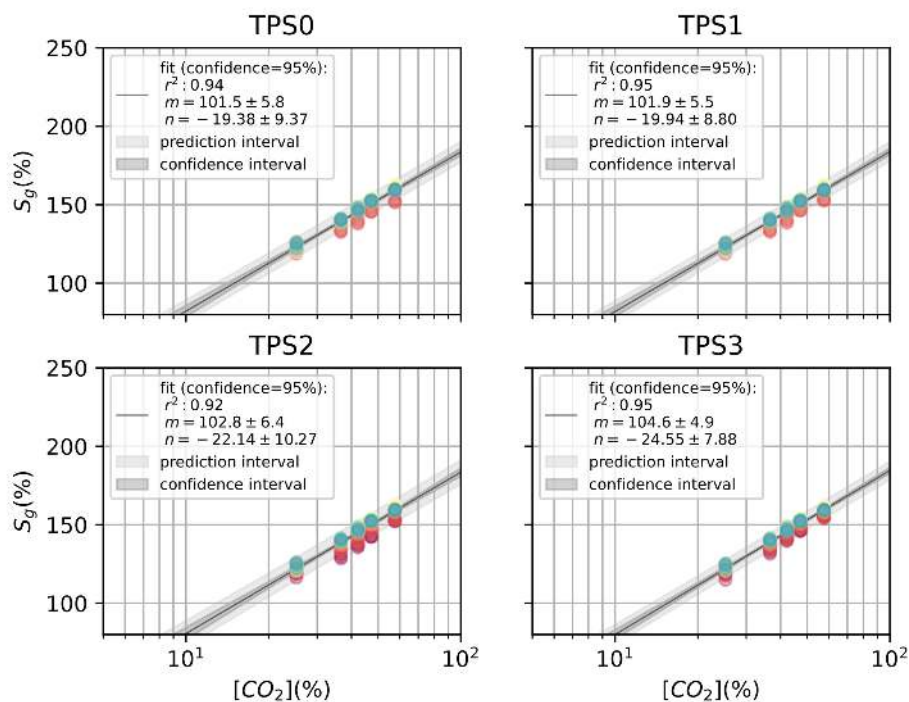


Figure 6.35: TPS0, TPS1, TPS2 and TPS3 fitted models for the green channel of the measured sensor data. All models scored slightly worse metrics than the AFF3 correction, despite this their training results are as good as the AFF3.

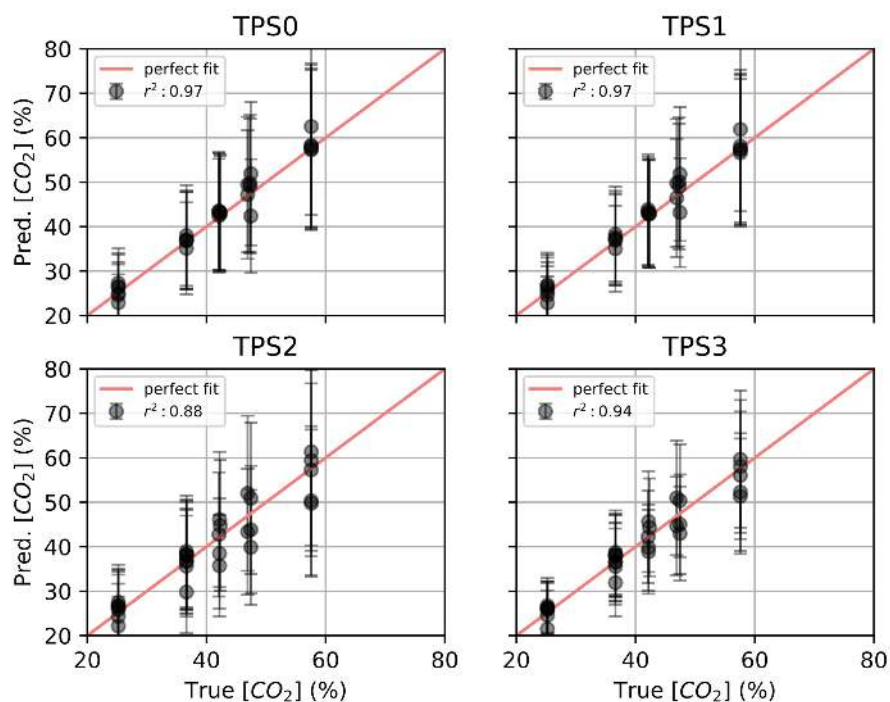


Figure 6.36: TPS0, TPS1, TPS2 and TPS3 validation regressions. TPS0 and TPS1 good results ($r^2 > 0.95$), followed by TPS3 and TPS2.

6.4 Conclusions

In this chapter, we demonstrated the application of our technology to colorimetric indicators. The process to design and acquire the signal of these colorimetric indicators was based upon our proposals of: Color QR Codes –chapter 3 and chapter 4– and a color correction framework to solve the image consistency problem –chapter 5–.

The studied example CO_2 colorimetric indicator [31] presented an excellent response in the green channel of our measured data. Also, the red channel presented a good response, although it was noisy. The blue channel was discarded due to its reduced response.

Then, the studied colorimetric indicator presented good reproducibility and performed linearly with the logarithm of the concentration (for the PERF scenario), as we anticipated in other related work for other colorimetric indicators (such as NH_3 , H_2S , etc.) [3; 29; 30; 159; 163].

Moreover, we tackled the problem of image consistency with our proposed framework. Results indicated that the NONE correction model was useless without applying any color correction. We correctly applied the AFF, VAN, CHE, FIN and TPS corrections. We even detected corner cases of ill-conditioned color corrections in TPS₀ and TPS₁ using the criteria defined in the previous chapter.

Furthermore, AFF₃ outperformed all corrections, which approximated the PERF scenario with the best r^2 scores both in training and validation. This was somehow expected as the problem was biased towards a white-balancing problem, as we used only white light sources from 2500K to 6500K color temperature. Despite this, we demonstrated that AFF₀ or AFF₁, the most common white-balance corrections, were not enough to color correct the data. On one hand, VAN, CHE and TPS followed the results of AFF₃ quite close, remaining in reserve for further analysis in more extreme illumination conditions. On the other hand, FIN presented the worse results (other than AFF₀, AFF₁ and AFF₂), this correlates with chapter 5 conclusions, where we found that the lack of translation components in the color correction, produced poor results for FIN corrections.

All in all, Table 6.2 summarizes all the model results displayed from Figure 6.25 to Figure 6.36. In the table, we also added four additional metrics: the Δc (see Equation 6.2) at 20% and 50% gas concentration and their respective relative metrics, namely the *relative error* ϵ_c of our model in those gas concentrations.

This summary highlights the above-presented evidence. The CO_2 color indicator presented around 10% of relative error in the studied range ($[\text{CO}_2] = 20\% - 50\%$) by itself (PERF). As reference, a commercial CO_2 sensor from Sensirion has a 3% relative error [173].

Then, our result is an excellent result for cost-effective disposable sensors, which are not meant to be persistent like the Sensirion one. Also, without color correction (NONE), it presented a 440% relative error, which is a useless result. Moreover, only correcting with white-balance (AFF₀, AFF₁) scored around 70-90% relative error. Only AFF₃ and related corrections (VAN, CHE) scored good results within 10-20%. TPS methods scored slightly worse results in the range of 20-30% relative error.

Finally, seeking for improving these results, let us discuss some future work for this chapter.

First, in this chapter, we concluded that AFF₃ was the best correction to color correct the presented color indicator for CO_2 sensing. As mentioned, this was probably a biased dataset towards this kind of color deformation. We should look for *more complex illumination configurations* to enhance the sample here presented, we already used those kinds of extreme light configuration in chapter 4 when we created the Color QR Codes.

Second, we could also modify the *camera capturing settings*, this is an interesting topic, as the image consistency problem is not only affected by the light source but also by the camera. Going further, we could perform captures with several devices at the same time. All these new approaches to the problem require more complex setups.

Third, in chapter 5 we concluded we ought to use more locally-bounded color references to specific problems, such the problem of colorimetric indicators. However, when we introduced this chapter we explained that we broaden the amount of encoded colors (from [29] to [30]) instead to keep them to a representative subset of the RGB color spaces that was representative of the problem. Both statements are compatible.

As explained before, we failed to obtain the proper *representative colors* of the problem due to a color reproduction problem, thus we broaden the color chart to a general-purpose 125 RGB colors, to obtain an equidistributed sample of the printer colors. In order to close the loop, as suggested in chapter 5, now that we have more than 24 colors (chapter 5, ColorChecker) we should implement newer color corrections which are based only using those color references that are representative of our data –i.e. the nearest colors–, and seek for an improvement of the results, specially in the TPS color corrections.

Correction	$m \left[\frac{\%}{\%} \right]$	$n[\%]$	$r^2[-]$	$r^2_{\text{valid}}[-]$	$\Delta c_{20}[\%]$	$\Delta c_{50}[\%]$	$\epsilon_{20}[\%]$	$\epsilon_{50}[\%]$
NONE	90 ± 80	200 ± 130	0.04	0.00	88	249	440	497
PERF	98.4 ± 1.8	-14.2 ± 3.0	0.99	0.99	2	5	9	10
AFF0	101 ± 19	-17 ± 31	0.56	0.10	18	51	90	102
AFF1	100 ± 14	-16 ± 23	0.69	0.45	14	38	68	77
AFF2	98 ± 10	-16 ± 16	0.81	0.34	10	27	48	55
AFF3	100.1 ± 2.9	-19 ± 5	0.98	0.97	3	8	14	16
VAN0	109 ± 4	-31 ± 6	0.97	0.96	3	10	17	19
VAN1	109 ± 4	-30 ± 7	0.97	0.97	4	11	19	21
VAN2	107 ± 4	-28 ± 6	0.97	0.97	3	10	17	20
VAN3	106 ± 4	-26 ± 6	0.97	0.97	4	10	18	20
CHE0	101.0 ± 3.5	-20 ± 6	0.97	0.94	3	9	17	19
CHE1	104.3 ± 3.5	-25 ± 6	0.98	0.97	3	9	16	18
CHE2	105 ± 4	-25 ± 6	0.97	0.97	3	9	17	19
CHE3	109 ± 4	-32 ± 6	0.97	0.96	3	10	17	20
FIN0	108 ± 4	-29 ± 7	0.97	0.95	4	11	19	21
FIN1	97 ± 8	-15 ± 12	0.88	0.62	7	21	37	42
FIN2	104 ± 8	-23 ± 13	0.88	0.77	8	22	38	43
FIN3	98 ± 9	-15 ± 15	0.83	0.35	9	26	46	52
TPS0	102 ± 6	-19 ± 9	0.94	0.97	5	15	27	31
TPS1	102 ± 5	-20 ± 9	0.95	0.97	5	14	26	29
TPS2	103 ± 6	-22 ± 10	0.92	0.88	6	17	30	33
TPS3	105 ± 5	-25 ± 8	0.95	0.94	4	13	22	25

Table 6.2: A summary of the presented results. The summary includes metrics for each color correction for 8 different metrics: the first 3 (m, n, r^2) refer to the training model found; r^2_{valid} is the validation score of our models; $\Delta c_{20}[\%]$ and $\Delta c_{50}[\%]$ are the model sensitivity in concentration, with $c = 20\%$ and $c = 50\%$, respectively; $\epsilon_{20}[\%]$ and $\epsilon_{50}[\%]$ are their respective relative error, computed as $100 \cdot \frac{\Delta c}{c}$.

Chapter 7. Conclusions

7.1 Thesis conclusions

This thesis tackled the problem of acquiring data in a quantitative manner from colorimetric indicators, and other colorimetric applications, to do so the problem of automating color calibration ought to be resolved in with a seamless integration to the colorimetric application without any additional barriers to the final consumer, thus using well-known 2D barcodes.

Here, we present a summary of the main conclusions for each one of the thesis objectives:

I Capture machine-readable patterns placed on top of challenging surfaces. Results demonstrated that our method performed better than other extraction methods. We proved so by using the same commercial QR Code reader (ZBar) on the same image which had been corrected by the above-mentioned methods for our three datasets (SYNT, FLAT and SURF), and computing a data readability factor \mathcal{R} for each method and dataset.

For the SYNT and FLAT datasets our method scored similar to the previous methods with almost a $\mathcal{R} = 100\%$, for the SURT dataset –where challenging surfaces were present–, AFF and PRO methods scored really poor results, a 0% and 2%, respectively. CYL method scored a 50%, and TPS up to 79%.

By combining both CYL and TPS methods, we arrived to a joint result of 84%. We even benchmarked this against ZBar without any image correction, this proved our method (TPS+CYL) scored 4 times better than a bare ZBar decoding (84% vs 19%).

II Define a back-compatible QR Code modification to extend QR Codes to act as color charts. Results indicated our method minimized the error applied to a QR Code when color is present, both SNR and BER figures demonstrated that for any of the channels considered. We also demonstrated that the data zone (D) is the more suitable candidate to embed color references, as it presents a higher resilience to be manipulated.

Our method outperformed a random placed of colors by far, for example for a version 5 QR Code, our method outperformed by a 150% the results of the random assignment method for the data zone, and almost a 500% for the global EC&D zone, embedding more than 300 colors in a QR Code.

III Achieve image consistency using color charts for any camera or light setup, enabling colorimetric applications to yield quantitative results. Results proved all TPS methods to be the best methods both in $\overline{\Delta_{RGB_{within}}}$ and $\overline{\Delta_{RGB_{inter}}}$ metrics, scoring half or less the distance of the nearest competitor, the general AFF correction.

Despite this, the original TPS3D method presented a huge number of ill-conditioned cases where the image was not properly corrected, around the 20% - 30% of the cases, this ill-conditioned scenarios were solved when imposing our smoothness proposal.

Also, results indicated that the change in the kernel RBF of the TPS did not improve, neither degrade, the TPS scores.

Moreover, regarding the execution time \mathcal{T} , AFF methods were the fastest methods available of the proposed framework, due to their computational simplicity.

All the other methods scored worse times than these corrections, specially FIN (root-polynomial) and TPS corrections. TPS were 20 to 100 times slower than AFF color corrections. Despite this, we proved that changing the kernel RBF of the TPS formulation did speed up by a 30% the result computation.

IV Demonstrate a specific application of the technology based on colorimetric indicators. Results demonstrated the general affine correction (AFF3) was the best correction in the color correction framework, probably because our experiment was biased towards white-balance corrections.

Our color indicator proved to a good cost-effective indicator with only a 10% relative error in the studied range (PERF), around 10% - 20% when corrected with AFF3 and similar corrections (VAN, CHE), and 20%-30% with TPS corrections. In front of the 440% relative error observed without any correction (NONE).

All in all, we demonstrated the feasibility of applying barcode technology to colorimetric applications, thus enhancing the previous state-of-the-art technologies in the field. Our new Color QR Code acted as substitutes of the traditional color charts, presenting more color capacity in a compact form. Altogether, with a new proposal for color correcting scenes using an improved TPS3D method, we demonstrate the use of our technology to colorimetric indicators.

7.2 Future work

During the presented thesis, we presented some ideas to further continue to work on the presented results in each chapter. How to pursue these partial research was detailed there. Along with this, our integrated solution on how to automate color correction using barcodes can be applied somewhere else. Let us expose some ideas on how to apply our technology beyond colorimetric indicators to other fields where color correction is still an open problem.

First, *other biochemical analytes* can be considered instead of environmental gases, temperature or humidity [2]. Taking as an example water, many authors have proposed colorimetric methods to detect substances in the water: such as *chlorine* [174] or *fluorine* [175], or even, *coliphages* [176].

All these examples, could be integrated straight-forward with our technology as their similarity to colorimetry indicators. Here, the solvent of the substance to sense is liquid (water), which is often mixed with a chemical reactive which contains a derivate of a color indicator. The main gap between our technologies would be a computer vision problem, on how to embed our Color QR Code in their system involving liquid water. Fortunately, in chapter 3 we tackled this problem and proposed a combined method using both TPS and CYL correction which, theoretically, would solve the implementation of our technology on top cylindrical surfaces like reactive vials.

Second, another example is the wide-spread in-vitro diagnostics *lateral-flow assays* [17; 177]. Lateral-flow assays were already popular before 2021, they were popular due to self-diagnosis *pregnancy tests*, that were based on this technology. But nowadays, they are even more popular due to the pandemic situation derived from COVID-19 disease, and the use of this technology to provide to the people of self-diagnosis *antigen tests* for detecting SARS-CoV-2 [178].

Many authors have attempt to perform readouts from lateral-flow assays using smartphones [179]. The most common approach from these authors is to overcome the image consistency problem by fixing the illumination and capture conditions using *ad hoc* hardware to the smartphone [180]. However, those extra hardware present a stopgap between their proposals and the final user, alongside with a price increase to fabricate and distribute the hardware.

Our solution here would overcome those problems, by simply adding a Color QR Code to the lateral-flow cassette, which is a cost-effective solution. Thus, leveraging all the color correction to the smartphone or remote post-processing.

Third, there exists an increasing need for achieving image consistency in other health-care fields, one of these is *dermatology* [33; 34]. Dermatology is a wide health-care field, we can find authors that have used smartphone or neural networks to ease the diagnosis of different diseases like *skin cancer* [181], *skin burns* [182] or other *skin lesions* [183].

Other authors have proposed to use previous color charts to color calibrate dermatology images [184; 185; 186]. For example, Vander-Heaghen et al. presented the use of a ColorChecker chart [13] to achieve consistent imaging in commercial cameras, and concluded that despite their efforts, the resultant images already had too much variability which cannot be eliminated [185].

We could use our technology to improve their results. First, they sought to use the ColorChecker to color correct the images using device-independent color spaces, as we discussed in this thesis, there exist more modern approaches to this problem, working directly in device-depending color spaces. Then, we could apply our color correction framework directly to their dataset. Moreover, our complete proposal of Color QR Codes could add more colors to the color correction that are representatives of the problem tackled. This is similar to the work presented by Cugmas et al. [186] in their teledermoscopy solution for *canine skin*, where they used two ColorChecker charts for this purpose. With our proposal, this seems redundant, the Color QR Codes could embed the colors of both color charts.

Finally, any colorimetric application is potentially approachable by our technology presented in this thesis. The adoption of the technology relies on two further challenges: one, to adapt the color correction to the colorimetric model present in the application, thus conditioning the colors to be embedded in the barcode; and two, to adapt the barcode definition to the desired conditions of the application.

List of Figures

1.1	The GasApp proposal is presented. Left, GasApp changed the core sensing technology from electronic to colorimetric indicators. Right, the initial idea of the GasApp project, a card where colorimetric dyes are printed alongside with color charts and a QR Code.	16
1.2	Simplified 1D representation of the color reproduction problem in reversible and in non-reversible conditions. For clarity only one color coordinate has been represented: x stands for R, G, or B, and x' stands for R', G', or B'. Object colors (x) appear to be different (x') after being acquired by digital means. In some situations, these alterations cannot be removed, because the transformation from x' to x is not single-valued (the critical color ranges where this problem occurs are highly lighted with the green marker).	17
1.3	Four examples of ArUco codes. These codes present certain feature uniqueness (rotation, non-symmetry, etc.), which enables easy location and identification on a scene.	17
1.4	Our thesis proposal to create machine-readable patterns that can accommodate colorimetric sensors and color charts, alongside with the digital information of the QR Code.	18
2.1	The color reproduction problem is represented: (a) a certain light source ($I(\lambda)$) illuminates a certain object with a certain reflectance ($R(\lambda)$), this scene is captured by a certain camera with its sensor response ($D(\lambda)$) and (b) the reproduced image of the object ($R'(\lambda)$) is then illuminated and captured again.	22
2.2	The imaging consistency problem is represented: (a) a certain light source ($I(\lambda)$) illuminates a certain object with a certain reflectance ($R(\lambda)$), this scene is captured by a certain camera with its sensor response ($D(\lambda)$) and (b) the same object is now illuminated by another light source ($I'(\lambda)$) and captured by another camera ($D'(\lambda)$).	23
2.3	A ColorChecker chart. The first row shows a set of six "natural colors"; the second one shows a set of "miscellaneous colors"; the third, primary and secondary colors; and the last row, a gray scale gradient. This set of colors samples the RGB space in a limited way, but it is convenient to carry out a few color corrections manually.	24
2.4	Previous state-of-the-art color correction charts from Pantone and X-Rite. (a) The X-Rite ColorChecker Passport Photo 2® kit. (b) The Pantone Color Match Card®.	25
2.5	Different 2D barcode standards. From left to right: a QR Code, a DataMatrix, an Aztec Code, a MaxiCode, a JAB Code and a HCC Barcode.	26

- 2.6 Block diagram for a general encoding-decoding process of a QR Code which features the embedding of a color layer. This color layer could be used for a wide range of applications, such as placing a brand logo inside a QR Code. The process can be seen as a global encoding process (digital encode and color encode), followed by a channel (print and capture) and a global decoding process (remove colors and decode digital information). 27
- 2.7 Some examples of QR Code versions. From left to right: Micro QR-Code (version M3), version 3 QR Code, and version 10 QR Code. Each of them can store up to 7, 42, 213 bytes, respectively, using a 15% of error correction capacity. 28
- 2.8 Some examples of DataMatrix codes. From left to right: rectangular DataMatrix code, square DataMatrix code and four square DataMatrix combined. Each of them can store up to 14, 28, 202 bytes, respectively, using approximately a 20% of error correction capacity. 28
- 2.9 QR Code encoding defines a complex layout with several patterns to be considered, some of them are non-variant patterns found in each QR Code, others may appear depending on the size of the QR Code, and area related to the data changes for each encoding process. (a) A QR Code with high error correction level and version 5. (b) The complex pattern structure of the pattern. 29
- 2.10 QR Code simplified areas corresponding to the encode process. (a) A QR Code with high error correction level and version 5. (c) Simplified view of the QR patterns, yellow frame corresponds to the "error correction" area and dark green frame corresponds to the "data" area. 30
- 2.11 Different examples of *Halftone QR Codes*, introduced by HK. Chu et al. [56]. These QR Codes exploit the error correction features of the QR Code to achieve back-compatible QR Codes with apparent grayscale –halftone– colors. 31
- 2.12 Original figure from Garateguy et al. [57], different QR Codes with color art are shown: (a) a QR Code with a logo overlaid; (b) a *QArt Code* [58], (c) a *Visual QR Code*; and (d) Garateguy et al. proposal. 31
- 2.13 Computer vision patterns featured in a QR Code. (a) Three finder or position patterns, (b) six alignment patterns, (c) two timing patterns and (d) the fourth corner that can be inferred from the external edges of the finder patterns. 32
- 2.14 Finder pattern definition in terms of modules. Finder pattern measures always 7×7 modules. If scanned with a line barcode scanner the 1:1:3:1:1 ratio is maintained no matter the direction of the scanner. If scanned using contour extraction the aspect ratio $7^2:5^2:3^2$ is maintained as well if the QR Code is captured within a projective scene (i.e. a handheld smartphone). 33
- 2.15 Alignment pattern definition in terms of modules. Alignment pattern measures always 5×5 modules. If scanned with a line barcode scanner the 1:1:1:1:1 ratio is maintained no matter the direction of the scanner. If scanned using contour extraction the aspect ratio $5^2:3^2:1^2$ is maintained as well if the QR Code is captured within a projective scene (i.e. a handheld smartphone). 33
- 2.16 The QR Code contour detection method. a) A QR Code from a certain perspective. b) All the contours detected in the image. c) The location of the position patterns following the area rule. Their respective centers of mass are indicated. 34

2.17	The different orientations of a QR Code are shown. (a) Representation of the slope of the diagonal connecting the corners m and the diagonal segment linked to the top-left corner s . (b) The four possible orientations of a QR-Code.	35
2.18	The QR Code projective correction steps. a) The orientation is deduced from the centers of the 3 finder patterns L , M , N . In this step, their contour corners are found. b) The fourth corner O is found, based on the previous three corners. c) A first projective transformation is carried out, but still subject to significant error shifts around the bottom-right corner; d) The alignment patterns are localized in a restricted contour search. The centers of the alignment patterns (shifted centers after the first projective correction (green) and the reference centers are both found (red). e) The error committed at this stage is shown by subtraction of the images. f) Finally, a second projective transformation recovers the final QR Code image, based on the reference, tabulated, positions of the alignment patterns.	36
2.19	A reduced representation of the reflectance model. For more details see Figure 2.1.	37
2.20	125 colors of an RGB color space. Each channel of the color space has been sampled 5 times. Assuming the space is a 24-bit color space, the values of the sampled colors correspond to: 0, 61, 127, 193 and 255. The combination (255, 255, 255) is the white color and (0, 0, 0) the black color.	38
2.21	An Airy disk is shown as a grayscale image with a color map (top) and as a function (bottom) with the same color map.	41
3.1	An example of an adverse situation, image of a QR Code in a bike-sharing service in Barcelona, where the QR Code is bent over the bike frame. User experience shows that capturing these QR Codes is difficult when approaching the camera to the QR Code due to the bending. (a) An image captured near the QR Code (~ 20 cm), (b) an image captured farther (~ 1 m) and (c) a zoomed version of (b) which despite the blur performs better because the QR Code resembles more to a flat QR Code.	45
3.2	Projection of different surfaces into the capture plane (img) when acquiring images from a digital camera. A QR Code placed on each one of these surfaces will show different deformations(a) an affine (coplanar) plane, (b) a projective (noncoplanar) plane, (c) a cylindrical surface and (d) a thin-plate spline surface, it is continuous and derivable.	47
3.3	Projection of an affine surface into the capture plane (img) when acquiring images from a digital camera.	48
3.4	Projection of a projective surface into the capture plane (img) when acquiring images from a digital camera.	49
3.5	Projection of a cylindrical surface into the capture plane (img) when acquiring images from a digital camera.	49
3.6	Projection of an arbitrary surface into the capture plane (img) when acquiring images from a digital camera.	50
3.7	Example images from the three datasets - (a) SYNT, (b) FLAT and (c) SURF - showing similar QR codes in different surface deformations.	53

3.8	(a) Block diagram for a general encoding-decoding process of a QR Code. (b) A modified diagram with the addition of a deformation due to a noncoplanar surface topography and surface fitting stage which contains a correction steps where image deformation is reverted to improve readout. In our experiments, also, an image augmentation step was added to be used in the proposed experiments for this work.	53
3.9	Two examples (a), (b) from the SYNT dataset. The surfaces were fitted by the four methods described (AFF, PRO, CYL and TPS). The surface fitting is shown as a lattice of red points back-projected onto the original image.	55
3.10	Two examples (a), (b) from the FLAT dataset. The surfaces were fitted by the four methods described (AFF, PRO, CYL and TPS). The surface fitting is shown as a lattice of red points back-projected onto the original image.	56
3.11	Three examples (a), (b), (c) from the SURF dataset. The surfaces were fitted by the four methods described (AFF, PRO, CYL and TPS). The surface fitting is shown as a lattice of red points back-projected onto the original image.	57
3.12	Data readability (\mathcal{R}) of each dataset (SYNT, FLAT, SURF) for each transformation method (AFF, PRO, CYL and TPS).	59
3.13	Data readability (\mathcal{R}) of the SYNT dataset, segregated by the kind of deformation (affine or perspective) that the QR Codes were exposed to, for each transformation method (AFF, PRO, CYL and TPS).	59
3.14	Data readability (\mathcal{R}) of the SURF dataset segregated by the kind of deformation (cylindrical or other) that the QR Codes were exposed to, for each transformation method (AFF, PRO, CYL and TPS).	59
3.15	Data readability (\mathcal{R}) of the three datasets (SYNT, FLAT and SURF) when processed with ZBar and our combined CYL and TPS methods.	60
4.1	A machine-readable pattern to allocate an ammonia sensor. Top: the designed pattern, with two spaces to print a colorimetric sensor. Bottom: the captured version of the pattern with a printed colorimetric dye in one slot. Notice this pattern resembles a QR Code, but it does not contain any data.	63
4.2	Block diagram for a back-compatible encoding-decoding process of a QR Code which features the embedding of a color layer for colorimetric applications. The process can be seen as a global encoding process (digital encode and color encode), followed by a channel (print and capture) and a global decoding process (extract colors and decode digital information). This process is back-compatible with state of the art scanners which remove colors and achieve the decoding of the data and compatible with new decoders which can benefit from color interrogation. The back-compatibility is achieved by following certain rules in the color encoding process (i.e. use the same threshold when placing the colors than when removing them).	64

- 4.3 Previous state-of-the-art QR Code variants that implement colors in some fashion. (a) A QR Code which is able to back-compatible embed an image. (b) A RGB implementation of QR Codes where 3 different QR Codes are packed in each RGB channel, each channel is back-compatible, although the resulting image is not. (c) A High Capacity Color Barcode, a re-implementation of a QR Code standard using colors, which is not back-compatible with QR Codes. 65
- 4.4 A QR Code is overlaid with a logo, which accumulates error due to the presence of the logo. (a) The QR Code is encoded. (b) The code is resized to accommodate the logo. (c) The logo is placed on top of the QR Code. (d) The code is "captured" and down-sampled again. (e) The sampled image is passed to grayscale. (f) The image is binarized, the apparent QR Code differs from the original QR Code (a). 66
- 4.5 A QR Code with a logo is created and read, which accumulates error due to the presence of the logo. (a) The original QR Code encoded. (b) The captured sampled grayscale QR Code. (c) The power difference between (a) and (b). (d) The original grayscale QR Code encoded is binarized, which it is represented exactly as (a). (e) The captured sampled grayscale image from (b) is binarized. (f) The difference between (d) and (e) is shown: light blue pixels correspond to white pixels turned into black by the logo, and dark blue pixels correspond to black pixels turned into white by the logo. 67
- 4.6 The color information from the ColorSensing logo is distributed using different criteria, each one of these distributions compute different measures of SNR and BER, although the total amount of colors is the same, the way they are distributed affects the signal quality. (a) The original QR Code with the logo. (b) The logo colors are sorted at the top of the QR Code. (c) The logo colors are randomly distributed among the QR Code. (d) The logo colors are distributed by using a threshold criterion among blacks and white colors. 72
- 4.7 Histogram comparison between uniform randomly generated RGB channels. (a) which yields to a non-uniform grayscale -L- and uniform randomly generated grayscale -L-. (b) with derived pseudo-uniform RGB channels. 74
- 4.8 The same QR Code is populated with different amounts of colors. (a) 1% of the pixels are substituted using a random placement method (yellow arrows show the colorized pixels). (b) 100% of the pixels are substituted using a random placement method. 74
- 4.9 The same QR Code is populated in different areas with 80% of colors for each area. (a) the whole QR Code is populated (EC&D). (b) Only the error correction area is populated (EC). c. Only the data area is populated. 75
- 4.10 The same QR Code with data and the same amount of colors (80% of the data area) is exposed to different channels. (a) The image passed-through an empty channel. (b) The image passed-through an augmentation channel which resembles a warm light scene. (c) The image passed-through a real environment channel, actually printed and captured in a scene with a lamp at 2500K (warm light). 76

- 4.11 SNR and BER results for Experiment 1 before sending the QR Codes to any channel, only taking into account the QR Codes where all the area has been used (EC&D). Lines and points show average data, light shadows show the min and max values, and heavy shadows show the standard deviation for each color substitution ratio. Left: SNR results for Greyscale (squares, black) and Random (dots, red) methods. Right: BER results for Greyscale (squares, black) and Random (dots, red) methods. 77
- 4.12 SNR and BER results for Experiment 2 after sending the QR Codes to an image augmentation channel, only taking into account the QR Codes where all the area has been used (EC&D). Lines and points show average data, light shadows show the min and max values, and heavy shadows show the standard deviation for each color substitution ratio. Left: SNR results for Greyscale (squares, black) and Random (dots, red) methods. Right: BER results for Greyscale (squares, black) and Random (dots, red) methods. 78
- 4.13 SNR results for Experiment 2, splitted by QR Code version, after sending the QR Codes to an image augmentation channel, only taking into account the QR Codes where all the area has been used (EC&D). SNR results are shown for Greyscale (squares, black) and Random (dots, red) methods. Lines and points show average data, light shadows show the min and max values, and heavy shadows show the standard deviation for each color substitution ratio. 79
- 4.14 SNR and BER results for Experiment 3 after sending the QR Codes to a real channel (printing and capturing the QR Code in a colorimetry setup), only taking into account the QR Codes where all the area has been used (EC&D). Lines and points show average data, light shadows show the min and max values, and heavy shadows show the standard deviation for each color substitution ratio Left: SNR results for Greyscale (squares, black) and Random (dots, red) methods. Right: BER results for Greyscale (squares, black) and Random (dots, red) methods. 80
- 4.15 Success ratio of decoded QR Codes before passing through a channel among different embedding zones (EC&D, Error Correction and Data), for each color mapping method (greyscale and random) for all QR Code versions. Each curve represents a QR Code version, there are up to 5 curves for each method, Greyscale (squares, black) and Random (dots, red). 81
- 4.16 Success ratio of decoded QR Codes after passing through an image augmentation channel among different embedding zones (EC&D, Error Correction and Data), for each color mapping method (greyscale and random) for all QR Code versions. Each curve represents a QR Code version, there are up to 5 curves for each method, Greyscale (squares, black) and Random (dots, red). 82
- 4.17 Success ratio of decoded QR Codes after passing through a real-life channel among different embedding zones (EC&D, Error Correction and Data), for each color mapping method, Greyscale (squares, black) and Random (dots, red), only for a QR Code of version 5. 83
- 4.18 Number of colors that can be embedded in the D zone as a function of the QR Code version (from v3 to v40). Lines show the theoretical maximum number of colors, for different substitution ratios. Square dots show the maximum number of colors that could be embedded in a QR Code with a demonstrated readability above 95% in the conditions of Experiment 2. In contrast to the other QR Code zones, such high readabilities are obtained, even in 100% substitution ratio only in the D zone. 84

4.19	A color QR Code (version 5 with H error correction level) which contains 240 pixels that are coloured. This is implemented with our back-compatible method. These color pixels reproduce the 24 original ColorChecker colors with a redundancy of 10 pixels per color. Only 22% of the digital data pixels are used in this process, almost all the Data (D) zone is used to allocate the colors.	85
5.1	An example of a Gehler's ColorChecker dataset image.	90
5.2	RGB colors of the ColorChecker of an image are projected in the <i>red-green</i> plane. The colors are replicated: (o) show the original colors of the ColorChecker and (x) show their augmented version, as in their captured values.	91
5.3	RGB colors of the ColorChecker of an image are projected in the <i>red-green</i> plane. The colors are replicated: (o) show the original colors of the ColorChecker and (x) show the corrected values of the augmented version shown in Figure 5.2 using a <i>white-balance correction</i> . The whitest point (upper right) is the only one that is properly corrected.	91
5.4	RGB colors of the ColorChecker of an image are projected in the <i>red-green</i> plane. The colors are replicated: (o) show the original colors of the ColorChecker and (x) show the corrected values of the augmented version shown in Figure 5.2 using a <i>white-balance with black-subtraction correction</i> . The whitest point (upper right) and the blackest point (lower left) are the only ones properly corrected.	92
5.5	RGB colors of the ColorChecker of an image are projected in the <i>red-green</i> plane. The colors are replicated: (o) show the original colors of the ColorChecker and (x) show the corrected values of the augmented version shown in Figure 5.2 using an <i>affine correction</i> . We could choose to fix 3 points, but here we applied an approximated solver to the system, so any of the points is strictly matched.	92
5.6	RGB colors of the ColorChecker of an image are projected in the <i>red-green</i> plane. The colors are replicated: (o) show the original colors of the ColorChecker and (x) show the corrected values of the augmented version shown in Figure 5.2 using an <i>affine correction with translation</i> . We could choose to fix 4 points, but here we applied an approximated solver to the system, so any of the points is strictly matched.	93
5.7	RGB colors of the ColorChecker of an image are projected in the <i>red-green</i> plane. The colors are replicated: (o) show the original colors of the ColorChecker and (x) show the corrected values of the augmented version shown in Figure 5.2 using a <i>geometric polynomial correction of degree 4</i> . Many of the points are almost matched due to the polynomial expansion.	94
5.8	RGB colors of the ColorChecker of an image are projected in the <i>red-green</i> plane. The colors are replicated: (o) show the original colors of the ColorChecker and (x) show the corrected values of the augmented version shown in Figure 5.2 using a <i>thin-plate spline correction</i> . All the points are strictly matched by the TPS definition.	95
5.9	RGB colors of the ColorChecker of an image are projected in the <i>red-green</i> plane. The colors are replicated: (o) show the original colors of the ColorChecker and (x) show the corrected values of the augmented version shown in Figure 5.2 using a <i>smoothed thin-plate spline correction</i> . Not all the points are strictly matched now, as we relaxed the the TPS definition.	97

- 5.10 Our pipeline: for each Gehler's dataset raw image (bayer) we develop an RGB image, which is already the half size of the original image, also this image is down-sampled to reduce its size 4 times. Then we augment this down-sampled image with 100 sample augmentation scenarios. For each augmented scenario we correct back before augmentation using 21 different correction methods described in Table 5.1. 100
- 5.11 An image from Gehler's dataset ($K=1$) is down-sampled with 3 factors ($K=4, 16, 64$), where K is the down-sampling factor. The figure also shows the histogram associated with each image and the size in pixels of the image. Down-sampled images by a factor 4 maintain the histogram representation, but further down-sampling alters the histogram. 102
- 5.12 Different examples of color augmentation using `imgaug` in Python. The upper-left image is the developed original image from the Gehler's dataset. The other images are augmentations of these image with variations in color, contrast and saturation. 102
- 5.13 The metric $\overline{\Delta_{RGB,within}}$ is represented. RGB colors of the ColorChecker of an image are projected in the *red-green* plane. The colors are present as their ground-truth value (\circ) and their augmented copy (\times). Dashed lines across the plane show the $\Delta_{RGB,within}$ between each color pair. Cyan, magenta and yellow pairs are highlighted above the other ColorChecker colors. 103
- 5.14 The set $\Delta_{RGB,pairwise}$ is represented. RGB colors of the ColorChecker of an image are projected in the *red-green* plane. The colors are present as their ground-truth value (\circ). Dashed lines across the plane show the $\Delta_{RGB,pairwise}$ between all the colors. The distances between cyan, magenta and yellow are highlighted above the other distances. 103
- 5.15 The metric $\overline{\Delta_{RGB,inter}}$ is represented. RGB colors of an entire image are projected in the *red-green* plane. The colors are present as their ground-truth value (black points, \circ) and their augmented copy (red points, \times). Then, three random colors are selected to show dashed lines across the plane to show the $\Delta_{RGB,inter}$ between each color pair. 104
- 5.16 An example of a failed and ill-conditioned correction. The augmented image shows saturated colors: the yellowish colors and the whitish colors. The corrected image is computed with the TPS method rendering an erroneous result. 104
- 5.17 A count of the failed corrections for each correction method is shown. Failed corrections are selected if their $\overline{\Delta_{RGB,within}}$ computation is greater than the NONE correction. After this, the count is divided in ill-conditioned results or not. Ill-condition is assessed using the $\Delta_{RGB,pairwise}$ comparison to a minimum distance of $\Delta_{RGB} = \sqrt{3}$ 105
- 5.18 The $\overline{\Delta_{RGB,within}}$ for each image in the dataset and for each augmentation is shown as a distribution against the color correction techniques. The means of the distributions are also present (Δ). PERF correction is not zero and shows the quantization effect. NONE is a reference of not applying any correction at all. The rest of the corrections are grouped in: AFF, VAN, CHE, FIN and TPS corrections. 107
- 5.19 The $\overline{\Delta_{RGB,inter}}$ for each image in the dataset and for each augmentation is shown as a distribution against the color correction techniques. The means of the distributions are also present (Δ). PERF correction is not zero and shows the quantization effect. NONE is a reference of not applying any correction at all. The rest of the corrections are grouped in: AFF, VAN, CHE, FIN and TPS corrections. 108

5.20	The execution time in milliseconds for each image in the dataset and for each augmentation is shown as a distribution against the color correction techniques. The means of the distributions are also present (Δ).	110
5.21	The execution time in milliseconds against the image size for a reduced set of images of the dataset. Results show a linear behavior for all the corrections techniques. Corrections are grouped by color and marker, within the same group different transparencies have been applied to differentiate the corrections.	111
6.1	Reaction mechanism of the pH indicator bromocresol green (BCG, pH 3.8–5.4) for the detection of NH_3 . Increase of the NH_3 concentration leads to a proton release, detectable as a color change from yellow over green to blue.	115
6.2	UV–VIS diffuse reflectance of the soaked pads with Griess-Saltzman reagent exposed to different NO_2 concentrations and the corresponding images of the colors developed (insets, 3 replicas per concentration).	116
6.3	Left, an ammonia (NH_3) colorimetric indicator has been dip-coated into a glass substrate, which exhibits a yellow color when exposed to synthetic air. Right, the same sensor is exposed to 100 ppm of NH_3 and it turns into purple.	116
6.4	(a) Standard tristimulus $X(\lambda)$, $Y(\lambda)$, $Z(\lambda)$ curves of the human eye. (b) The integrated sRGB colors represented in the RGB cube. (c) The rendered sequence of RGB colors corresponding to the gas sensing spectra $c(\lambda)$.	117
6.5	The proposed pipeline for creating machine-readable patterns proposed in 2018 [29].	118
6.6	A machine-readable pattern to allocate an ammonia sensor. Left: the designed pattern, with two spaces to print a colorimetric sensor. Right: the captured version of the pattern with a printed colorimetric dye in one slot. Notice this pattern resembles a QR Code, but it does not contain any data.	118
6.7	RGB 8-bit color data acquired from a colorimetric sensor captured with a digital camera at 5500K color temperature exposition, with the centers of 32 clusters generated by K-means clustering. Data is presented as a projection into the red-channel plane of the RGB space.	119
6.8	32 clusters centers from Figure 6.7 data, and color clustering regions. Data is presented as a projection into the red-channel plane of the RGB space.	119
6.9	The layer structure of the machine-readable pattern for colorimetric indicators: a) the colorimetric indicator ink, b) the machine-readable pattern inks, c) the plastic substrate and d) white cardboard.	120
6.10	Five machine-readable patterns (a), (b), (c), (d) and (e) are exposed to different atmospheres (1), (2), (3), (4), (5), the value of the mean measured RGB color for each ink at each atmosphere is represented as a transposed vector. (a) a NH_3 sensor using the BPB and BGC indicators. (b) a CH_2O dosimeter using the BGC indicator. (c) a H_2S dosimeter using the Cu-PAN. (d) a CH_2O dosimeter using the BTB+ODA indicator. And, (e) a CH_2O dosimeter using the BCP+ODA indicator. The different 5 atmospheric conditions can be consulted in Engel et al. [30].	121

- 6.11 A back-compatible Color QR Code for colorimetric indicators. This QR Code will be read by commercial scanners, and it should display the URL: `c-s.is/#38RmtGVV6RQ5f`. The Color QR Code includes up to 125 reference colors and blank space to allocate a colorimetric indicator ink (above the lower finder pattern). 122
- 6.12 The structure of the Color QR Code from Figure 6.11 is detailed. a) and b) show possible sensor inks placements, a) shows a big sensor outside the QR Code, b) shows smaller factor forms (3×2 , 1×1 , ...) inside the QR Code. c) Shows the color references and how they are spread in the QR Code areas. Finally, d) shows the whole layout of the sensor with the Color QR Code. 122
- 6.13 16 different Color QR Codes for colorimetric indicators with different encoded data that differs in an alphanumeric ID. The encoded reference colors in each QR Code is the same, however the position of the colors is distributed following the digital data in a back-compatible manner. Each Color QR Code has a reserved area (white) above the lower finder pattern to allocate a colorimetric ink. 123
- 6.14 Two screens and one substrate sheet. Each screen can print one color indicator, and both can be combined into the same pattern. The substrate has DINA4 measures, it also contains up to 10 Color QR Codes with an approximated size of 1 inch. 124
- 6.15 Several substrate sheets already printed, each sheet contains up to 10 CO_2 sensors and 10 NH_3 sensors. 124
- 6.16 Schema of our laboratory setup. The setup features 3 subsystems: a massflow controller station, a capture station and a user-access computer. The massflow controller station provides modified atmospheres to a chamber where the gas sensors are placed. The capture station can see the chamber through an optical window, and take time-lapses with a controlled light setting. Finally, the user computer presents a web page interface to the user to operate the system. 125
- 6.17 The 3D design of the circular sensor chamber. The chamber is transparent to enable optical readings, and it is sealed using rubber (orange). The chamber also has four threaded input/output holes. 125
- 6.18 The expected (black) and the measured gas concentration (red) for each gas pulse is shown on a temporal axis along the experiment duration. The measured values were taken from the BROOKS instrumentation reading while applying a correction algorithm provided by the manufacturer [167]. 127
- 6.19 A printed sensor, featuring a Color QR Code and two colorimetric indicators is displayed inside the sensor chamber of our setup. Then is exposed to different light conditions. From left to right, the illumination changes following 3 color temperatures of white light: 2500K, 4500K and 6500K. 127
- 6.20 From top to bottom: a representation of the color of the sensor over time for the D65 standard illuminant (6500K), it can be observed it changes from blueish colors to yellowish colors; the same colors as RGB channel signals; and the response (%) for all the RGB channels. 129
- 6.21 The response of the green channel exposed to a D65 standard illuminant (6500K) for all the pulses are overlapped in the same time frame. This results in 15 pulses, each reference target gas concentration (20, 30, 35, 40, 50) has a pulse replica (0, 1, 2). 130

6.22 The response of the green channel exposed to nine illuminants (2500K to 6500K) for all the pulses are overlapped in the same time frame. This results in 135 apparent pulses, now each reference target gas concentration (20, 30, 35, 40, 50) has a pulse replica (0, 1, 2) for each illuminant. Legend is omitted to favor clearness, results should be compared to Figure 6.21. 131

6.23 The captured color references from the Color QR Code in each illumination condition. D65 is the reference space. 131

6.24 The response of the green channel exposed to nine illuminants (2500K to 6500K), then corrected using the TPS3 method (Table 5.1), for all the pulses are overlapped in the same time frame. This results in 135 apparent pulses, now each reference target gas concentration (20, 30, 35, 40, 50) has a pulse replica (0, 1, 2) for each illuminant. Legend is omitted to favor clearness, results should be compared to Figure 6.21. The shadowed area is the area corresponding to the 5 minutes windows used to integrate the response of the sensor for the model computation. 132

6.25 NONE and PERF fitted models, which represent the worst and the best case scenarios, respectively. The fitted model in NONE is the one performed without correcting any captured color. The PERF model is an artificial model in which each captured color has been mapped to its correspondent D65 color. 133

6.26 NONE and PERF regression for the validation data. The coefficient r^2 was computed for this data. This result confirms that NONE is the worst case scenario with a null r^2 , and PERF is the best score in the whole set of results. 133

6.27 AFF0, AFF1, AFF2 and AFF3 fitted models for the green channel of the measured sensor data. AFF0 to AFF2 scored the worst results in the whole dataset. However, AFF3 scored the best. 135

6.28 AFF0, AFF1, AFF2 and AFF3 validation regressions. Once again, AFF0 to AFF2 present bad results, where their r^2 shows these models are meaningless. On the other hand, AFF3 presents a really close result to PERF. 135

6.29 VAN0, VAN1, VAN2 and VAN3 fitted models for the green channel of the measured sensor data. All models scored slightly worse metrics than the AFF3 correction, despite this their training results are as good as the AFF3. 136

6.30 VAN0, VAN1, VAN2 and VAN3 validation regressions. All models scored good results ($r^2 > 0.95$) approximating to the PERF results. 136

6.31 CHE0, CHE1, CHE2 and CHE3 fitted models for the green channel of the measured sensor data. All models scored slightly worse metrics than the AFF3 correction, despite this their training results are as good as the AFF3. 137

6.32 CHE0, CHE1, CHE2 and CHE3 validation regressions. All models scored good results ($r^2 > 0.95$), except CHE0 which scored 0.94, approximating to the PERF results. 137

6.33 FIN0, FIN1, FIN2 and FIN3 fitted models for the green channel of the measured sensor data. Only FIN0 resembles to AFF3, the other three methods performed worse. 138

6.34 FIN0, FIN1, FIN2 and FIN3 validation regressions. Only FIN0 scores good results, compared to AFF3 ($r^2 = 0.95$). The other methods scored worse, resulting in meaningless models. . . . 138

6.35 TPS₀, TPS₁, TPS₂ and TPS₃ fitted models for the green channel of the measured sensor data. All models scored slightly worse metrics than the AFF₃ correction, despite this their training results are as good as the AFF₃. 139

6.36 TPS₀, TPS₁, TPS₂ and TPS₃ validation regressions. TPS₀ and TPS₁ good results ($r^2 > 0.95$), followed by TPS₃ and TPS₂. 139

List of Tables

2.1	A summary of QR Code data encoding capacity is shown. The total capacity for each configuration is expressed in <i>symbol capacity</i> . Columns are ordered left to right from higher to lower capacity.	30
3.1	Summary of dataset sizes. All datasets attempt to have the same size employing QR Code generation, different captures or image augmentation.	54
4.1	The values for the SNR and BER are computed for the QR Code with a logo from Figure 4.4. The SNR is computed using grayscale images. The BER is computed using binary images (see Figure 4.4).	68
4.2	Values of SNR and BER computed for each criteria in Figure 4.6. Using the logo as it is, the sorted criteria and random criteria yield to similar results. However, the use of a simple grayscale threshold criteria slightly increases the SNR and hugely depletes the BER, showing a good result for encoding colors in a back-compatible way.	72
4.3	Summary of parameter values for each experiment designed. All experiments share common parameters, at least each experiment has 72 different QR Codes that will be generated using as reference the multiplication of the shared parameters. Experiment 1 generates 360.000 different QR Codes.	73
4.4	Number of different colors that can be embedded inside a QR Code with a 95% success ratio during the decoding process for each insertion mask (EC&D, EC or D), for both color mapping methods (greyscale and random). In absolute terms, the mask corresponding with only the Data zone beats the other two, as expected the Grayscale method performs better than the Random one.	84
4.5	Properties of the proposed QR Code with the ColorChecker colors embedded in it. Properties are related with different steps in the QR Code life-cycle, from encoding to decoding. . . .	85
5.1	All the color corrections performed in this work. The table shows the name of the correction, the tag used in this work to refer to the correction and the augmented definition for each vector of P , the color references or color to be corrected. In this table we use a reduced notation $\Delta_i = \Delta_{RGB}(\mathbf{s}_i, \mathbf{c})$ for simplicity.	99
5.2	Sizes in pixels (x, y) of the images along our pipeline. Notice raw pixels are natural pixels of the sensor, this means each pixel only represents one color (red, green or blue).	101

- 5.3 A summary of the presented results. The summary includes metrics for each color correction for 7 different metrics (see left), the within-distances and inter-distances also include some statistical information such as the mean (μ), the standard deviation (σ) and the median ($\tilde{\mu}$). The median should be considered as the reference figure in those metrics as their distributions are quite asymmetric. 114
- 6.1 The expected and the measured gas concentration for each gas pulse is shown. The measured values were taken from the BROOKS instrumentation reading while applying a correction algorithm provided by the manufacturer [167]. 126
- 6.2 A summary of the presented results. The summary includes metrics for each color correction for 8 different metrics: the first 3 (m, n, r^2) refer to the training model found; r_{valid}^2 is the validation score of our models; $\Delta c_{20}[\%]$ and $\Delta c_{50}[\%]$ are the model sensitivity in concentration, with $c = 20\%$ and $c = 50\%$, respectively; $\epsilon_{20}[\%]$ and $\epsilon_{50}[\%]$ are their respective relative error, computed as $100 \cdot \frac{\Delta c}{c}$ 142

Bibliography

- [1] Jose C. Contreras-Naranjo, Qingshan Wei, and Aydogan Ozcan. Mobile phone-based microscopy, sensing, and diagnostics. *IEEE Journal of Selected Topics in Quantum Electronics*, 22(3):1–14, May 2016. DOI: 10.1109/jstqe.2015.2478657. URL <https://doi.org/10.1109/jstqe.2015.2478657>.
- [2] Ajay Piriya V.S, Printo Joseph, Kiruba Daniel S.C.G., Susithra Lakshmanan, Takatoshi Kinoshita, and Sivakumar Muthusamy. Colorimetric sensors for rapid detection of various analytes. *Materials Science and Engineering: C*, 78:1231–1245, September 2017. DOI: 10.1016/j.msec.2017.05.018. URL <https://doi.org/10.1016/j.msec.2017.05.018>.
- [3] Cristian Fàbrega, Luis Fernández, Oriol Monereo, Alba Pons-Balagué, Elena Xuriguera, Olga Casals, Andreas Waag, and Joan Daniel Prades. Highly specific and wide range NO₂ sensor with color readout. *ACS Sensors*, 2(11):1612–1618, October 2017. DOI: 10.1021/acssensors.7b00463. URL <https://doi.org/10.1021/acssensors.7b00463>.
- [4] Gabriel Martins Fernandes, Weida R. Silva, Diandra Nunes Barreto, Rafaela S. Lamarca, Paulo Clairmont F. Lima Gomes, João Flávio da S Petrucci, and Alex D. Batista. Novel approaches for colorimetric measurements in analytical chemistry – a review. *Analytica Chimica Acta*, 1135:187–203, October 2020. DOI: 10.1016/j.aca.2020.07.030. URL <https://doi.org/10.1016/j.aca.2020.07.030>.
- [5] Bettersense – nanodevice engineering for a better chemical gas sensing technology. <http://www.bettersense.eu/default.asp>, 2014-2019.
- [6] Gasapp – making complex gas analytics friendly and available asap. <https://cordis.europa.eu/project/id/727297>, 2017-2018.
- [7] Snapgas – a smartphone-based dosimeter of the exposure to toxic gases. <http://snap-gas.eu/>, 2018-2020.
- [8] Ana Moya, Gemma Gabriel, Rosa Villa, and F. Javier del Campo. Inkjet-printed electrochemical sensors. *Current Opinion in Electrochemistry*, 3(1):29–39, June 2017. DOI: 10.1016/j.coelec.2017.05.003. URL <https://doi.org/10.1016/j.coelec.2017.05.003>.
- [9] Ahmed Salim and Sungjoon Lim. Review of recent inkjet-printed capacitive tactile sensors. *Sensors*, 17(11):2593, November 2017. DOI: 10.3390/s17112593. URL <https://doi.org/10.3390/s17112593>.
- [10] R.H. Leach, R. Leach, and R. Pierce. *The Printing Ink Manual*. Springer, 1993. ISBN 9780948905810. URL <https://books.google.es/books?id=2PwKTq05dioC>.
- [11] R.W.G. Hunt. The reproduction of colour. *Color Research & Application*, 30(6):466–467, 2005. ISSN 0361-2317. DOI: 10.1002/col.20163.
- [12] Mahmoud Afifi, Brian Price, Scott Cohen, and Michael S. Brown. When color constancy goes wrong: Correcting improperly white-balanced images. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2019. DOI: 10.1109/cvpr.2019.00163. URL <https://doi.org/10.1109/cvpr.2019.00163>.

- [13] C. S. McCamy, H. Marcus, and J. G. Davidson. COLOR-RENDITION CHART. *J Appl Photogr Eng*, 2(3):95–99, 1976.
- [14] G.D. Finlayson, S.D. Hordley, and R. Xu. Convex programming colour constancy with a diagonal-offset model. In *IEEE International Conference on Image Processing 2005*. IEEE, 2005. DOI: 10.1109/icip.2005.1530550. URL <https://doi.org/10.1109/icip.2005.1530550>.
- [15] Paolo Menesatti, Claudio Angelini, Federico Pallottino, Francesca Antonucci, Jacopo Aguzzi, and Corrado Costa. RGB color calibration for quantitative image analysis: The "3D Thin-Plate Spline" warping approach. *Sensors (Switzerland)*, 12(6):7063–7079, 2012. ISSN 14248220. DOI: 10.3390/s120607063.
- [16] Kenneth D. Long, Elizabeth V. Woodburn, Huy M. Le, Utsav K. Shah, Steven S. Lumetta, and Brian T. Cunningham. Multimode smartphone biosensing: the transmission, reflection, and intensity spectral (TRI)-analyzer. *Lab on a Chip*, 17(19):3246–3257, 2017. DOI: 10.1039/c7lc00633k. URL <https://doi.org/10.1039/c7lc00633k>.
- [17] Joonchul Shin, Sudesna Chakravarty, Wooseok Choi, Kyungyeon Lee, Dongsik Han, Hyundoo Hwang, Jaekyu Choi, and Hyo-Il Jung. Mobile diagnostics: next-generation technologies for in vitro diagnostics. *The Analyst*, 143(7):1515–1525, 2018. DOI: 10.1039/c7an01945a. URL <https://doi.org/10.1039/c7an01945a>.
- [18] Rafael C. Gonzalez, Richard E. Woods, and Steven L. Eddins. *Digital image processing using MATLAB*. Tata McGraw Hill Education, 2. ed., 4. repr edition, 2011. ISBN 9780070702622.
- [19] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and R. Medina-Carnicer. Generation of fiducial marker dictionaries using Mixed Integer Linear Programming. *Pattern Recognition*, 51:481–491, 2016. ISSN 00313203. DOI: 10.1016/j.patcog.2015.09.023.
- [20] ISO Central Secretary. Information technology - automatic identification and data capture techniques - qr code bar code symbology specification. ISO ISO/IEC 18004:2015, International Organization for Standardization, 2015. URL <https://www.iso.org/standard/62021.html>.
- [21] Yuan Xu, Zhangming Liu, Rui Liu, Mengxue Luo, Qi Wang, Liqin Cao, and Shuangli Ye. Inkjet-printed pH-sensitive QR code labels for real-time food freshness monitoring. *Journal of Materials Science*, 56(33):18453–18462, September 2021. DOI: 10.1007/s10853-021-06477-x. URL <https://doi.org/10.1007/s10853-021-06477-x>.
- [22] João F.C.B. Ramalho, L.C.F. António, S.F.H. Correia, L.S. Fu, A.S. Pinho, C.D.S. Brites, L.D. Carlos, P.S. André, and R.A.S. Ferreira. [INVITED] luminescent QR codes for smart labelling and sensing. *Optics & Laser Technology*, 101:304–311, May 2018. DOI: 10.1016/j.optlastec.2017.11.023. URL <https://doi.org/10.1016/j.optlastec.2017.11.023>.
- [23] Ismael Benito Altamirano, Olga Casals Guillen, Cristian Fàbrega Gallego, Juan Daniel Prades García, Andreas Hans Wilhelm Waag. Colour correction, August 2019. URL <https://patents.google.com/patent/W02019145390A1/>.
- [24] Colorsensing – color imaging revolution. <http://color-sensing.com/>, 2018.
- [25] Laslo Tarjan, Ivana Šenk, Srdjan Tegeltija, Stevan Stankovski, and Gordana Ostojic. A readability analysis for qr code application in a traceability system. *Computers and Electronics in Agriculture*, 109:1–11, 2014. ISSN 0168-1699. DOI: <https://doi.org/10.1016/j.compag.2014.08.015>. URL <https://www.sciencedirect.com/science/article/pii/S0168169914002142>.
- [26] Jianping Qian, Bin Xing, Baohui Zhang, and Han Yang. Optimizing QR code readability for curved agro-food packages using response surface methodology to improve mobile phone-based traceability. *Food Packaging and Shelf Life*, 28:100638, June 2021. DOI: 10.1016/j.fpsl.2021.100638. URL <https://doi.org/10.1016/j.fpsl.2021.100638>.

- [27] Vien Cheung, Stephen Westland, David Connah, and Caterina Ripamonti. A comparative study of the characterisation of colour cameras by means of neural networks and polynomial transforms. *Coloration Technology*, 120(1):19–25, 2004. ISSN 14723581. DOI: 10.1111/j.1478-4408.2004.tb00201.x.
- [28] Graham D. Finlayson, Michal MacKiewicz, and Anya Hurlbert. Color Correction Using Root-Polynomial Regression. *IEEE Transactions on Image Processing*, 24(5):1460–1470, 2015. ISSN 10577149. DOI: 10.1109/TIP.2015.2405336.
- [29] Ismael Benito-Altamirano, Peter Pfeiffer, Oriol Cusola, and J. Daniel Prades. Machine-Readable Pattern for Colorimetric Sensor Interrogation. *Proceedings*, 2(13):906, 2018. ISSN 2504-3900. DOI: 10.3390/proceedings2130906.
- [30] Laura Engel, Ismael Benito-Altamirano, Karina R. Tarantik, Carolin Pannek, Martin Dold, J. Daniel Prades, and Jürgen Wöllenstein. Printed sensor labels for colorimetric detection of ammonia, formaldehyde and hydrogen sulfide from the ambient air. *Sensors and Actuators, B: Chemical*, 330 (December 2020), 2021. ISSN 09254005. DOI: 10.1016/j.snb.2020.129281.
- [31] Yanan Zhang and Loong-Tak Lim. Inkjet-printed CO₂ colorimetric indicators. *Talanta*, 161:105–113, December 2016. DOI: 10.1016/j.talanta.2016.08.014. URL <https://doi.org/10.1016/j.talanta.2016.08.014>.
- [32] Ivor J Church and Anthony L Parsons. Modified atmosphere packaging technology: A review. *Journal of the Science of Food and Agriculture*, 67(2):143–152, February 1995. DOI: 10.1002/jsfa.2740670202. URL <https://doi.org/10.1002/jsfa.2740670202>.
- [33] Newton M. Kinyanjui, Timothy Odonga, Celia Cintas, Noel C. F. Codella, Rameswar Panda, Prasanna Sattigeri, and Kush R. Varshney. Fairness of classifiers across skin tones in dermatology. In Anne L. Martel, Purang Abolmaesumi, Danail Stoyanov, Diana Mateus, Maria A. Zuluaga, S. Kevin Zhou, Daniel Racoceanu, and Leo Joskowicz, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*, pages 320–329, Cham, 2020. Springer International Publishing. ISBN 978-3-030-59725-2.
- [34] Kerstin Bunte, Michael Biehl, Marcel F. Jonkman, and Nicolai Petkov. Learning effective color features for content based image retrieval in dermatology. *Pattern Recognition*, 44(9):1892–1902, 2011. ISSN 0031-3203. DOI: <https://doi.org/10.1016/j.patcog.2010.10.024>. URL <https://www.sciencedirect.com/science/article/pii/S003132031000508X>. Computer Analysis of Images and Patterns.
- [35] Zhongyu Li, Xiaofan Zhang, Henning Müller, and Shaoting Zhang. Large-scale retrieval for medical image analytics: A comprehensive review. *Medical Image Analysis*, 43:66–84, 2018. ISSN 1361-8415. DOI: <https://doi.org/10.1016/j.media.2017.09.007>. URL <https://www.sciencedirect.com/science/article/pii/S136184151730138X>.
- [36] Sergio Cubero, Nuria Aleixos, Enrique Moltó, Juan Gómez-Sanchis, and Jose Blasco. Advances in machine vision applications for automatic inspection and quality evaluation of fruits and vegetables. *Food and Bioprocess Technology*, 4(4):487–504, July 2010. DOI: 10.1007/s11947-010-0411-8. URL <https://doi.org/10.1007/s11947-010-0411-8>.
- [37] Pankaj B. Pathare, Umezuruike Linus Opara, and Fahad Al-Julanda Al-Said. Colour measurement and analysis in fresh and processed foods: A review. *Food and Bioprocess Technology*, 6(1):36–60, May 2012. DOI: 10.1007/s11947-012-0867-9. URL <https://doi.org/10.1007/s11947-012-0867-9>.
- [38] Di Wu and Da-Wen Sun. Colour measurements by computer vision for food quality control – a review. *Trends in Food Science & Technology*, 29(1):5–20, January 2013. DOI: 10.1016/j.tifs.2012.08.004. URL <https://doi.org/10.1016/j.tifs.2012.08.004>.
- [39] Hyo Sung Jung, Peter Verwilt, Won Young Kim, and Jong Seung Kim. Fluorescent and colorimetric sensors for the detection of humidity or water content. *Chem. Soc. Rev.*, 45(5):1242–1256, 2016. DOI: 10.1039/c5cs00494b. URL <https://doi.org/10.1039/c5cs00494b>.

- [40] Arno Seeboth, Detlef Löttsch, Ralf Ruhmann, and Olaf Muehling. Thermochromic polymers—function by design. *Chemical Reviews*, 114(5):3037–3068, January 2014. DOI: 10.1021/cr400462e. URL <https://doi.org/10.1021/cr400462e>.
- [41] Yanan Zhang and Loong-Tak Lim. Colorimetric array indicator for NH₃ and CO₂ detection. *Sensors and Actuators B: Chemical*, 255:3216–3226, February 2018. DOI: 10.1016/j.snb.2017.09.148. URL <https://doi.org/10.1016/j.snb.2017.09.148>.
- [42] Xu dong Wang and Otto S. Wolfbeis. Optical methods for sensing and imaging oxygen: materials, spectroscopies and applications. *Chem. Soc. Rev.*, 43(10):3666–3761, 2014. DOI: 10.1039/c4cs00039k. URL <https://doi.org/10.1039/c4cs00039k>.
- [43] Steven A. Shafer. Using color to separate reflection components. *Color Research & Application*, 10(4): 210–218, 1985. DOI: 10.1002/col.5080100409. URL <https://doi.org/10.1002/col.5080100409>.
- [44] Ming Gong, Hua Li, and Weiguo Cao. Moment invariants to affine transformation of colours. *Pattern Recognition Letters*, 34(11):1240–1251, August 2013. DOI: 10.1016/j.patrec.2013.03.038. URL <https://doi.org/10.1016/j.patrec.2013.03.038>.
- [45] Christian Driau, Cristian Fàbrega, Ismael Benito-Altamirano, Peter Pfeiffer, Olga Casals, Hongqiang Li, and Joan Daniel Prades. How to implement a selective colorimetric gas sensor with off the shelf components? *Sensors and Actuators, B: Chemical*, 293(October 2018):41–44, 2019. ISSN 09254005. DOI: 10.1016/j.snb.2019.04.117.
- [46] ISO Central Secretary. Information technology - automatic identification and data capture techniques - qr code bar code symbology specification. ISO ISO/IEC 16022:2006, International Organization for Standardization, 2006. URL <https://www.iso.org/standard/44230.html>.
- [47] ISO Central Secretary. Information technology — international symbology specification — maxicode. ISO ISO/IEC 16023:2000, International Organization for Standardization, 2000. URL <https://www.iso.org/standard/29835.html>.
- [48] ISO Central Secretary. Information technology — automatic identification and data capture techniques — aztec code bar code symbology specification. ISO ISO/IEC 24778:2008, International Organization for Standardization, 2000. URL <https://www.iso.org/standard/41548.html>.
- [49] Waldemar Berchtold, Huajian Liu, Martin Steinebach, Dominik Klein, Tobias Senger, and Nicolas Thenee. JAB code - a versatile polychrome 2d barcode. *Electronic Imaging*, 2020(3):207–207, January 2020. DOI: 10.2352/issn.2470-1173.2020.3.mobmu-207. URL <https://doi.org/10.2352/issn.2470-1173.2020.3.mobmu-207>.
- [50] Gavin Jancke. High capacity color barcodes (hccb) - microsoft research, 2021. URL <https://www.microsoft.com/en-us/research/project/high-capacity-color-barcodes-hccb/>.
- [51] Hazem Al-Otum and Nour Emad Al-Shalabi. Copyright protection of color images for android-based smartphones using watermarking with quick-response code. *Multimedia Tools and Applications*, 77(12): 15625–15655, 2018. ISSN 15737721. DOI: 10.1007/s11042-017-5138-3.
- [52] Luis Rosales-Roldan, Jinhui Chao, Mariko Nakano-Miyatake, and Hector Perez-Meana. Color image ownership protection based on spectral domain watermarking using QR codes and QIM. *Multimedia Tools and Applications*, 77(13):16031–16052, 2018. ISSN 15737721. DOI: 10.1007/s11042-017-5178-8.
- [53] S Annadurai. *Fundamentals of digital image processing*. Pearson Education India, 2007.
- [54] Gang Xu, Renzhe Li, Lu Yang, and Xiaochen Liu. Identification and recovery of the blurred qr code image. In *2012 International Conference on Computer Science and Service System*, pages 2257–2260, 2012. DOI: 10.1109/CSSS.2012.560.
- [55] Stephen B Wicker and Vijay K Bhargava. *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.

- [56] Hung Kuo Chu, Chia Sheng Chang, Ruen Rone Lee, and Niloy J. Mitra. Halftone QR codes. *ACM Transactions on Graphics*, 32(6):1–8, 2013. ISSN 07300301. DOI: 10.1145/2508363.2508408.
- [57] Gonzalo J. Garateguy, Gonzalo R. Arce, Daniel L. Lau, and Ofelia P. Villarreal. QR images: Optimized image embedding in QR codes. *IEEE Transactions on Image Processing*, 23(7):2842–2853, 2014. ISSN 10577149. DOI: 10.1109/TIP.2014.2321501.
- [58] Russ Cox. Qart codes. <https://research.swtch.com/qart>, 2012.
- [59] Itseez. Open source computer vision library. <https://github.com/itseez/opencv>, 2015.
- [60] Lindsey M. Higgins, Marianne McGarry Wolf, and Mitchell J. Wolf. Technological change in the wine market? the role of qr codes and wine apps in consumer wine purchases. *Wine Economics and Policy*, 3(1):19–27, 2014. ISSN 2212-9774. DOI: <https://doi.org/10.1016/j.wep.2014.01.002>. URL <https://www.sciencedirect.com/science/article/pii/S2212977414000039>.
- [61] Simona Violino, Francesca Antonucci, Federico Pallottino, Cristina Cecchini, Simone Figorilli, and Corrado Costa. Food traceability: a term map analysis basic review. *European Food Research and Technology*, 245(10):2089–2099, Oct 2019. ISSN 1438-2385. DOI: 10.1007/s00217-019-03321-0. URL <https://doi.org/10.1007/s00217-019-03321-0>.
- [62] P. Márquez-Neila, J. López-Alberca, J. M. Buenaposada, and L. Baumela. Speeding-up homography estimation in mobile devices. *Journal of Real-Time Image Processing*, 11(1):141–154, 2016. URL www.scopus.com.
- [63] Hugh S. Fairman, Michael H. Brill, and Henry Hemmendinger. How the CIE 1931 color-matching functions were derived from wright-guild data. *Color Research & Application*, 22(1):11–23, February 1997. DOI: 10.1002/(sici)1520-6378(199702)22:1<11::aid-col4>3.0.co;2-7. URL [https://doi.org/10.1002/\(sici\)1520-6378\(199702\)22:1<11::aid-col4>3.0.co;2-7](https://doi.org/10.1002/(sici)1520-6378(199702)22:1<11::aid-col4>3.0.co;2-7).
- [64] Practice for computing the colors of objects by using the CIE system. URL <https://doi.org/10.1520/e0308-15>.
- [65] David L. Fridge. Aberration synthesizer. *Journal of the Optical Society of America*, 50(1):87, January 1960. DOI: 10.1364/josa.50.000087. URL <https://doi.org/10.1364/josa.50.000087>.
- [66] Günter Wyszecki. Proposal for a new color-difference formula. *Journal of the Optical Society of America*, 53(11):1318, November 1963. DOI: 10.1364/josa.53.001318. URL <https://doi.org/10.1364/josa.53.001318>.
- [67] Alan R. Robertson. The CIE 1976 color-difference formulae. *Color Research & Application*, 2(1):7–11, March 1977. DOI: 10.1002/j.1520-6378.1977.tb00104.x. URL <https://doi.org/10.1002/j.1520-6378.1977.tb00104.x>.
- [68] Janos Schanda. *Colorimetry : understanding the CIE system*. CIE/Commission internationale de l’éclairage Wiley-Interscience, Vienna, Austria Hoboken, N.J, 2007. ISBN 9780470049044.
- [69] Li Long and Shan Dongri. Review of camera calibration algorithms. In *Advances in Intelligent Systems and Computing*, pages 723–732. Springer Singapore, 2019. DOI: 10.1007/978-981-13-6861-5_61. URL https://doi.org/10.1007/978-981-13-6861-5_61.
- [70] J.-P. Braquelaire and L. Brun. Comparison and optimization of methods of color image quantization. *IEEE Transactions on Image Processing*, 6(7):1048–1052, July 1997. DOI: 10.1109/83.597280. URL <https://doi.org/10.1109/83.597280>.
- [71] Mary Nielsen and Michael Stokes. The creation of the srgb icc profile. In *Color and Imaging Conference*, volume 1998, pages 253–257. Society for Imaging Science and Technology, 1998.
- [72] Huw Morgan and Miloslav Druckmüller. Multi-scale gaussian normalization for solar image processing. *Solar Physics*, 289(8):2945–2955, April 2014. DOI: 10.1007/s11207-014-0523-9. URL <https://doi.org/10.1007/s11207-014-0523-9>.

- [73] Magudeeswaran Veluchamy and Bharath Subramani. Image contrast and color enhancement using adaptive gamma correction and histogram equalization. *Optik*, 183:329–337, April 2019. DOI: 10.1016/j.ijleo.2019.02.054. URL <https://doi.org/10.1016/j.ijleo.2019.02.054>.
- [74] Payel Roy, Saurab Dutta, Nilanjan Dey, Goutami Dey, Sayan Chakraborty, and Ruben Ray. Adaptive thresholding: A comparative study. In *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. IEEE, July 2014. DOI: 10.1109/iccicct.2014.6993140. URL <https://doi.org/10.1109/iccicct.2014.6993140>.
- [75] Yao Xiang, Beiji Zou, and Hong Li. Selective color transfer with multi-source images. *Pattern Recognition Letters*, 30(7):682–689, May 2009. DOI: 10.1016/j.patrec.2009.01.004. URL <https://doi.org/10.1016/j.patrec.2009.01.004>.
- [76] John E Greivenkamp. *Field guide to geometrical optics*, volume 1. SPIE press Bellingham, WA, 2004.
- [77] Naoto Yokoya, Claas Grohnfeldt, and Jocelyn Chanussot. Hyperspectral and multispectral data fusion: A comparative review of the recent literature. *IEEE Geoscience and Remote Sensing Magazine*, 5(2):29–56, June 2017. DOI: 10.1109/mgrs.2016.2637824. URL <https://doi.org/10.1109/mgrs.2016.2637824>.
- [78] Guido van Rossum. *Python programming language*. Python Software Foundation, 1990. URL <https://www.python.org>.
- [79] Guido Van Rossum et al. Python programming language. In *USENIX annual technical conference*, volume 41, page 36, 2007.
- [80] Jan Erik Solem. *Programming Computer Vision with Python: Tools and algorithms for analyzing images*. "O'Reilly Media, Inc.", 2012.
- [81] Huaxiong Cao, Naijie Gu, Kaixin Ren, and Yi Li. Performance research and optimization on CPython's interpreter. In *Annals of Computer Science and Information Systems*. IEEE, October 2015. DOI: 10.15439/2015f139. URL <https://doi.org/10.15439/2015f139>.
- [82] Joseph Howse, Prateek Joshi, and Michael Beyeler. *OpenCV: computer vision projects with python*. Packt Publishing Ltd, 2016.
- [83] Anaconda software distribution, 2020. URL <https://docs.anaconda.com/>.
- [84] pyenv. pyenv – simple python version management. <https://github.com/pyenv/pyenv>, 2022.
- [85] Dirk Merkel. Docker – lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.
- [86] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. DOI: 10.1038/s41586-020-2649-2. URL <https://doi.org/10.1038/s41586-020-2649-2>.
- [87] Ralf Gommers, Pauli Virtanen, Evgeni Burovski, Warren Weckesser, Travis E. Oliphant, David Cournapeau, Tyler Reddy, Matt Haberland, alexbrc, Pearu Peterson, Andrew Nelson, Josh Wilson, endolith, Nikolay Mayorov, Ilhan Polat, Stefan van der Walt, Denis Laxalde, Matthew Brett, Eric Larson, Jarrod Millman, Lars, peterbell10, Paul van Mulbregt, Pamphile Roy, CJ Carey, eric jones, Atsushi Sakai, Eric Moore, Robert Kern, and Kai. *scipy/scipy: Scipy 1.8.0rc2*, December 2021. URL <https://doi.org/10.5281/zenodo.5796897>.
- [88] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3): 90–95, 2007. DOI: 10.1109/mcse.2007.55. URL <https://doi.org/10.1109/mcse.2007.55>.

- [89] Jeff Reback, jbrockmendel, Wes McKinney, Joris Van den Bossche, Tom Augspurger, Phillip Cloud, Simon Hawkins, Matthew Roeschke, gyoung, Sinhrks, Adam Klein, Patrick Hoefler, Terji Petersen, Jeff Tratner, Chang She, William Ayd, Shahar Naveh, Marc Garcia, JHM Darbyshire, Jeremy Schendel, Andy Hayden, Richard Shadrach, Daniel Saxton, Marco Edward Gorelli, Fangchen Li, Matthew Zeitlin, Vytautas Jancauskas, Ali McMaster, Pietro Battiston, and Skipper Seabold. pandas-dev/pandas: Pandas 1.4.0rc0, January 2022. URL <https://doi.org/10.5281/zenodo.5824773>.
- [90] Stephan Hoyer, Alex Kleeman and Eugene Brevdo. xarray – n-d labeled arrays and datasets in python. <https://github.com/pydata/xarray>, 2014.
- [91] Hugo van Kemenade, Andrew Murray, wiredfool, Alex Clark, Alexander Karpinsky, Ondrej Baranovič, Christoph Gohlke, Jon Dufresne, Brian Crowell, David Schmidt, Konstantin Kopachev, Alastair Houghton, Sandro Mani, Steve Landey, vashek, Josh Ware, Jason Douglas, Stanislaw T., David Caro, Uriel Martinez, Steve Kossouho, Riley Lahd, Antony Lee, Eric W. Brown, Oliver Tonnhofer, Mickael Bonfill, Peter Rowlands, Fahad Al-Saidi, and German Novikov. python-pillow/pillow: 9.0.0, January 2022. URL <https://doi.org/10.5281/zenodo.5813885>.
- [92] Almar Klein, Sebastian Wallkötter, Steven Silvester, Anthony Tanbakuchi, Paul Müller, Juan Nunez-Iglesias, actions user, Mark Harfouche, Antony Lee, Matt McCormick, OrganicIrradiation, Arash Rai, Ariel Ladegaard, Tim D. Smith, Ghislain Vaillant, jackwalker64, Joel Nises, Miloš Komarčević, rreilink, Ischr, Dennis, Hugo van Kemenade, Maximilian Schambach, Chris Dusold, DavidKorzczynski, Felix Kohlgrüber, Ge Yang, Graham Inggs, Joe Singleton, and Michael. imageio/imageio: v2.13.5, December 2021. URL <https://doi.org/10.5281/zenodo.5800390>.
- [93] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte, et al. imgaug. <https://github.com/aleju/imgaug>, 2020. Online; accessed 01-Feb-2020.
- [94] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
- [95] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014. ISSN 2167-8359. DOI: 10.7717/peerj.453. URL <https://doi.org/10.7717/peerj.453>.
- [96] Nikhil Ketkar. Introduction to keras. In *Deep learning with Python*, pages 97–111. Springer, 2017.
- [97] Lincoln Loop. Pure python qr code generator. <https://github.com/lincolnloop/python-qr-code>, 2010.
- [98] SourceForge. Zbar. <http://zbar.sourceforge.net/>, 2009.
- [99] London Natural History Museum. pyzbar - python wrapper for zbar. <https://github.com/NaturalHistoryMuseum/pyzbar>, 2016.
- [100] Sean Owen, Daniel Switkin, and ZXing Team. Zxing ("zebra crossing"). <https://github.com/zxing/zxing>, 2008.
- [101] Yaoqi Peng, Lingxian Zhang, Zhixing Song, Jin Yan, Xinxing Li, and Zhenbo Li. A qr code based tracing method for fresh pork quality in cold chain. *Journal of Food Process Engineering*, 41(4):e12685, 2018. DOI: <https://doi.org/10.1111/jfpe.12685>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/jfpe.12685>.
- [102] K. Seino, S. Kuwabara, S. Mikami, Y. Takahashi, M. Yoshikawa, H. Narumi, K. Koganezaki, T. Wakabayashi, and A. Nagano. Development of the traceability system which secures the safety of fishery

- products using the qr code and a digital signature. In *Oceans '04 MTS/IEEE Techno-Ocean '04 (IEEE Cat. No.04CH37600)*, volume 1, pages 476–481, Nov 2004. DOI: 10.1109/OCEANS.2004.1402962.
- [103] Jian-Ping Qian, Xin-Ting Yang, Xiao-Ming Wu, Li Zhao, Bei-Lei Fan, and Bin Xing. A traceability system incorporating 2d barcode and rfid technology for wheat flour mills. *Computers and Electronics in Agriculture*, 89:76–85, 2012. ISSN 0168-1699. DOI: <https://doi.org/10.1016/j.compag.2012.08.004>. URL <https://www.sciencedirect.com/science/article/pii/S0168169912002050>.
- [104] Thomas F. Scherr, Sparsh Gupta, David W. Wright, and Frederick R. Haselton. An embedded barcode for “connected” malaria rapid diagnostic tests. *Lab Chip*, 17:1314–1322, 2017. DOI: 10.1039/C6LC01580H. URL <http://dx.doi.org/10.1039/C6LC01580H>.
- [105] Bora Yoon, Hyora Shin, Eun-Mi Kang, Dae Won Cho, Kayeong Shin, Hoeil Chung, Chan Woo Lee, and Jong-Man Kim. Inkjet-compatible single-component polydiacetylene precursors for thermochromic paper sensors. *ACS Applied Materials & Interfaces*, 5(11):4527–4535, Jun 2013. ISSN 1944-8244. DOI: 10.1021/am303300g. URL <https://doi.org/10.1021/am303300g>.
- [106] Aidong Sun, Yan Sun, and Caixing Liu. The QR-code reorganization in illegible snapshots taken by mobile phones. *IEEE*, 2007. DOI: 10.1109/iccsa.2007.86.
- [107] Jeng-An Lin and Chiou-Shann Fuh. 2D barcode image decoding. *Hindawi Limited*, pages 1–10, 2013. DOI: 10.1155/2013/848276.
- [108] Kejing Li, Fanwu Meng, Zhipeng Huang, and Qi Wang. A correction algorithm of QR code on cylindrical surface. *Journal of Physics: Conference Series*, 1237:022006, 6 2019. DOI: 10.1088/1742-6596/1237/2/022006. URL <https://doi.org/10.1088%2F1742-6596%2F1237%2F2%2F022006>.
- [109] K. Lay, L. Wang, and C. Wang. Rectification of qr-code images using the parametric cylindrical surface model. *2015 International Symposium on Next-Generation Electronics (ISNE)*, pages 1–5, 2015.
- [110] Kuen-Tsair Lay and Ming-Hao Zhou. Perspective projection for decoding of qr codes posted on cylinders. *2017 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 39–42, 2017.
- [111] Xiaochao Li, Zhifeng Shi, Donghui Guo, and Shan He. Reconstruct algorithm of 2d barcode for reading the qr code on cylindrical surface. *2013 International Conference on Anti-Counterfeiting, Security and Identification (ASID)*, pages 1–5, 2013.
- [112] Kazumoto Tanaka. Bent qr code image rectification method based on image-to-image translation network. In Xin-She Yang, Simon Sherratt, Nilanjan Dey, and Amit Joshi, editors, *Proceedings of Sixth International Congress on Information and Communication Technology*, pages 685–692, Singapore, 2022. Springer Singapore. ISBN 978-981-16-2377-6.
- [113] Lina Huo, Jianxing Zhu, Pradeep Kumar Singh, and Pljonkin Anton Pavlovich. Research on qr image code recognition system based on artificial intelligence algorithm. *Journal of Intelligent Systems*, 30(1): 855–867, 2021. DOI: [doi:10.1515/jisys-2020-0143](https://doi.org/10.1515/jisys-2020-0143). URL <https://doi.org/10.1515/jisys-2020-0143>.
- [114] Ryosuke Kikuchi, Sora Yoshikawa, Pradeep Kumar Jayaraman, Jianmin Zheng, and Takashi Maekawa. Embedding qr codes onto b-spline surfaces for 3d printing. *Computer-Aided Design*, 102:215–223, 2018. ISSN 0010-4485. DOI: <https://doi.org/10.1016/j.cad.2018.04.025>. URL <https://www.sciencedirect.com/science/article/pii/S0010448518302537>. Proceeding of SPM 2018 Symposium.
- [115] F. L. Bookstein. Principal warps: thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–585, 1989.
- [116] A.M. Bazen and Sabih H. Gerez. Fingerprint matching by thin-plate spline modelling of elastic deformations. *Pattern recognition*, 36(8):1859–1867, 2003. ISSN 0031-3203. DOI: 10.1016/S0031-3203(03)00036-0. SAS 03-061.

- [117] Arun Ross, Sarat Dass, and Anil Jain. A deformable model for fingerprint matching. *Pattern Recognition*, 38(1):95–103, 2005. ISSN 0031-3203. DOI: <https://doi.org/10.1016/j.patcog.2003.12.021>. URL <https://www.sciencedirect.com/science/article/pii/S0031320304002444>.
- [118] Baoguang Shi, Mingkun Yang, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. ASTER: An attentional scene text recognizer with flexible rectification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2035–2048, September 2019. DOI: 10.1109/tpami.2018.2848939. URL <https://doi.org/10.1109/tpami.2018.2848939>.
- [119] Yang Yang, Sim Heng Ong, and Kelvin Weng Chiong Foong. A robust global and local mixture distance based non-rigid point set registration. *Pattern Recognition*, 48(1):156–173, January 2015. DOI: 10.1016/j.patcog.2014.06.017. URL <https://doi.org/10.1016/j.patcog.2014.06.017>.
- [120] E. Casas-Alvero. *Analytic Projective Geometry*. European Mathematical Society, Zürich, Switzerland, 2014.
- [121] Alexander Jung. *imgaug Documentation*, 2018.
- [122] Yves Van Gennip, Prashant Athavale, Jérôme Gilles, and Rustum Choksi. A Regularization Approach to Blind Deblurring and Denoising of QR Barcodes. *IEEE Transactions on Image Processing*, 24(9):2864–2873, 2015. ISSN 10577149. DOI: 10.1109/TIP.2015.2432675.
- [123] Adrien Bartoli, Mathieu Perriollat, and Sylvie Chambon. Generalized thin-plate spline warps. *International Journal of Computer Vision*, 88(1):85–110, October 2009. DOI: 10.1007/s11263-009-0303-4. URL <https://doi.org/10.1007/s11263-009-0303-4>.
- [124] N. Arad, N. Dyn, D. Reissfeld, and Y. Yeshurun. Image warping by radial basis functions: Application to facial expressions. *CVGIP: Graphical Models and Image Processing*, 56(2):161–172, 1994. ISSN 1049-9652. DOI: <https://doi.org/10.1006/cgip.1994.1015>. URL <https://www.sciencedirect.com/science/article/pii/S1049965284710157>.
- [125] Gianluca Donato and Serge Belongie. Approximate thin plate spline mappings. In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *Computer Vision — ECCV 2002*, pages 21–31, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-47977-2.
- [126] Boxuan Li, Benfei Wang, Xiaojun Tan, Jiezhong Wu, and Liangliang Wei. Corner location and recognition of single ArUco marker under occlusion based on YOLO algorithm. *Journal of Electronic Imaging*, 30(03), May 2021. DOI: 10.1117/1.jei.30.3.033012. URL <https://doi.org/10.1117/1.jei.30.3.033012>.
- [127] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. URL <http://arxiv.org/abs/1804.02767>.
- [128] Markéta Dubská, Adam Herout, and Jiří Havel. Real-time precise detection of regular grids and matrix codes. *Journal of Real-Time Image Processing*, 11(1):193–200, February 2013. DOI: 10.1007/s11554-013-0325-6. URL <https://doi.org/10.1007/s11554-013-0325-6>.
- [129] Christoph Ruppert, Navneet Phogat, Stefan Laufer, Matthias Kohl, and Hans Peter Deigner. A smartphone readout system for gold nanoparticle-based lateral flow assays: application to monitoring of digoxigenin. *Microchimica Acta*, 186(2), 2019. ISSN 14365073. DOI: 10.1007/s00604-018-3195-6.
- [130] Henryk Blasinski, Orhan Bulan, and Gaurav Sharma. Per-colorant-channel color barcodes for mobile applications: An interference cancellation framework. *IEEE Transactions on Image Processing*, 22(4):1498–1511, 2013. ISSN 10577149. DOI: 10.1109/TIP.2012.2233483.
- [131] Marco Querini and Giuseppe F. Italiano. Reliability and data density in high capacity color barcodes. *Computer Science and Information Systems*, 11(4):1595–1616, 2014. ISSN 18200214. DOI: 10.2298/CSIS131218054Q.

- [132] Max E. Vizcarra Melgar, Alexandre Zaghetto, Bruno Macchiavello, and Anderson C A Nascimento. QQR codes: Colored quick-response codes. In *2012 IEEE Second International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, volume 2401, pages 321–325. IEEE, sep 2012. ISBN 978-1-4673-1547-0. DOI: 10.1109/ICCE-Berlin.2012.6336526.
- [133] Götz Trenkler. Continuous univariate distributions. *Computational Statistics & Data Analysis*, 21(1): 119, 1996. ISSN 01679473. DOI: 10.1016/0167-9473(96)90015-8.
- [134] Mary Pagnutti, Robert E. Ryan, George Cazenavette, Maxwell Gold, Ryan Harlan, Edward Leggett, and James Pagnutti. Laying the foundation to use Raspberry Pi 3 V2 camera module imagery for scientific and engineering purposes. *Journal of Electronic Imaging*, 26(1):013014, 2017. ISSN 1017-9909. DOI: 10.1117/1.jei.26.1.013014.
- [135] Claudio Cusano, Paolo Napoletano, and Raimondo Schettini. Evaluating color texture descriptors under large variations of controlled lighting conditions. *Journal of the Optical Society of America A*, 33(1):17, 2016. ISSN 1084-7529. DOI: 10.1364/josaa.33.000017.
- [136] A Grillo, A Lentini, M Querini, and G F Italiano. High capacity colored two dimensional codes. In *Proceedings of the International Multiconference on Computer Science and Information Technology*. IEEE, October 2010. DOI: 10.1109/imcsit.2010.5679869. URL <https://doi.org/10.1109/imcsit.2010.5679869>.
- [137] Jean Duchon. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 10(R3):5–12, 1976.
- [138] Jean Meinguet. Multivariate interpolation at arbitrary points made simple. *Zeitschrift für angewandte Mathematik und Physik ZAMP*, 30(2):292–304, March 1979. DOI: 10.1007/bf01601941. URL <https://doi.org/10.1007/bf01601941>.
- [139] K. Rohr, H.S. Stiehl, R. Sprengel, T.M. Buzug, J. Weese, and M.H. Kuhn. Landmark-based elastic registration using approximating thin-plate splines. *IEEE Transactions on Medical Imaging*, 20(6): 526–534, June 2001. DOI: 10.1109/42.929618. URL <https://doi.org/10.1109/42.929618>.
- [140] W R Crum, T Hartkens, and D L G Hill. Non-rigid image registration: theory and practice. *The British Journal of Radiology*, 77(suppl_2):S140–S153, December 2004. DOI: 10.1259/bjr/25329214. URL <https://doi.org/10.1259/bjr/25329214>.
- [141] Philippe Colantoni, Jean-Baptiste Thomas, and Jon Y. Hardeberg. High-end colorimetric display characterization using an adaptive training set. *Journal of the Society for Information Display*, 19(8):520, 2011. DOI: 10.1889/jSID19.8.520. URL <https://doi.org/10.1889/jSID19.8.520>.
- [142] Ante Poljicak, Jurica Dolic, and Jesenka Pibernik. An optimized radial basis function model for color characterization of a mobile device display. *Displays*, 41:61–68, January 2016. DOI: 10.1016/j.displa.2015.12.005. URL <https://doi.org/10.1016/j.displa.2015.12.005>.
- [143] Gaurav Sharma and Mark Q. Shaw. Thin-plate splines for printer data interpolation. In *2006 14th European Signal Processing Conference*, pages 1–5, 2006.
- [144] M. D. Buhmann. Radial basis functions. *Acta Numerica*, 9:1–38, January 2000. DOI: 10.1017/S0962492900000015. URL <https://doi.org/10.1017/S0962492900000015>.
- [145] R. Sprengel, K. Rohr, and H.S. Stiehl. Thin-plate spline approximation for image registration. In *Proceedings of 18th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. DOI: 10.1109/iembs.1996.652767. URL <https://doi.org/10.1109/iembs.1996.652767>.
- [146] Peter Vincent Gehler, Carsten Rother, Andrew Blake, Tom Minka, and Toby Sharp. Bayesian color constancy revisited. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2008. DOI: 10.1109/cvpr.2008.4587765. URL <https://doi.org/10.1109/cvpr.2008.4587765>.

- [147] Ghalia Hemrit, Graham D. Finlayson, Arjan Gijsenij, Peter V. Gehler, Simone Bianco, and Mark S. Drew. Rehabilitating the color checker dataset for illuminant estimation. *CoRR*, abs/1805.12262, 2018. URL <http://arxiv.org/abs/1805.12262>.
- [148] Weixin Luo, Xuan Yang, Xiaoxiao Nan, and Bingfeng Hu. GPU accelerated 3d image deformation using thin-plate splines. In *2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICSS)*. IEEE, August 2014. DOI: 10.1109/hpcc.2014.168. URL <https://doi.org/10.1109/hpcc.2014.168>.
- [149] Dan Kalman. The generalized vandermonde matrix. *Mathematics Magazine*, 57(1):15–21, January 1984. DOI: 10.1080/0025570x.1984.11977069. URL <https://doi.org/10.1080/0025570x.1984.11977069>.
- [150] David R. Bull. Digital picture formats and representations. In *Communicating Pictures*, pages 99–132. Elsevier, 2014. DOI: 10.1016/b978-0-12-405906-1.00004-0. URL <https://doi.org/10.1016/b978-0-12-405906-1.00004-0>.
- [151] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.
- [152] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22, 2011.
- [153] Thomas Mansencal, Michael Mauderer, Michael Parsons, Nick Shaw, Kevin Wheatley, Sean Cooper, Jean D. Vandenberg, Luke Canavan, Katherine Crowson, Ofek Lev, Katrin Leinweber, Shriramana Sharma, Troy James Sobotka, Dominik Moritz, Matt Pppp, Chinmay Rane, Pavithra Eswaramoorthy, John Mertic, Ben Pearlstine, Manuel Leonhardt, Olli Niemitalo, Marek Szymanski, Maximilian Schambach, Sianyi Huang, Mike Wei, Nishant Joywardhan, Omar Wagih, Pawel Redman, Joseph Goldstone, and Stephen Hill. Colour 0.3.16, January 2020. URL <https://doi.org/10.5281/zenodo.3757045>.
- [154] Dilip Prasad, Rang Nguyen, and Michael Brown. Quick approximation of camera’s spectral response from casual lighting. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 844–851, 2013.
- [155] Roy S. Berns. Predicting camera color quality. *Archiving Conference*, 2021(1):61–64, June 2021. DOI: 10.2352/issn.2168-3204.2021.1.0.14. URL <https://doi.org/10.2352/issn.2168-3204.2021.1.0.14>.
- [156] R. Fry and S. McManus. Smooth bump functions and the geometry of banach spaces. *Expositiones Mathematicae*, 20(2):143–183, 2002. DOI: 10.1016/s0723-0869(02)80017-2. URL [https://doi.org/10.1016/s0723-0869\(02\)80017-2](https://doi.org/10.1016/s0723-0869(02)80017-2).
- [157] Bitu Akram, Usman R. Alim, and Faramarz F. Samavati. Cinapact-splines: A family of infinitely smooth, accurate and compactly supported splines. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Ioannis Pavlidis, Rogerio Feris, Tim McGraw, Mark Elendt, Regis Kopper, Eric Ragan, Zhao Ye, and Gunther Weber, editors, *Advances in Visual Computing*, pages 819–829, Cham, 2015. Springer International Publishing. ISBN 978-3-319-27857-5.
- [158] C. Fàbrega, O. Casals, F. Hernández-Ramírez, and J.D. Prades. A review on efficient self-heating in nanowire sensors: Prospects for very-low power devices. *Sensors and Actuators B: Chemical*, 256: 797–811, March 2018. DOI: 10.1016/j.snb.2017.10.003. URL <https://doi.org/10.1016/j.snb.2017.10.003>.
- [159] Luis Fernández, Alba Pons, Oriol Monereo, Ismael Benito-Altamirano, Elena Xuriguera, Olga Casals, Cristian Fàbrega, Andreas Waag, and Joan Daniel Prades. NO₂ measurements with RGB sensors for easy in-field test. *Proceedings*, 1(4):471, August 2017. DOI: 10.3390/proceedings1040471. URL <https://doi.org/10.3390/proceedings1040471>.

- [160] K. Schmitt, K. Tarantik, C. Pannek, I. Benito-Altamirano, O. Casals, C. Fàbrega, A. Romano-Rodríguez, J. Wöllenstein, and J. D. Prades. Colorimetric sensor for bad odor detection using automated color correction. In Luis Fonseca, Mika Prunnila, and Erwin Peiner, editors, *SPIE Proceedings*. SPIE, June 2017. DOI: 10.1117/12.2265990. URL <https://doi.org/10.1117/12.2265990>.
- [161] Christian Driau, Cristian Fabrega, Ismael Benito-Altamirano, Peter Pfeiffer, Olga Casals, and Joan Daniel Prades. Compact, versatile and cost-effective colorimetric gas sensors. In *2019 IEEE International Symposium on Olfaction and Electronic Nose (ISOEN)*. IEEE, May 2019. DOI: 10.1109/isoen.2019.8823240. URL <https://doi.org/10.1109/isoen.2019.8823240>.
- [162] Christian Driau, Olga Casals, Ismael Benito-Altamirano, Joan Daniel Prades, and Cristian Fàbrega. Revisiting colorimetric gas sensors: Compact, versatile and cost-effective. *Proceedings*, 56(1): 20, December 2020. DOI: 10.3390/proceedings2020056020. URL <https://doi.org/10.3390/proceedings2020056020>.
- [163] Laura Engel, Ismael Benito-Altamirano, Karina R. Tarantik, Martin Dold, Carolin Pannek, J. Daniel Prades, and Jürgen Wöllenstein. Printable colorimetric sensors for the detection of formaldehyde in ambient air. *ECS Meeting Abstracts*, MA2020-01(27):2029–2029, May 2020. DOI: 10.1149/ma2020-01272029mtgabs. URL <https://doi.org/10.1149/ma2020-01272029mtgabs>.
- [164] Andrew Mills and Graham A. Skinner. Water-based colourimetric optical indicators for the detection of carbon dioxide. *The Analyst*, 135(8):1912, 2010. DOI: 10.1039/c000688b. URL <https://doi.org/10.1039/c000688b>.
- [165] Andrew Mills, Graham A. Skinner, and Pauline Grosshans. Intelligent pigments and plastics for CO₂ detection. *Journal of Materials Chemistry*, 20(24):5008, 2010. DOI: 10.1039/c0jm00582g. URL <https://doi.org/10.1039/c0jm00582g>.
- [166] Pradeep Puligundla, Junho Jung, and Sanghoon Ko. Carbon dioxide sensors for intelligent food packaging applications. *Food Control*, 25(1):328–333, May 2012. DOI: 10.1016/j.foodcont.2011.10.043. URL <https://doi.org/10.1016/j.foodcont.2011.10.043>.
- [167] Brooks Instruments. *Brooks® Smart-Series Digital Mass Flow Meters and Controllers – Models 5800-S*, 2008. URL <https://www.brooksinstrument.com/~media/brooks/documentation/products/legacy%20products/brooks/x-tmf-5800s-mfc-eng.pdf?la=en>.
- [168] Rick Bitter, Taqi Mohiuddin, and Matt Nawrocki. *LabVIEW: Advanced programming techniques*. Crc Press, 2006.
- [169] Brooks Instruments. *Smart DDE Software – for use with Brooks Digital Mass Flow Meter/Controller Series*, 2013. URL https://www.brooksinstrument.com/-/media/Brooks/documentation/products/Accessories-And-Software/Software/0162-Smart-DDE/software-installation-manual-smart-dde.ashx?rev=a5f7e7e5b78b442bb1de389dcefbff29&sc_lang=en.
- [170] Miguel Grinberg. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.
- [171] Bokeh Development Team. *Bokeh: Python library for interactive visualization*, 2018. URL <https://bokeh.pydata.org/en/latest/>.
- [172] Czarek Tomczak et al. cefpython. <https://github.com/cztomczak/cefpython>, 2022. Online; accessed 09-Jan-2022.
- [173] *Datasheet Sensirion SCD30 Sensor Module – CO₂, humidity, and temperature sensor*. Sensirion, the sensor company, 07 2019. Version 0.94.
- [174] Samuel Schaefer. *Colorimetric water quality sensing with mobile smart phones*. PhD thesis, University of British Columbia, 2014.

- [175] Yunpeng Xing, Qian Zhu, Xiaohong Zhou, and Peishi Qi. A dual-functional smartphone-based sensor for colorimetric and chemiluminescent detection: A case study for fluoride concentration mapping. *Sensors and Actuators B: Chemical*, 319:128254, September 2020. DOI: 10.1016/j.snb.2020.128254. URL <https://doi.org/10.1016/j.snb.2020.128254>.
- [176] M Muniesa, E Ballesté, Lejla Imamovic, M Pascual-Benito, D Toribio-Avedillo, F Lucena, AR Blanch, and J Jofre. Bluephage: A rapid method for the detection of somatic coliphages used as indicators of fecal pollution in water. *Water research*, 128:10–19, 2018.
- [177] I. Hernández-Neuta, F. Neumann, J. Brightmeyer, T. Ba Tis, N. Madaboosi, Q. Wei, A. Ozcan, and M. Nilsson. Smartphone-based clinical diagnostics: towards democratization of evidence-based health care. *Journal of Internal Medicine*, 285(1):19–39, September 2018. DOI: 10.1111/joim.12820. URL <https://doi.org/10.1111/joim.12820>.
- [178] Wesley Wei-Wen Hsiao, Trong-Nghia Le, Dinh Minh Pham, Hui-Hsin Ko, Huan-Cheng Chang, Cheng-Chung Lee, Neha Sharma, Cheng-Kang Lee, and Wei-Hung Chiang. Recent advances in novel lateral flow technologies for detection of COVID-19. *Biosensors*, 11(9):295, August 2021. DOI: 10.3390/bios11090295. URL <https://doi.org/10.3390/bios11090295>.
- [179] Evgeni Eltzov, Sarah Guttel, Adarina Low Yuen Kei, Prima Dewi Sinawang, Rodica E. Ionescu, and Robert S. Marks. Lateral flow immunoassays - from paper strip to smartphone technology. *Electroanalysis*, 27(9):2116–2130, August 2015. DOI: 10.1002/elan.201500237. URL <https://doi.org/10.1002/elan.201500237>.
- [180] Andrew S. Paterson, Balakrishnan Raja, Vinay Mandadi, Blane Townsend, Miles Lee, Alex Buell, Binh Vu, Jakoah Brgoch, and Richard C. Willson. A low-cost smartphone-based platform for highly sensitive point-of-care testing with persistent luminescent phosphors. *Lab on a Chip*, 17(6):1051–1059, 2017. DOI: 10.1039/c6lc01167e. URL <https://doi.org/10.1039/c6lc01167e>.
- [181] Fleur W. Kong, Caitlin Horsham, Alexander Ngoo, H. Peter Soyer, and Monika Janda. Review of smartphone mobile applications for skin cancer detection: what are the changes in availability, functionality, and costs to users over time? *International Journal of Dermatology*, 60(3):289–308, September 2020. DOI: 10.1111/ijd.15132. URL <https://doi.org/10.1111/ijd.15132>.
- [182] Evgin Goceri. Impact of deep learning and smartphone technologies in dermatology: Automated diagnosis. In *2020 Tenth International Conference on Image Processing Theory, Tools and Applications (IPTA)*. IEEE, November 2020. DOI: 10.1109/ipta50016.2020.9286706. URL <https://doi.org/10.1109/ipta50016.2020.9286706>.
- [183] David Boccara, Farid Bekara, Sabri Soussi, Matthieu Legrand, Marc Chaouat, Maurice Mimoun, and Kevin Serror. Ongoing development and evaluation of a method of telemedicine: Burn care management with a smartphone. *Journal of Burn Care & Research*, 39(4):580–584, December 2017. DOI: 10.1093/jbcr/irx022. URL <https://doi.org/10.1093/jbcr/irx022>.
- [184] C. Grana, G. Pellacani, S. Seidenari, and R. Cucchiara. Color calibration for a dermatological video camera system. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004*. IEEE, 2004. DOI: 10.1109/icpr.2004.1334649. URL <https://doi.org/10.1109/icpr.2004.1334649>.
- [185] Yves Vander Haeghen and Jean Marie Naeyaert. Consistent cutaneous imaging with commercial digital cameras. *Archives of Dermatology*, 142(1), January 2006. DOI: 10.1001/archderm.142.1.42. URL <https://doi.org/10.1001/archderm.142.1.42>.
- [186] Blaž Cugmas and Eva Štruc. Accuracy of an affordable smartphone-based teledermoscopy system for color measurements in canine skin. *Sensors*, 20(21):6234, October 2020. DOI: 10.3390/s20216234. URL <https://doi.org/10.3390/s20216234>.