UNIVERSITY OF CALIFORNIA
SANTA CRUZ

MOBILE VISION MULTICOLOR TARGET DETECTION AND COLOR INFORMATION DECODING

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

by

Homayoun Bagherinia

December 2014

The Dissertation of Homayoun Bagherinia
is approved:

Table of Contents

List of Figures

List of Tables

Abstract

Mobile Vision Multicolor Target Detection and Color Information Decoding

by

Homayoun Bagherinia

Color-based computer vision approaches have proven pertinent in detecting and classifying objects in various areas ranging from industrial inspection to mobile vision and biomedical applications.

Smartphones are becoming an increasing research platform due to their high-quality cameras and programmability as well as their portability. The computational power of smartphones has been improving since the last decade which enables us to make use of them as a target detection and decoding device.

In this thesis, we use computer vision approaches to propose effective detection and decoding of multicolor surfaces. Our methods address the main issues related to designing a practical detection and decoding problems, namely robustness and computational efficiency.

To this end, this thesis offers contributions in multicolor detection and decoding in mobile vision. In the first part of this thesis, we explore the potential use of smartphones to detect a special multicolor marker that could potentially help blind persons to find their way around in a suitably equipped environment. The use of multicolor surfaces not only increases the distinctiveness of the marker with respect to the background and thus more reliable detection, but also enables detection by a model-based method that explicitly takes into account the variability of illumination. In the second part of this thesis, we explore color information access by allowing users to decode a color barcode from a barcode image. Our image-based color barcode decoding approaches are motivated by the necessity of increasing information density in a limited space. In our approaches, we address practical issues such as changes of the observed colors due to changing illuminant, specular reflection, and blur-induced color mixing from neighboring barcode patches. In our initial decoding approaches, we consider groups of color surfaces that can be decoded under variety of illuminants, exploiting the fact that joint color changes can be represented by a low-dimensional subspace. Thus, decoding a group of color surfaces is equivalent to searching for the nearest subspace in a dataset. In another approach, rather than decoding individual patches or using a clustering method, our algorithm iteratively decodes the colors of all patches across the barcode image by minimizing the overall observation error. We achieve high information rates using only three reference colors with a very low probability of decoding error.

To my wife Tamara, and my daughter Ava for their love, support, and sacrifices

Acknowledgments

The work contained in this dissertation represents the results of many years of work made possible only by the collective support of family, friends, colleagues and mentors.

I would like to express my deep gratitude to my advisor, Professor Roberto Manduchi, for his patient guidance, commitment to my growth and development, and most importantly, for encouraging me to study color computer vision several years ago.

I would like to thank Professors Hamid Sadjadpour and James Davis for serving on my committee and reviewing this thesis. I am grateful to all UCSC professors specially Professors Peyman Milanfar, Ali Shakouri, Benjamin Friedlander.

I would like to acknowledge friends and family members who have provided immeasurable support throughout this journey.

The text of this dissertation includes reprints of the following previously published material:

- Homayoun Bagherinia and Roberto Manduchi. Robust real-time detection of multicolor markers on a cell phone. Journal of real-time image processing, 2011.
- Homayoun Bagherinia and Roberto Manduchi. A theory of color barcodes. In IEEE Color and Photometry in Computer Vision (CPCV 2011) Workshop, 2011.
- Homayoun Bagherinia and Roberto Manduchi. High information rate and efficient color barcode decoding. In International Workshop on Color and Photometry in Computer Vision (CPCV 2012) , 2012.

- Homayoun Bagherinia and Roberto Manduchi. Color barcode decoding in the presence of specular reflection. In 4th Color and Photometry in Computer Vision (CPCV 2014) Workshop, 2014.

- Homayoun Bagherinia and Roberto Manduchi. A novel approach for color barcode decoding using smart phones. In IEEE International Conference on Image Processing (ICIP 2014), 2014.

The co-author listed in this publication directed and supervised the research which forms the basis for the dissertation.

Chapter 1

Introduction

Color is crucial to many pattern detection and recognition systems. The image recorded by a camera depends on the physical content of the scene and the characteristics of the camera as well as the surface reflectance spectrum and the unknown illuminant spectrum. Consequently, determining the colors of the surface of an object represented by multiple colors is a challenging operation, especially if multiple light conditions are expected. Therefore, illumination must be controlled or taken into account. The ability of a vision system to model illumination change is central to the recovery of information about the scene from image data, which inevitably has the scene illumination intertwined with the information of interest. Other nuisance parameters include: specularities; unknown or poorly calibrated camera color response; camera non-linearity; color mixing from two nearby surfaces due to blur; quantization and noise.

One of the interesting applications using color is to embed information in the color of printed surfaces. For example, a group of color surfaces can represent a symbol.

This symbol can be detected as a *multicolor marker* (or fiducials) or decoded as a *color barcode* for mobile vision applications. There is surprisingly little research work in the literature on the topic of multicolor-based detection and decoding. Most research works have been mainly focused on color clustering and color indexing, where color indexing determines which surface among a set of possible color surfaces best represents the unknown surface.

1.1 Multicolor-based detection

The first part of this thesis addresses the detection of a special multicolor marker. One of the application of such multicolor marker is to help blind persons to find their way around in a suitably equipped environment. Specifically, the system should be based on simple markers, easily detectable by a cell phone that can be placed in key locations in the environment. A blind person can search for such markers by orienting the camera phone in different directions, effectively scanning the environment. Once a marker is detected by the camera phone, the user is notified by an acoustic signal. If desired, the user can move towards the marker (which could be placed near a point of interest) by keeping track of the marker location via the camera phone. The detection algorithm is a challenging problem due to color mixing caused by motion and incorrect focus, perspective projection, power-constrained mobile platforms, and nuisance parameters mentioned above. A number of markers and algorithms have been proposed enabling fast and robust detection for applications such as augmented reality and robotic localization. The majority of the markers described in the literature use black-and-white patterns. In this case, detection relies on the shape properties of the pattern in the marker. Relatively little attention has been given to design concepts that rely on distinctive color rather than (or in addition to) shape analysis. The use of multicolor surfaces increases the distinctiveness of the marker with respect to the background and thus more reliable detection. Our multicolor-based design approach is a joint design of the color selection of the marker and of the detection algorithm. Without careful color choice and proper processing, color marker detection is likely to fail in realistic conditions with multiple distractors and large variation in illumination. The use of multi-color surfaces enables detection by a model-based method that explicitly takes into account the variability of illumination.

1.2 Multicolor-based decoding

The second part of this thesis addresses the use of color to embed information in color barcodes. The necessity of increasing information density in a limited space led to the development of color barcodes. The use of color barcodes is gaining popularity as a pervasive technology to encode more information per area unit than regular black-and-white barcodes. There is surprisingly little research work in the literature on the topic of color barcodes. Especially, none of the previous works tried to model the changes of the observed color due to changing illuminant and blur-induced color mixing from neighboring barcode patches. The goal of color barcodes is to encode information with high spatial density while ensuring robust reading by a camera. The data capacity can be increased in a color barcode either by increasing the color patch density by using more patches in a barcode of a given size or by increasing the number of colors in a barcode. Both approaches have their limitations. Color patch density can only be increased up to a point prior to where adjacent patches can no longer be clearly distinguished without error due to the camera resolution. The main problem with using many colors in a barcode is that the color distribution of one color can overlap significantly with color distributions of other colors. Our contributions were motivated by improving the performance of a color barcode decoding under variety of illuminants and other nuisance parameters such as specularities and blur-induced color mixing from neighboring patches. Another goal of this work is development of new approaches to encode more information per area unit than existing color barcodes which may potentially lead to higher information rates per area unit.

1.2.1 Introduction to color barcodes

Barcodes have become tremendously popular as a means for information embedding technologies. The goal of a barcode is to encode information with high spatial density while ensuring robust reading by an optical system. Typical 1-D barcode technology use dark ink on a light-colored surface (or vice-versa); the resulting spatial pattern encodes the information [66]. 2-D barcodes encode data in both vertical and horizontal directions which increases significantly the data capacity of barcodes [2]. Using only two colors (e.g. black and white) with maximal color distance ensures robust

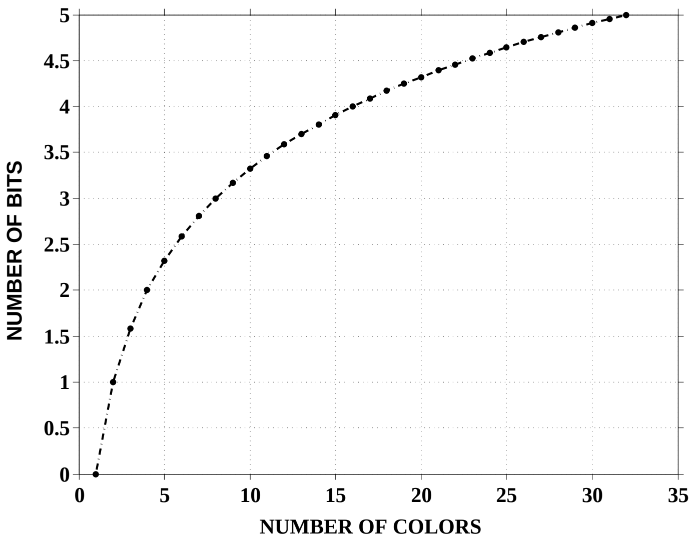


Figure 1.1: Number of Bits vs. Number of Colors.

decoding capability even with decoding devices that are of relatively low resolution. However, the information density of such barcodes is limited in a given space. In order to increase the density of information, different ink colors could be used to create color barcodes. Color barcodes increase data capacity without increasing the number of patches in a barcode. A color patch can represent more than one bit, depending on the number of colors used for encoding the data. When four different colors are used to encode information, one color patch can represent two bits. When eight different colors are used, three bits can be encoded in a patch. Using 16 colors instead of 4 would allow one to encode twice as many bits in the same area. The density of information (in bits per area unit) is proportional to $\log_2 N$, where N is the number of colors used. Thus, it would be desirable to use a large number of colors to embed more information in the same barcode area. Figure 1.1 illustrates the number of bits vs. number of colors.

There are several challenges using color. For instance, color is more susceptible to variations introduced by the printing device, the printing technology, the medium, such as paper, and its aging. The resulting color value may therefore differ from the color value that was printed by a printing device. The color value may change over time due to the changes in color tone, materials used to imprint symbols, smear or absorbance of moisture. In addition, the observed surface color depends not only on the surface reflectance spectrum, but also on the (unknown) illuminant spectrum, which represents a nuisance parameter. Consequently, determining the color index of each surface in the barcode is a challenging operation, especially if multiple light conditions (indoor/outdoor) are expected. Other nuisance parameters include: specularities; color drift during printing; unknown or poorly calibrated camera color response; camera non-linearity; color mixing from two nearby patches due to blur; quantization and noise. Most color barcode technologies overcome some of these difficulties by incorporation of a reference color palette within the barcode itself. It allows a comparison between the imaged patch colors and a set of reference colors so that robust color decoding is ensured. However, using many colors in a barcode makes the color indexing of barcode patches a challenging problem.

For example, Microsoft's High Capacity Color Barcode (HCCB) decoding method [53] clusters four or eight colors, and assigns each cluster to the closest color in the palette displayed in the barcode (Fig. 1.2). This clustering approach may not perform well if a large number of colors are used in the barcode system. One challenge facing color clustering methods using a large number of colors is that the color distri-

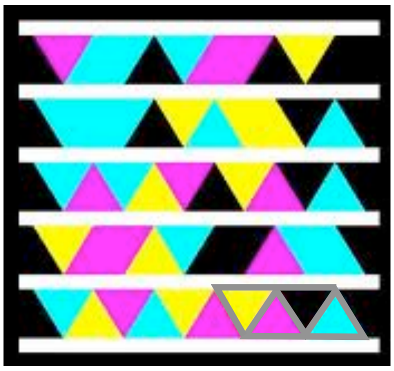


Figure 1.2: An example of HCCB code. The rightmost four patches in the last row (shown here with a gray border for display purpose) represent the palette of reference colors.

bution of one class can overlap significantly with color distributions of other classes. In addition, a palette of colors may ensure the robustness of the decoding, however, information density is reduced because a color palette with a large number of colors occupies the content space and cannot be used to encode information. In addition, this strategy only works for dense barcodes, with the number of patches largely exceeding the number of colors available, so that the addition of the color palette to the barcode has minimum impact on the spatial density of information. In addition of using a reference color palette, most 2D barcodes have error detection and correction capability [53] , which increase the robustness of decoding at the cost of reducing information density.

Another way to increase information density in a given space is a robust decoding method that allwos for a smaller patch size compared to methods using larger patch sizes to ensure the robustness of their decoding algorithm. For instance, in chapter 5 we describe a method that enables higher information density in a given space by allwoing smaller patch sizes using only three colors.

In conclusion of this section, we point out that we are not considering any form of channel coding to reduce the decoding error rate at the cost of increased redundancy. Channel coding, which is used commonly for barcodes, could certainly be implemented in the barcodes considered here as well. Our focus is only on algorithms for decoding the color information in a barcode. Other important issues such as detecting the presence of a barcode in an image or localizing each individual patch are outside the scope of this thesis.

1.2.2 Previous work

Perhaps the first reported attempt to use color in a 2-D barcode can be found in a patent by Han *et al.* [34], who used reference cells to provide standard colors for correct indexing. This technology, named ColorCode$^{\mathrm{TM}}$, is marketed by Colorzip Media (colorzip.com). A different type of color barcodes [58] is marketed by ImageId (imageid.com). Bulan *et al.* [14] proposed to embed data in two different printer colorant channels via halftone-dot orientation modulation. Grillo *et al.* [33] used 4 or 16 colors in a regular QR code. PM codes [69] use color to define layers, each of which makes up a 2-D barcode. Kato *et al.* [38] select colors that are maximally separated in a plane of the RGB color cube. The same type of color barcode (named MMCC) was used in a study the effect of JPEG compression on decoding [68]. Pei *et al.* used four colors in a color barcode technology named “Continuous Color Barcode Symbols” [55].

The HCCB decoding method [53] classifies N clusters in color space using mean shift clustering, and assigns each cluster center to one of the reference colors in a color palette, printed with the barcode.

Blasinski and et al [12] use a model-based interference cancellation procedure to recover the encoded data from each color channel. They propose a color interference cancellation algorithm that estimates the cyan (C), magenta (M), and yellow (Y) print colorant Channels from the Red (R), Green (G), and Blue (B) channels of the captured barcode. The encoded data is then extracted from each estimated colorant channel.

Grillo and and et al [33] introduced the High Capacity Colored Two Dimensional (HCC2D) QR-based code (Quick Response), a new 2D code which increases the code data density. HCC2D increases the storage space by using 4, 8, or 16 reference colors in the palette.

Querini and Italiano [56] investigated the decoding error rate on HCC2D color barcodes by using different classifiers such as Minimum Distance, Decision Trees, K-means, Naive Bayes, and Support Vector Machines (SVM) classifiers. They showed that K-means algorithm is the most effective classifier in their experiments.

None of these previous works tried to model the changes of the observed color due to changing illuminant, except for the patent of Sali and Lax [58], which uses a k-means classifier to assign the (R,G,B) value of a color patch to one reference color. Note that existing color constancy algorithms (e.g.[45, 30, 29, 27, 13]) are not of much use here. Classic color constancy assumes that neither the surface reflectance nor the illumination are known *a priori*, and aims to infer the surface colors under some specific scene hypotheses (e.g., gray world model, low-dimension illuminant/reflectance spectra).

In our case, we have full control over the selection of the surface colors that can be part of a barcode. This facilitates detection, but also poses the problem of which surface colors and color combinations are best suited to the task. Closest to our work is a recent paper by Wang and Manduchi [73], who studied the problem of information embedding via printed color. Their algorithm used one or more reference patches of known color, seen under the same illuminant as the color to be decoded. Observation of the reference patch(es) produces an estimate of a parametric color transformation between a canonical illuminant and the current illuminant, which is then used to decode the information-carrying color patches. The reference patches thus play a similar role to the color palette attached to the HCCB barcode, without the need to display all colors in the palette (usually one or two reference color patches suffice).

1.3 Organization of this thesis

In what follows in this thesis, we introduce a robust multicolor detection method and study several important problems in color barcode decoding that helps us provide fast and robust solutions.

- In Chapter 2, we introduce a detection algorithm that is specifically designed for multicolor, pie-shaped markers. We justify the use of colors we selected, and propose a detection algorithm that is computationally very light and has excellent performance in terms of detection and false alarm rate [6]. We also compare our system with a popular grayscale marker (ARToolKit), and show that our color markers enable more robust detection in various realistic conditions for a similar processing time.

- In Chapter 3, we introduce a new color barcode decoding approach that considers groups of k color surfaces that can be decoded under variety of illuminants, exploiting the fact that joint color changes can be represented by a low-dimensional subspace. Thus, decoding a group of color surfaces is equivalent to searching for the nearest subspace in a dataset. We show that by carefully selecting a subset of all possible k color surfaces, it is possible to achieve good information rate at low decoding error probability [7].

- In Chapter 4, we propose a significantly faster color barcode decoding method than the method described in chapter 3. This approach requires few reference colors attached to the color barcode and can handle the presence of specular reflection. We show that by carefully selecting a set of reference colors, it is possible to achieve a higher information rate than mainstream methods at a low probability of decoding error with relatively low computational complexity [8, 9].

- In Chapter 5, we present a new algorithm for color barcode decoding that can handle barcode images taken with a cell phone camera in which uniform blur across the barcode image can be present. In this work, rather than decoding individual patches or using a clustering method, our iterative algorithm decodes the colors of all patches across the barcode image by minimizing the overall observation error. We show that our method uses only three reference colors ensuring higher information density than mainstream approaches [10].

- In Chapter 6, we conclude the thesis and propose several possible directions for future work.

Part I

Multicolor Markers Detection

Chapter 2

Robust Real-time Detection of Multicolor Markers on a Cellphone

2.1 Introduction

Camera-equipped programmable cell phones have become the platform of choice for a wide variety of mobile computer vision applications, including augmented reality ([67]), gaming ([74]), mobile OCR (www.knfbreader.com), and barcode reading ([31]). Our work is motivated by a specific goal: helping a blind person to find their way around in a suitably equipped environment. Specifically, our system is based on simple 'markers', easily detectable by a cell phone, that can be placed in key locations in the environment. A blind person can search for such markers by orienting the camera phone in different directions, effectively 'scanning' the environment. Once a marker is detected by the camera phone, the user is prompted by an acoustic signal. If desired, the user can move towards the marker (which could be placed near a point of interest) by keeping track of the marker location via the camera phone. The marker may also contain a certain amount of information, for example in the form of an ID that can be used as a query to a locational database. In this way, the user could be provided with turn-by-turn instructions to reach a specific destination.

Our system uses multi-colored pie-shaped markers, specifically designed for fast recognition via mobile vision (see Fig. 2.1). Normally, color-based recognition requires some sort of color constancy operation to deal with varying and unknown illuminants ([35]). In our case, this is not necessary because the colors of the different surfaces in the marker are approximately co-variant with respect to changes in illumination. Since no pre-processing is necessary, our color-based detection algorithm is inherently fast. For added speed, a cascaded scheme is implemented. Most pixels are ruled out by the first stages of the cascade, which reduces the overall average computational cost. Further processing stages filter out any remaining false detections and compute the approximate distance to the marker (by measuring the amount of foreshortening).

The marker design was introduced for the first time in ([22]), along with a very simple detector and a post-processing (segmentation) algorithm ([23]). User studies with blind testers of this system have been reported in ([49]). In this contribution, we present a new marker detection algorithm, which is more efficient and accurate than previous approaches, while achieving high computational efficiency. For example, our system only needs to perform 1.1 multiplication and additions and 1.65 comparisons per pixel (on average) when searching for a color marker with 98% correct detection rate and

0.001% false positive rate. Note that the false alarms rate is then reduced further via geometry-based processing ([23]). On a Nokia N95 8GB cell phone processing images at VGA resolution, we achieve an effective frame rate of 8 fps (frames per second) when no marker is visible in the scene, which reduces to 5 fps (due to shape analysis) when a marker is visible. We also describe a simple technique to choose surfaces for the marker that have good Lambertian characteristics, and thus small specular reflection. Reducing the effect of specular reflection is critical for our color-based detection algorithm, as specular reflection may change the perceived color of a surface in a way that is difficult to model.

This chapter is organized as follows. We first review the related work in Sec. 2.2. Our approach to marker design is described in Sec. 2.3. We review the color rendering models used in our work (including linearization) in Sec. 2.4. Our detection algorithm is described in detail in Sec. 2.5, and experimental results are given in Sec. 3.4. An experimental comparison between our color marker and the popular ARToolKit marker is presented in Sec. 2.7. Sec 2.8 has the conclusions.

2.2 Related work

Image-based labeling has been used extensively for product tagging (1-D and 2-D barcodes) and for robotic positioning and navigation. The ubiquitous 1-D barcode (e.g. UPC and EAN) and 2-D barcode (e.g. Semacode and Shotcode) are designed so as to contain a high density of information. However, barcodes are not the ideal choice for environmental labeling when the goal is quick detection from various distances in possibly complex and cluttered environments.

A number of markers (or *fiducials*) have been proposed enabling fast and robust detection, for applications such as augmented reality and robotic localization. The majority of the markers described in the literature use black-and-white patterns (e.g., [40], [39], [59], [52], [57], [26], [18], [1]). In this case, detection relies on the shape properties of the pattern in the marker. Relatively little attention has been given to design concepts that rely on distinctive color, rather than (or in addition to) shape analysis (e.g. [17], [64], [43]). Unfortunately, these systems have been tested mostly in laboratory conditions, where one has the possibility to choose colors that are uniquely identifiable (possibly after color correction to account for different illuminants ([43]). Without careful color choice and proper processing, color marker detection is likely to fail in realistic conditions with multiple distractors and large variation in illumination. The color markers presented in this chapter use multiple colors to increase distinctiveness, and are detected by a model-based algorithm that takes explicitly into account the variability of illumination.

The use of color has also been proposed for increasing the capacity of 2-D bar codes. Examples include Microsofts High Capacity Color Barcode (HCCB) technology ([53]) and Colorzip Media's ColorCodes technology (www.colorzip.com).

2.3 Marker design

Our color markers are pie-shaped with four colored sectors. We briefly justify our design choice in the following; for an in-depth treatment, the reader is referred to ([22]). Marker detection is obtained by sliding a 'probe' across the image, where a probe is the set of four pixels at the vertices of a square (with fixed size of a few pixels). Fig. 2.1 shows a picture of our marker with a probe superimposed. When the probe is placed at or near the center of the image of a marker, each pixel in the probe records a color value from a different sector of the marker. A carefully designed classifier determines whether a given probe is on a marker or not. The advantage of the pie-shaped design is that the same probe size can be used regardless of the apparent (foreshortened) size of the marker (see Fig. 2.1). In addition, the probe does not need to be perfectly aligned with the marker for detection. Theoretically, a rotation of up to $\pm 45°$ around the camera's optical axes would still enable detection, as the points in the probe would still fit within the correct sectors. In practice, the system works well if the rotation is within $\pm 40°$ (see Fig. 2.1). Note that more color sectors would increase distinctiveness of the marker with respect to the background, and thus more reliable detection. However, this would affect the rotation invariance properties, as a smaller rotation angle would cause the probe to fall outside of the correct sectors. In addition, if the marker's real estate is divided into too many sectors, the number of pixels within each sector's image is reduced, which in turn reduces the maximum distance for detection due to foreshortening. Our four-color markers have a diameter of 15 cm and can be detected at a distance of about 4.5 m

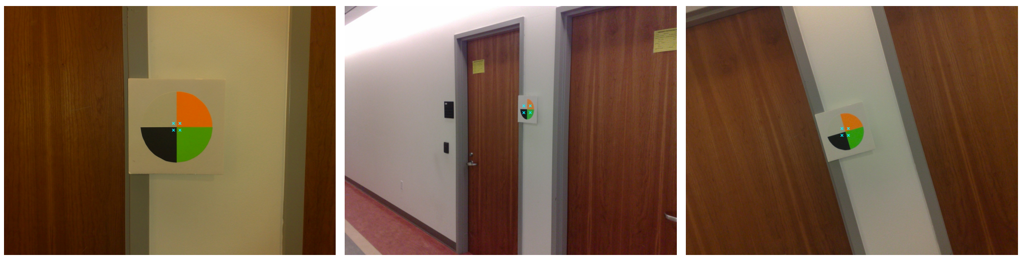


Figure 2.1: Our color marker, seen from different viewpoint and with different camera rotations. The blue superimposed crosses represent a fixed-size 'probe'.

using a Nokia N95 8GB cell phone (with a field of view of 45° by 35°) using images with VGA resolution (640 × 480). This marker's size and detection distance are acceptable for our application.

The four color values in the probe are fed into a classifier, which decides whether or not they are consistent with the image of a marker. This operation needs to be very fast, as it is performed at each pixel in the image. Further processing is then performed at all locations that passed this initial detection step. The largest cluster of nearby locations is extracted, and if its size is larger than a threshold, the whole marker image area is segmented out using a fast greedy region growing algorithm seeded with the cluster center. A number of shape consistency tests are then performed on the segmented region to reduce the risk of false alarms. Analysis of the segmentation mask enables robust identification of four non-collinear points of known position (see Fig. 2.2) which can be used to estimate the homography between the marker pattern and the camera image plane, and thus the camera pose. The spatial ordering of the colors in the marker can be permuted ([23]) providing a very simple way to embed information

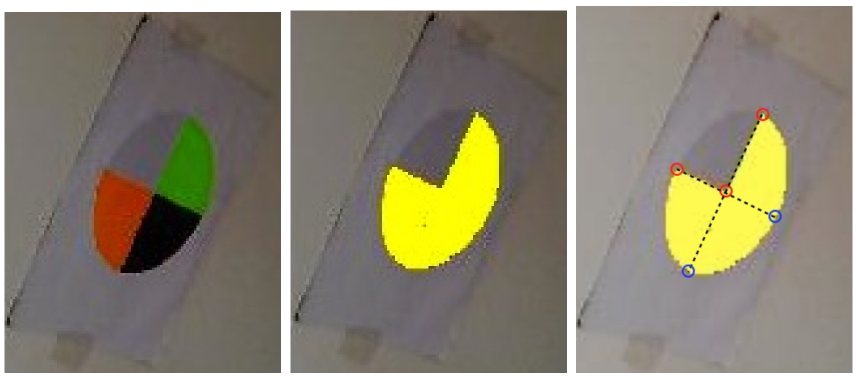


Figure 2.2: A color marker (left) and its segmentation (center) using a fast greedy region growing algorithm ([23]). Based on this segmentation, five points of known position can be identified (right). The points circled in red are detected as the three corners of the segmentation mask. By extending lines from the outer corners through the center corner, and computing their intersection with the mask's edge, two more points are identified (circled in blue). The location of the four outer points can be used to estimate the homography between the marker pattern and the camera image plane.

within the pattern itself (at an added computational cost - see Sec. 2.5.5).

At the core of the marker recognition algorithm is the initial color-based detection. It is critical that the detection rate be high, since a missed marker cannot be recovered by subsequent stages. The rate of false alarms must also be kept to a minimum. Even though the post-processing operations can rule out a good number of false alarms, these operations (especially the segmentation) are more computationally demanding , and can reduce the effective frame rate if too many false positives need to be examined. Our classification algorithm is introduced in Sec. 2.5; in the following, we describe our strategy for choosing the surfaces in the marker in such a way so as to facilitate robust recognition.

2.3.1 Choice of surfaces

In previous work ([22]) the markers were created using a color printer. The colors were chosen according to a criterion of maximum distinctiveness. Extensive experimentation with this marker revealed that the colored surfaces produced by a color printer, far from being Lambertian (matte), typically have a strong specular reflection component. This represents a serious problem for classification. In the presence of specular reflection, a viewpoint-dependent component with the same color as the illuminant adds to the perceived color of the surface from diffuse reflection ([60]). Classification thus becomes more challenging, as the specular color component needs to be removed.

In order to reduce the effect of specular reflection, we decided to create new markers as collages of colored paper with good characteristics of Lambertianity. We purchased 25 colored paper samples from various sources, all with good nominal opacity characteristics. Fifteen images of each individual paper were taken under the same illuminant from multiple viewpoints. The scatterplot of the normalized values $R/(R+G+B)$ vs. $G/(R+G+B)$ of the color of all surfaces is shown in Fig. 2.3. Note that, under an ideal Lambertian model, the normalized color for a given surface should be constant with respect to the viewpoint (as long as the illuminant does not change). In fact, the scatterplot shows that the normalized colors from most of the surfaces we tested are widely scattered. The point scatter within each color cluster is due in large part to the specular component, as a certain amount of illuminant color is recorded

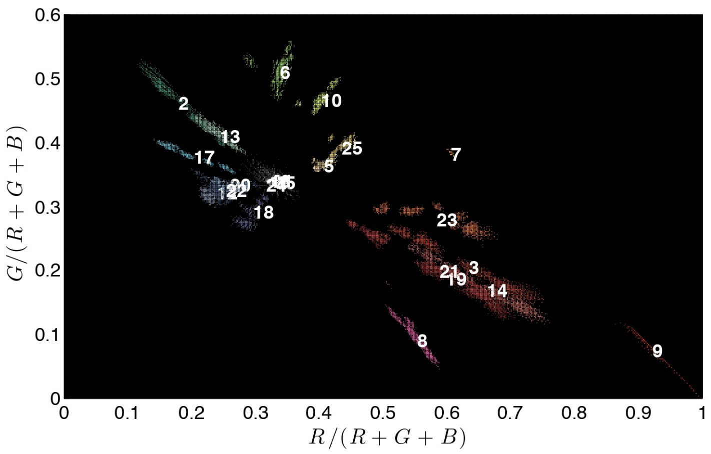


Figure 2.3: A scatterplot of the normalized color values of the considered surfaces (the surface's ID is printed in the center of each cluster). Images of the surfaces were taken under the same illuminants from 15 different viewpoints.

along with the surface color. In particular, most point clusters are oriented towards the cluster of the 'white' surface points, whose color is similar to the color of the illuminant.

This scatterplot allows us to quickly identify the surfaces with best Lambertian characteristics as the ones which produce the more compact point clusters. The compactness of a cluster can be measured, for example, by the sum or the product of the singular values of the collection of the normalized color values that form the cluster. Based on this analysis, we selected one orange surface (ID=7) which has an exceptionally compact point cluster in normalized color space (see Fig. 2.3).

We also selected a white and a black surface, following ([22]). The white surface is little sensitive to specular reflection (since the color of the diffuse and specular components are similar for a white surface). A white and a black surface nearby in the marker provide a strong albedo gradient that contributes to the marker's distinctiveness.

For the remaining patch, we selected a surface on the basis of a combination of criteria: compactness of the point cluster in normalized color space; albedo, measured by the sum of the R, G, and B values for a given illuminant (large albedo values result in higher SNR); distance in RGB space to the other selected colors (large distance means higher distinctiveness). We selected a green patch (ID=6) which provided a good trade-off based on these criteria. The resulting marker is shown in Fig. 2.1.

2.4 Color modeling

Our detection algorithm is based on a model of the perceived color of the four sectors in the marker under varying viewpoint and varying illuminant. We assume that the marker is on a flat surface, and that the illumination can be considered constant on the marker (i.e., that there are no shadow lines crossing it). In the following, we briefly review existing 'color rendering' models that can predict the color of a Lambertian surface when seen under different illuminants. We then describe our strategy for photometric camera calibration and linearization using a gamma model with bias. Finally, we show that the vectors formed by the colors in a probe live in a low-dimensional subspace. This notion is used for the design of the detector, presented in the next section.

2.4.1 Rendering model

Let us denote by $c_{(s)}^{(i)} = [c_{(s),1}^{(i)}, c_{(s),2}^{(i)}, c_{(s),3}^{(i)}]^T$ the recorded color of the surface of index s in the marker seen under an illuminant indexed by i, where $c_{(s),k}^{(i)}$ represents the

k-th color channel (R, G or B). As is well known, the recorded color $c_{(s)}^{(i)}$ changes with the illuminant i. These changes must be modeled and taken into account for the design of the color-based classifier. The model must consider the spectrum of the illuminant, the reflectance spectrum of the surface, the spectral sensitivity of the color filters in the camera, and the viewpoint and incidence angles. For Lambertian (opaque) surfaces, and assuming that the spectra of all possible illuminants and of all possible surface reflectances live in finite dimensional spaces (of dimension M_i and M_s respectively), the dependence of the surface color on the illuminant is described by the following quadratic form ([47],[71]):

$$c_{(s),k}^{(i)} = \alpha^{(i)T} Q_k \beta_{(s)} \tag{2.1}$$

where Q_k is an $M_i \times M_s$ matrix with positive entries that only depend on the camera and on any linear transformation of the color channels (e.g. white-point calibration); $\alpha^{(i)}$ only depends on the illuminant and on its incidence angle on the surface, as well as on the camera's exposure time and gain; and $\beta_{(s)}$ only depends on the surface. In particular, $\alpha^{(i)}$ and $\beta_{(s)}$ are independent of the color channel k.

It is well known ([28]) that if $M_s = 3$, the relationship between the color of a surface s when seen under two different illuminants can be expressed as follows:

$$c_{(s)}^{(i_2)} = A^{(i_1,i_2)} c_{(s)}^{(i_1)} \tag{2.2}$$

where $A^{(i_1,i_2)}$is a 3×3 matrix that is independent of the surface s. A further simplification is obtained by assuming that $A^{(i_1,i_2)}$ is diagonal (the so-called 'diagonal model'), in which case the color channels transform independently:

$$c^{(i_2)}_{(s),k} = a^{(i_1,i_2)}_k c^{(i_1)}_{(s),k} \tag{2.3}$$

where a_k is the k-th diagonal entry of matrix A. Likewise, under the diagonal model, the color values of two different surfaces s_1 and s_2 seen under the same illuminant i are related by a diagonal matrix that is independent of the illuminant:

$$c^{(i)}_{(s_1),k} = a_{(s_1,s_2),k} c^{(i)}_{(s_2),k} \tag{2.4}$$

Although the diagonal model (2.3,2.4) is extremely convenient in practice, it should only be considered as an approximation. In fact, it holds true only if each matrix Q_k has only one non–null column. [28] showed that when $M_i = 3$ and $M_s = 2$ (or $M_i = 2$ and $M_s = 3$), there exists an invertible linear color transformation such that the diagonal model holds exactly on the transformed colors.

2.4.2 Linearization

Ideally, the color values produced by the camera would be linearly related to the recorded color vector $c^{(i)}_{(s)}$ defined in the previous section. In practice, this relation is more complex due to several factors: color values are obtained via a color mosaic; noise adds to the measurements (including quantization noise); non-linear (e.g., gamma) correction compresses the measured values; and white-point correction applies independent rescaling of the color channels. The last two factors (non-linear correction and white-point correction) are dominant and need to be modeled and possibly compensated for before processing. Fortunately, most camera and camera cell phones on the market let the user set white-point calibration to a fixed value. As for the non-linear correction, this can be by-passed in some high-end cameras but not in cheaper cameras nor in most camera cell phones. Thus, in order to undo the undesired non-linear processing, it is necessary to "reverse engineer" the system via suitable calibration.

We modeled the non-linear correction performed by the camera as a biased gamma correction ([48], [25], [62]):

$$\bar{c}_k = b_k + c_k^{\gamma_k} \tag{2.5}$$

where $\bar{c}_k$ is the k-th channel color value at a given pixel produced by the camera via non-linear transformation of the recorded value c_k. (Note that any constant multiplicative coefficient can be considered to be already absorbed in c_k.) We assume that both the gamma coefficients $\{\gamma_k\}$ and the bias terms $\{b_k\}$ depend on the color channel.

In order to estimate $\{\gamma_k\}$ and $\{b_k\}$, we used a Macbeth color checkerboard with 24 color chips. The albedo value $v_{(s),k}$ of each color chip s in the checkerboard for each color channel k is known. Thus, under the diagonal model (2.4), one may expect the recorded color values $c_{(s),k}$ to be proportional to the albedos $v_{(s),k}$ for a given picture of the checkerboard. In order to increase the variety of sample data, we took pictures of the checkerboard under 8 different illuminant conditions. Note that with our camera cell phone (Nokia N95 8GB), we are unable to control the camera's exposure time and the gain, as they are set by the camera's exposure control. Thus, each picture of the checkerboard may have a different (and unknown) value of these parameters, which can be expressed as a constant multiplicative term $h^{(i)}$ (where we highlighted the dependence on the illuminant i, which determines the overall scene brightness and thus the parameters chosen by the automatic exposure control). In addition, due to the varying illuminant, each color channel may undergo a different linear transformation (multiplication by a factor $a_k^{(i)}$) according to the diagonal model (2.3). In summary, we model the k-th color channel value of the s-th color patch under the i-th illuminant as:

$$\bar{c}_{(s),k}^{(i)} = b_k + (h^{(i)} a_k^{(i)} v_{(s),k})^{\gamma_k} = b_k + g_k^{(i)} v_{(s),k}^{\gamma_k} \tag{2.6}$$

where all linear coefficients have been absorbed in $g_k^{(i)}$.

Thus, our calibration task can be expressed as follows: Given a set of measurements $\{\bar{c}_{(s),k}^{(i)}\}$ and known albedos $\{v_{(s),k}\}$, estimate $\{\gamma_k\}$ and $\{b_k\}$ based on (2.6). The measurements $\{\bar{c}_{(s),k}^{(i)}\}$ were obtained by averaging the color values over hand-drawn rectangles in the images of the color checkerboard.

We obtained a least-squares solution using Matlab's `fminsearch`() minimization function. As part of this procedure, we also estimated the coefficients $g_k^{(i)}$, which allowed us to plot the results of our calibration in Fig. 2.4. Note that these coefficients are irrelevant for our detection algorithm. The parameters found by our calibration procedure are: $\gamma_1 = 0.34$; $\gamma_2 = 0.24$; $\gamma_3 = 0.39$; $b_1 = -52$; $b_2 = -127$; $b_1 = -35$.

2.4.3 Dimensionality considerations

As mentioned in Sec. 2.3, the camera collects measurements of 'probes', where a probe is formed by the pixels at the vertices of a square of fixed side length. When a probe is centered at or near the center of the image of a marker, and the camera is

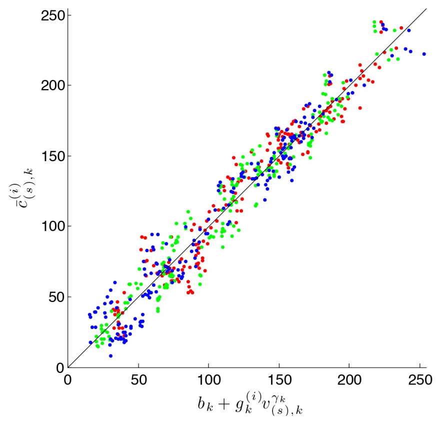


Figure 2.4: An illustration of our calibration results based on (2.6). The predicted values $b_k + g^{(i)}_k v^{\gamma_k}_{(s),k}$ (using the values for γ_k, b_k, and $g^{(i)}_k$ estimated by our procedure and the known albedo values of the color patches $v_{(s),k}$) are plotted against the values $\bar{c}^{(i)}_{(s),k}$ produced by the camera. Each dot represents a color patch under a certain illuminant. The dot's color (red, green, or blue) indicates the color channel (R, G or B) for that value.

correctly aligned, the probe contains color values of all surfaces in the marker. Let the 12-dimensional vector $p^{(i)}$ represent the concatenation of the three vectors (sub-probes) $p^{(i)}_k = [c^{(i)}_{(1),k}, \ldots, c^{(i)}_{(4),k}]^T$, where k from 1 to 3 represents the color channel, and let $\mathcal{S}$ be the linear space spanned by $p^{(i)}$ over varying illuminant i.

Fact 1. *Under the finite dimensional assumption (2.1), the following inequality hold:* $\dim(\mathcal{S}) \leq \min(M_i, 12)$.

Proof. From (2.1) it follows that $p^{(i)T}_k = \alpha^{(i)T} B_k$, with $B_k = [Q_k\beta_{(1)}| \ldots |Q_k\beta_{(4)}]$. Thus, a generic vector $p^{(i)}$ can be written as $B\alpha^{(i)}$ where $B = [B_1|B_2|B_3]^T$. Since the matrices Q_k have size $M_i \times M_s$, we conclude that $\dim(\mathcal{S}) = \text{rank } B = \min(M_i, 3 \cdot 4)$. $\square$

Note that the dimension of the space of illuminants M_i can be safely considered to be less than 12. For example, [50] showed that 3 basis functions were enough to model the color observation of 462 Munsell chips. It follows that the probes live in a subspace of dimension equal to M_i. In the case of the diagonal color model, one easily proves the following:

Fact 2. *If the diagonal color model (2.3) holds, then the space $\mathcal{S}$ can be expressed as the direct sum of $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$, where $\mathcal{S}_k$ is the one-dimensional subspace of $\mathcal{S}$ spanned by vectors $p_k^{(i)}$ associated to one color component only.*

Fig. 2.5, top, shows the sorted singular values of a matrix whose columns are the 12-D color probes extracted from 150 pictures of the color marker taken in a variety of indoor and outdoor illumination conditions and under different viewing angles (as described in detail in Sec. 2.6.1). Any probe that contained a color value larger than 245 was considered saturated and removed from the data set used to create this plot. All values were linearized by inverting the gamma transformation (2.5). It is seen that the first singular value is much larger than the others. This phenomenon is typical, as the first eigenvector represents the variation in intensity, which dominates other sources of variability. The remaining singular values decrease fairly smoothly, and thus do not provide a strong indication for the dimensionality of $\mathcal{S}$. Looking at the three subspaces $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ (Fig. 2.5, bottom), spanned by the sub-probes with only one color channel, one again notices a dominant eigenvector in all three cases. In the following, we will assume for simplicity's sake that the diagonal model holds and therefore that the subspaces $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ are one-dimensional.

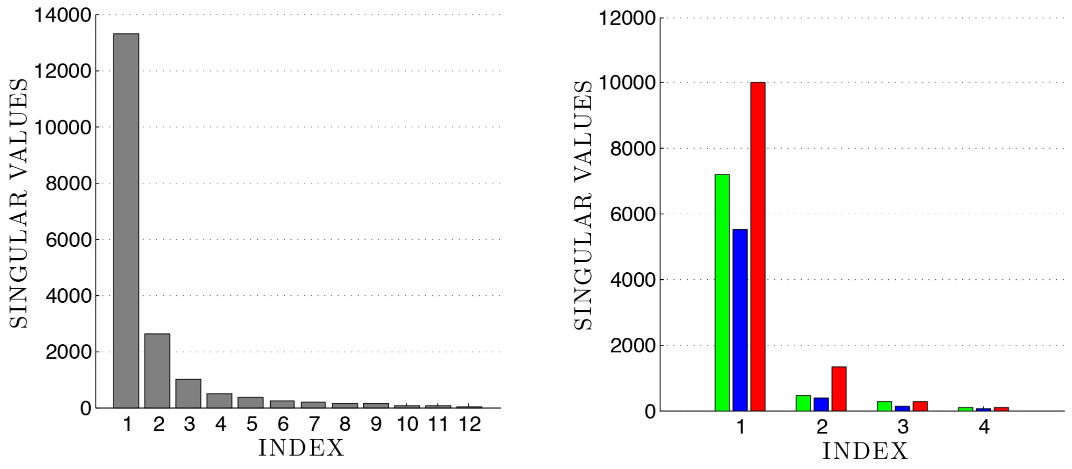


Figure 2.5: Sorted singular values of the 12-D color probes (top) and of the 4-D sub-probes with only one color channel (bottom). The color of the bars indicates the color channel of the sub-probe.

2.5 Detection algorithm

The design of the marker detection algorithm requires careful consideration of several factors. Our application calls for robust detection with a very low rate of misses and false alarms under a wide variety of viewing conditions and of background. At the same time, the algorithm needs to be very light, so as to enable analysis of several frames per second at reasonably high resolution when implemented on a cell phone. In the following, we describe our design choices vis-a-vis these application requirements.

Our detection algorithm is based on a one–class classifier model: it analyzes a probe to determine whether or not it may belong to the image of a marker, without explicitly modeling the 'background' (non-marker) class. The reason for choosing a one-class classifier is that it would be very difficult to produce a general model of the background that can apply to any environment. The typical approach of collecting representative images of the background may not generalize well to previously unseen situations. Fortunately, as discussed earlier, the distribution of color values of the probes is well structured, as the 12-dimensional probe color vectors actually live in a much lower dimensional space. This allows us to use a relatively simple classifier, which is implemented in a cascaded structure ([72]) for improved efficiency.

2.5.1 Classifier design

Since the color probes are expected to live in a subspace $\mathcal{S}$ of R^{12}, a simple classifier could be obtained by thresholding the Euclidean distance of a probe p to such subspace. However, even this simple operation turns out to be too computationally expensive for real-time implementation on our cell phone with the desired resolution (640×480 pixels). We can reduce the computational complexity as follows. Firstly, we assume that the diagonal color model (2.3) holds. Based on Fact 2, we note that the square of the distance $d(p, \mathcal{S})$ of the probe p to the subspace $\mathcal{S}$ is equal to the sum of the squared distances of p to $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$ respectively. Accordingly, the distance-based classifier declares a detection when:

$$\sum_{k=1}^{3} d(p_k, \mathcal{S}_k)^2 < \bar{\tau}^2 \tag{2.7}$$

where $\bar{\tau}$ is a suitable threshold, and p_k is the vector formed by the four values in the probe for the k-th channel (for the sake of notational simplicity, we neglect to indicate the dependence on the illuminant i in the following). Additionally, rather than setting a threshold on the sum of squared distances to the $\{\mathcal{S}_k\}$, we set a threshold $\tau = \bar{\tau}/3$ on the *maximum* of such distances. In other words, we declare a detection when:

$$\max_k d(p_k, \mathcal{S}_k) < \tau \tag{2.8}$$

The advantage of this detector is that it can be implemented as a cascade of three tests, each involving computation of $d(p, \mathcal{S}_k)$ and comparison with a constant.

If q_k is the unit vector originating the one-dimensional subspace $\mathcal{S}_k$, then

$$d(p_k, \mathcal{S}_k)^2 = p_k^T p_k - (p_k^T q_k)^2 \tag{2.9}$$

This operation requires 9 multiplications and 7 additions per pixel, which is still too demanding for our real-time implementation. In order to reduce the computational cost further, we introduce the following procedure. We begin by observing that, given any two surface types (s_1, s_2) in the color marker, the following inequalities hold:

$$d(p_k, \mathcal{S}_k) \geq d(p_{(s_1,s_2),k}, \mathcal{S}_{(s_1,s_2),k}) \tag{2.10}$$

$$d(p_k, \mathcal{S}_k)^2 \leq \sum_{s_1=1}^{3} \sum_{s_2=s_1+1}^{4} d(p_{(s_1,s_2),k}, \mathcal{S}_{(s_1,s_2),k})^2 \tag{2.11}$$

where $p_{(s_1,s_2),k}$ contains the k-th color channel for surfaces s_1 and s_2, and $\mathcal{S}_{(s_1,s_2),k}$ is the (one-dimensional) projection of $\mathcal{S}_k$ onto the plane $P_{(s_1,s_2)}$ spanned by the vectors $p_{(s_1,s_2),k}$ for varying i. These inequalities show that a *necessary* condition for $d(p_k, \mathcal{S}_k) < \tau$ to hold is that, for all surface pairs (s_1, s_2),

$$d(p_{(s_1,s_2),k}, \mathcal{S}_{(s_1,s_2),k}) < \tau \tag{2.12}$$

while a *sufficient* condition is that, for all surface pairs (s_1, s_2),

$$d(p_{(s_1,s_2),k}, \mathcal{S}_{(s_1,s_2),k}) < \tau/\sqrt{6} \tag{2.13}$$

as there are six terms in the sum in (2.11). This suggests the use of a cascaded implementation with six *elemental* classifiers, each computing $d(p_{(s_1,s_2),k}, \mathcal{S}_{(s_1,s_2),k})$ for a choice of (s_1, s_2). In practice, each elemental classifier examines whether $p_{(s_1,s_2),k}$ lies within a strip in the plane $P_{(s_1,s_2),k}$ (see Fig. 2.11), where the strip is parallel to $q_{(s_1,s_2),k}$, the projection of q_k onto $P_{(s_1,s_2),k}$. A fast cascaded implementation of each elemental classifiers can be derived by observing that (2.13) is satisfied when both of these inequalities are satisfied:

$$\begin{aligned} p_{(s_2),k} - a_{(s_1,s_2),k} p_{(s_1),k} &< \hat{\tau} \\ p_{(s_2),k} - a_{(s_1,s_2),k} p_{(s_1),k} &> -\hat{\tau} \end{aligned} \tag{2.14}$$

where $a_{(s_1,s_2),k} = q_{(s_2),k}/q_{(s_1),k}$ and
$\hat{\tau} = (\tau/\sqrt{6})/\cos(\arctan a_{(s_1,s_2),k})$. Thus, each individual classifier in $P_{(s_1,s_2),k}$ requires one multiplication, one addition, and one or two comparisons. In fact, as we discuss below, we don't use a single threshold but two distinct thresholds, $\hat{\tau}_1$ and $\hat{\tau}_2$, that are learned from training data. The computation cost remains the same whether the same or different thresholds are used.

The vectors $\{q_k\}$ is computed via SVD from training data. As for the thresholds $\hat{\tau}_1$ and $\hat{\tau}_2$, we could use a simple criterion: choose the smallest values that ensure correct detection of all probes in the training data. In practice, this means expanding the 'strip' on either side of $q_{(s_1,s_2),k}$ until all training probes $p_{(s_1,s_2),k}$ are contained in the strip. This ensures that all training data is correctly classified ([15]), ([22]). In our experiments, we noted that this choice is often too conservative, in which case we can multiply both thresholds by a constant *margin ratio (MR)* coefficient.

It is instructive to compare these elemental detectors with those originally proposed by [22], which simply compared $p_{(s_2),k} - p_{(s_1),k}$ with a threshold τ. This is equivalent to declaring detection when a sub-probe lies in a 'detection sector', formed by all points $p_{(s_1,s_2),k}$ that are above (or below) a 45° line with intercept at $p_{(s_2),k} = \tau$. Clearly, this detector is less selective than the newly proposed one. In terms of computational cost, our new method requires one multiplication and up to one comparison more per pixel per elemental detector with respect to the original detector ([22]).

2.5.2 Computational cost

Since there are six permutations of the four surfaces in the marker taken two at a time, and three color channels, the total number of elemental (cascaded) classifiers is eighteen. In order for a probe to be classified as a candidate marker, it must pass all eighteen tests. Assuming statistical independence of the tests, it is well known that the overall probability of (correct) detection is

$$P_D = \prod_{i=1}^{18} P_{D_i} \tag{2.15}$$

where P_{D_i} is the probability of detection for the i-th elemental classifier. Likewise, the overall probability of false alarm is equal to

$$P_F = \prod_{i=1}^{18} P_{F_i} \tag{2.16}$$

The expected number of operations (multiplications or additions) for a non-marker probe is

$$N_{\text{ops}} = 1 + \sum_{i=2}^{18} \prod_{j=1}^{i-1} P_{F_i} \tag{2.17}$$

As for the number of comparisons in the case of a non-marker probe, one can reason as follows. Suppose that color values of a non-marker probe are uniformly distributed in the space outside the 'detection strip' defined by (2.13), and that the classification strip is oriented approximately at 45 degrees. The first test in (2.14) checks whether the probe is above the detection strip (with probability of 0.5 of finding it there). If this is not the case, the second test in (2.14) checks if it is below the strip. The average number of comparisons is thus $1.5 \cdot N_{\text{ops}}$. This number should be taken as an upper bound: if the detection strip is at a different angle, the order of the tests in (2.14) can be chosen so that the first test detects a non-probe marker with probability larger than 0.5, thus reducing the average number of tests.

It is clear that the order of the elemental classifiers critically affects the average computational cost N_{ops} ([16]). The maximum efficiency is obtained when the classifiers are ordered according to their false alarm rate P_{F_i}, with the first classifiers removing most of the false alarms.

2.5.3 Dealing with saturated point

When a color value is saturated, our linear model no longer applies. The effect of saturation to the color distribution can be seen clearly in the scatterplots of Fig. 2.11. In order to deal with saturated points, we considered three possible strategies.

The first strategy is to identify those color values that are saturated, and change the classification rule for those points. This requires an additional number of comparisons for each pixel, thus increasing the computational cost. In our experiments, this resulted in an effective frame rate that was too low for our application. The second approach is to simply neglect the presence of saturation, and treat saturated pixels and unsaturated pixels alike. The third approach we considered is to remove saturated samples from the training data before computing the eigenvectors $\{q_k\}$. This helps ensuring that the slope of the stripe in the $P_{(s_1,s_2),k}$ plane is not biased by the saturated pixels. However, we consider all training samples when computing the thresholds $\hat{\tau}_1$ and $\hat{\tau}_2$. This is necessary as we require that all training samples are correctly detected. The resulting classifier is then applied on all new samples, regardless of whether they are saturated or not.

2.5.4 The advantage of gamma correction

The detector design introduced in Sec. 2.5.1 is based on the distance between the probe vector p and the subspace $\mathcal{S}$ in R^{12} where probe vectors are assumed to live. This is a simple and powerful approach, with one major pitfall. Since the subspace $\mathcal{S}$ contains the origin, any very dark probe (with color values close to zero) will be classified as a marker. Indeed, this was the single major cause of false positives in our experiments.

A simple fix could be to isolate very dark probes (via suitable thresholding) to avoid that they be mistakenly classified as markers. Note in passing that our marker contains surfaces with high albedo (except for the black one) so it is unlikely that all surfaces would produce very low color values. Unfortunately, our experiments have shown that choosing a correct threshold is very difficult, leading to the risk of missing a marker due to poor exposure.

We have found another (somewhat unexpected) solution by considering the gamma-corrected data produced by the camera, rather than the linearized data. Indeed, as we elaborate below, the linear color rendering model applies with only a small modification to the gamma-corrected data, and this modification is key to an improved algorithm with much reduced false alarms.

According to the diagonal model, the values $p_{(s_1),k}$ and $p_{(s_2),k}$ of a probe are linearly related (2.4):

$$p_{(s_2),k} = a_{(s1,s2),k} p_{(s_1),k} \tag{2.18}$$

The gamma-corrected versions of $p_{(s_1),k}$ and $p_{(s_2),k}$ (i.e., the values produced by the camera) are, according to (2.5):

$$\bar{p}_{(s_1),k} = b_k + p_{(s_1),k}^{\gamma_k} \ , \ \bar{p}_{(s_2),k} = b_k + p_{(s_2),k}^{\gamma_k} \tag{2.19}$$

Combining (2.18) and (2.19) one obtains:

$$\bar{p}_{(s_2),k} = b_k(1 - a_{(s1,s2),k}^{\gamma}) + a_{(s1,s2),k}^{\gamma} \bar{p}_{(s_1),k} \tag{2.20}$$

Hence, the gamma-corrected values satisfy a linear equation with non-null intercept (except for the case

$a_{(s1,s2),k} = 1$, in which the intercept is null). An example with $a_{(s1,s2),k} = 4$ using the values for γ_1 and b_1 found in Sec. 2.4.2, is shown in Fig. 2.6. This suggests a simple modification to our algorithm, which allows it to work with gamma-corrected data. We first note that, since the (non-linearized) probes $\bar{p}_k$ span a line that does not necessarily intersect the origin, our previous approach to characterize this line by the dominant eigenvector of the probe data would fail. We can correct for this by simply removing the mean of the probes $\{\bar{p}_k\}$ before eigenvector analysis. Since the mean is supposed to lie on the line spanned by the $\bar{p}_k$, the dominant eigenvector q_k of the zero-mean data now reliably characterizes the line. Projection of q_k onto planes $P_{(s_1,s_2)}$ gives the slope $\bar{a}_{(s1,s2),k}$ of the line where points $\bar{p}_{(s_1,s_2),k}$ are supposed to live. Then, similarly to the previous case, a strip is expanded on either side of this line until all training points are contained in the strip. Note that this classification region is structurally identical to the case considered in Sec. 2.5.1, except that now it need not contain the origin. It is exactly this characteristic that allows this approach to substantially reduce the rate of false positives, as shown by the experimental results described in the next section.

2.5.5 Encoding information via color permutation

It is possible to encode a few bits of information within the marker by simply permuting the position of the color patches (with the permutation index representing the marker ID). Since there are 4!=24 permutations of the four colors, the information content is $\log_2 24 = 4.6$ bits. Note that this form of information embedding comes at no "area cost" - the overall marker area remains the same. Of course, more information could be embedded by adding other patterns (e.g. 2-D bar codes) near the color markers.

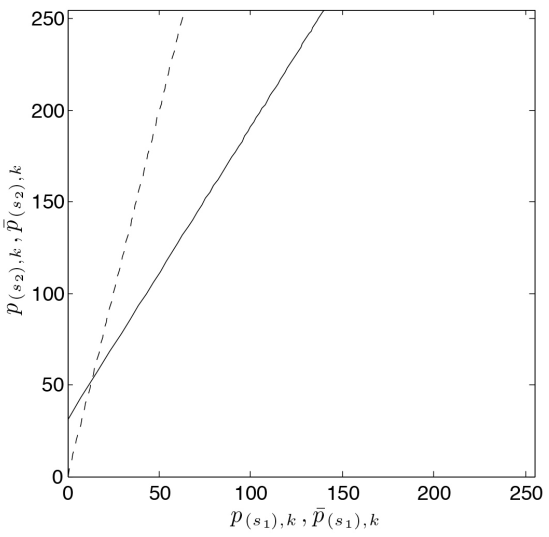


Figure 2.6: The dashed line shows the relationship $p_{(s_2),k} = a_{(s1,s2),k} p_{(s_1),k}$ with $a_{(s1,s2),k} = 4$. The solid line shows the relationship between the gamma-corrected (non-linearized) values $\bar{p}_{(s_1),k}$ and $\bar{p}_{(s_2),k}$.

In this case, the marker would be used simply as a distinctive fiducial, allowing for quick and reliable identification of the pattern location.

A disadvantage of this simple approach is that it increases the computational cost of marker detection. Whereas in the single-ID case (in which color patches have a fixed position) detection is obtained via a cascade of tests, each involving two patches (s_1, s_2) and one color channel (k), now the first step involves testing all possible permutation of surfaces taken two at a time for the same color channel k (in total, 12 tests). Each patch pair that passes the first test "fixes" the position of two patches in the permutation. Computing the expected number of operations for non-marker probes is difficult. Empirically, we have observed an increase in the number of operations by a factor of 16 when considering all possible color permutations (see Table 2.1). We should emphasize that this increased computational cost is incurred only when the marker's ID is unknown. If one is searching for a specific ID (that is, if the color permutation is known), then the computational cost is the same as described in Sec. 2.5.2.

2.6 Experiments

2.6.1 Data sets

We collected 150 images, taken with the Nokia N95 8GB phone, of the marker under different conditions of illumination (both indoors and outdoors), from different viewing angles, and from different distances. Each image was hand-labeled. More precisely, a 15×15 square was drawn on each color sector, and a list was created with 25 pixels picked from each square. Then, probes were built by scanning the lists for the four squares in parallel, taking the color values for each point in each list. Thus, our marker training set contains $150 \cdot 25 = 3,750$ probes. Note that we don't low-pass filter the training data (nor the test data). Although low-pass filtering would help removing noise, its computational cost would reduce the effective frame rate. In addition, the blur generated by a low-pass filter could potentially corrupt the color values within the marker when the marker image is small (because taken from a distance).

We also took 5 different images (indoors and outdoors) of various 'background', that were used to estimate the false alarm rates. (Note that the classifier is designed only based on the marker images.) These background images, shown in Fig. 2.7, were

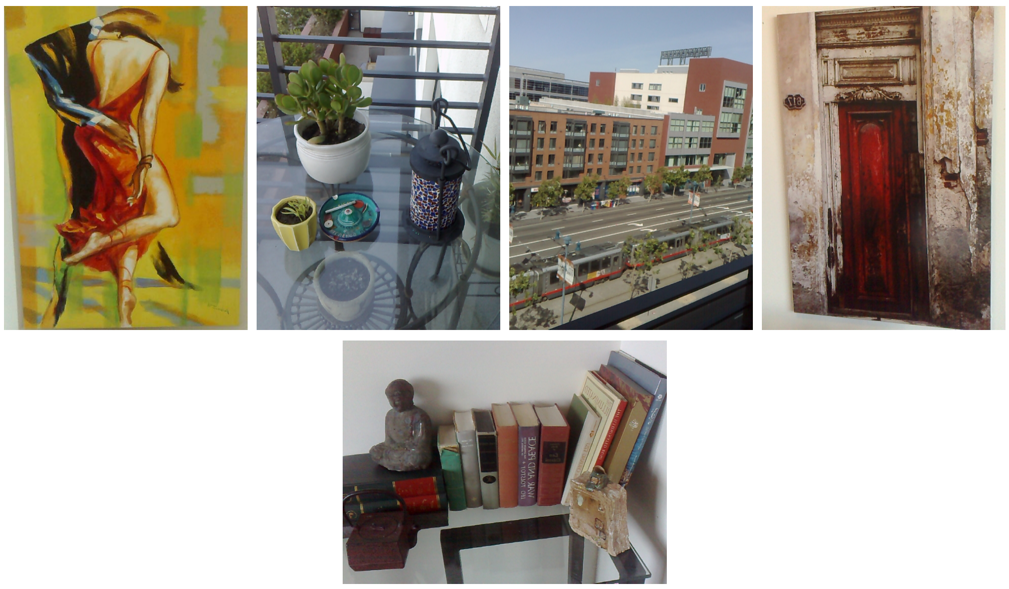


Figure 2.7: The background image data set.

scanned with a probe with width of 12 pixels. In total, we collected 163,020 samples of background data.

Fig. 2.8 show the sorted false positive rates P_F for the different elemental detectors, trained on all marker data and tested on the background images. These detectors were designed on the gamma-corrected data $\bar{p}_{(s_1,s_2}, k)$ after mean removal (Sec. 2.5.4) and without removing saturated pixels for the computation of the $\{q_k\}$. The margin rate MR was set to 1.

Figs. 2.11 and 2.12 show the scatterplots of the original gamma-corrected ($\bar{p}_{(s_1,s_2),k}$) and linearized ($p_{(s_1,s_2),k}$) probes for different choices of the surfaces s_1 and s_2 and for different color channels k. The plots are ordered according to the false positive rates P_F of the elemental filters in Fig. 2.8. In each figure we show the 'classification strip', that is, the region in the $P_{(s_1,s_2),k}$ plane where a probe is classified as a marker

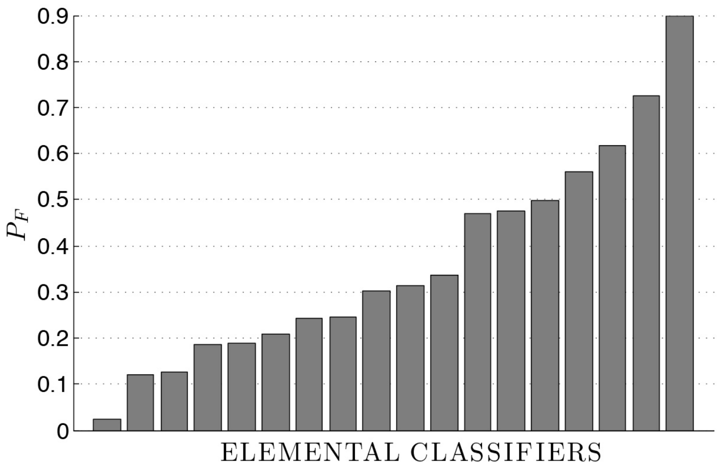


Figure 2.8: The sorted false positive rate P_F for the different elemental detectors operating on the gamma-corrected training data. The detectors were designed after mean removal (Sec. 2.5.4) and without removing saturated pixels for the computation of the $\{q_k\}$. The margin rate MR was set to 1.

by an elemental classifier. The solid lines identify the classification strip for the case in which all samples are used for training, while the dashed lines represent the case in which saturated points are removed before computing the eigenvectors $\{q_k\}$ (see Sec. 2.5.3). Note that in several cases, the classification strips for the original gamma-corrected probes $\bar{p}_{(s_1,s_2),k}$ does not contain the origin.

2.6.2 Performance evaluation

In order to test our system, we performed a number of cross-validation experiments. At each experiment n, half of the marker images were chosen at random. The detector was trained on such images, and then tested on the data from the remaining marker images to establish the detection rate $P_D(n)$, as well on the 'background' data to compute the false alarm rate $P_F(n)$. The results of 30 such experiments were then averaged together. Note that even though the classifier is guaranteed to produce $P_D(n) = 1$ for the data it was trained on, it is still prone to false negatives for the remaining data. Fig. 2.9, top, illustrates the results in the form of 'pseudo-ROC' curves. Each pseudo-ROC curve is obtained by plotting P_F against P_D for varying 'cascade length', where the cascade length is the number of elemental detectors in the cascade. More precisely, we ordered the 18 elemental detectors (designed with margin rate MR set to 1) according to increasing value of their false alarm rate. Then, we removed detectors from the tail of the cascade to create cascaded detectors with different length. It should be clear from (2.15), (2.16) and (2.17) that reducing the number of elemental detectors increases P_D as well as P_F while reducing the computational cost N_{ops} (see also Fig. 2.9, bottom). These curves can be useful to choose the correct cascade length if the application at hand sets specific requirements in terms of P_D, P_F or N_{ops}.

The different pseudo-ROC curves in Fig. 2.9 correspond to different choices of design parameters:

- Whether training (and testing) was performed on the original, gamma-corrected data or on the linearized data (sec. 2.5.4);
- Whether saturated data was removed before computing the slope of the strips in the elemental detectors or all trained data was used (see Sec. 2.5.3);
- Whether or not the eigenvectors $\{q_k\}$ originating $\mathcal{S}_k$ were computed by first removing the mean of the training probes (see Sec. 2.5.4).

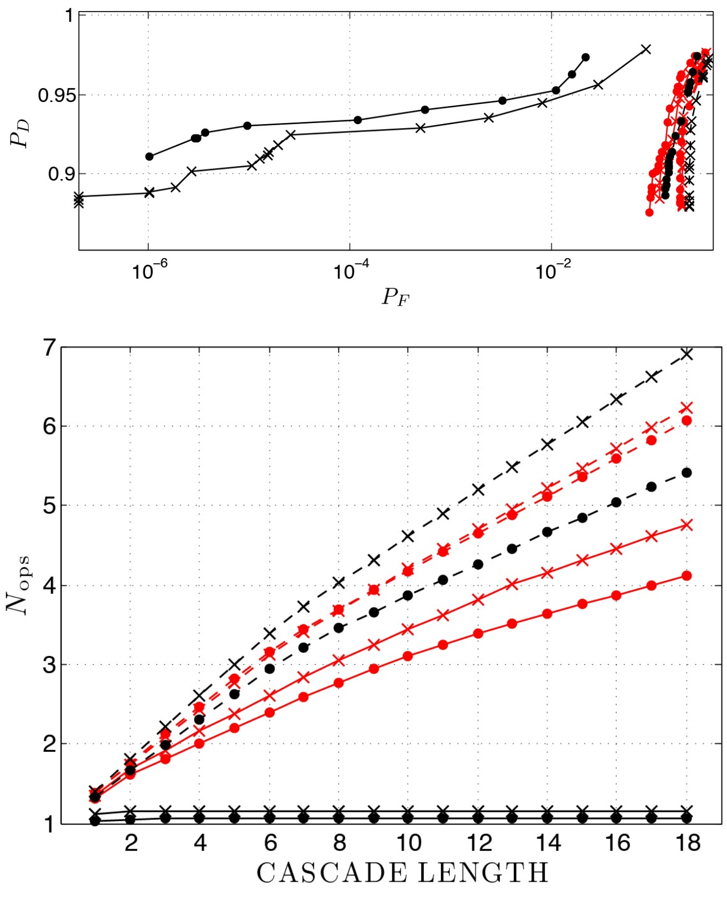


Figure 2.9: Top: Pseudo-ROC curves for different design parameters. Bottom: Expected number N_{ops} of multiplication for input pixel when a marker is not visible as a function of the cascade length. The values for N_{ops} were computed using the actual values of P_D and P_F for different cascade lengths rather than their approximation (2.15)–(2.17). Each marker in the curve represents a different cascade length. Solid line: Original gamma-corrected data. Dashed line: Linearized data. Red line: Data mean not removed before computing the eigenvectors $\{q_k\}$. Black line: Data mean removed. Circles: Saturated points not removed before computing the eigenvectors $\{q_k\}$. Crosses: Saturated points removed. The margin rate MR was set to 1 for these experiments.

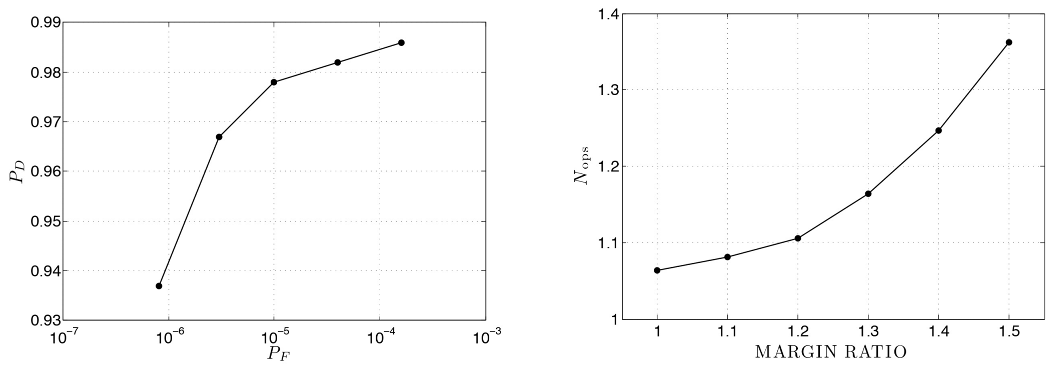


Figure 2.10: Left: ROC curve for a detector operating on the original gamma-corrected data (data mean removed and saturated points not removed before computing the eigenvectors $\{q_k\}$). Right: Expected number N_{ops} of multiplication for input pixel when a marker is not visible. The different points in the curves correspond to different margin ratios *MR*.

From Fig. 2.9 it results clear that using the original gamma-corrected data and removing the mean of the training probes before computing the eigenvectors $\{q_k\}$ produces the best results. Other choices of parameters have large false alarm rates. A single elemental detector with the lowest false positive rate (the first one shown in Fig. 2.11, comparing data from the probe corresponding to the orange and the black surface in the red channel) has $P_D = 0.97$ and $P_F = 0.02$. Increasing the cascade length reduces both P_D and P_F, as expected.

Even with the best choice of parameters, it is seen that these results are not satisfactory. For example, a false positive rate P_F of 10^{-5} is achieved only at the cost of a relatively low detection rate ($P_D = 0.93$). As noted earlier, it is important to keep the false alarm rate low even if subsequent post-processing may remove remaining false alarms. For example, if $P_F = 10^{-5}$, a VGA-sized image without a marker will generate about 3 false alarms on average, which then need to be processed further. The detection performance can be improved by increasing the margin ratio *MR*. Fig. 2.10, top, shows the ROC curve for a detector operating on the original gamma-corrected data, with data mean removed, and saturated points not removed before computing the eigenvectors $\{q_k\}$ (corresponding to the best-performing pseudo-ROC curve in Fig. 2.9). This curve is obtained by increasing the margin ratio from 1 to 1.5. Note that now the detection rate P_D increased to 0.98 for $P_F = 10^{-5}$ with *MR*=1.2. The corresponding computational cost per pixel is of $N_{\text{ops}} = 1.1$ multiplications and additions, and $1.5 \cdot N_{\text{ops}} = 1.65$ comparisons.

We implemented the cascaded detection algorithm on the Nokia N95 8GB, programmed in C under the Symbian OS 9.6 S60. This cell phone is equipped with an ARM 11 332 MHz processor with 128 MB of RAM and 8GB of Flash memory. The images are processed at full VGA resolution. The effective frame rate is about 8 frames per second (fps) when no marker is present. Due to post-processing (including segmentation), the frame rate reduces to 5 fps when a marker is present. We tested the system extensively as a wayfinding tool for persons who are blind ([49] - see Fig. 2.13).

2.7 Comparison with ARToolKit

We benchmarked the performance of our color marker system against a popular marker, the ARToolKit ([40, 39, 1]) that does not use color information. The ARToolKit marker has been used extensively for Augmented Reality (AR) applications. It consists

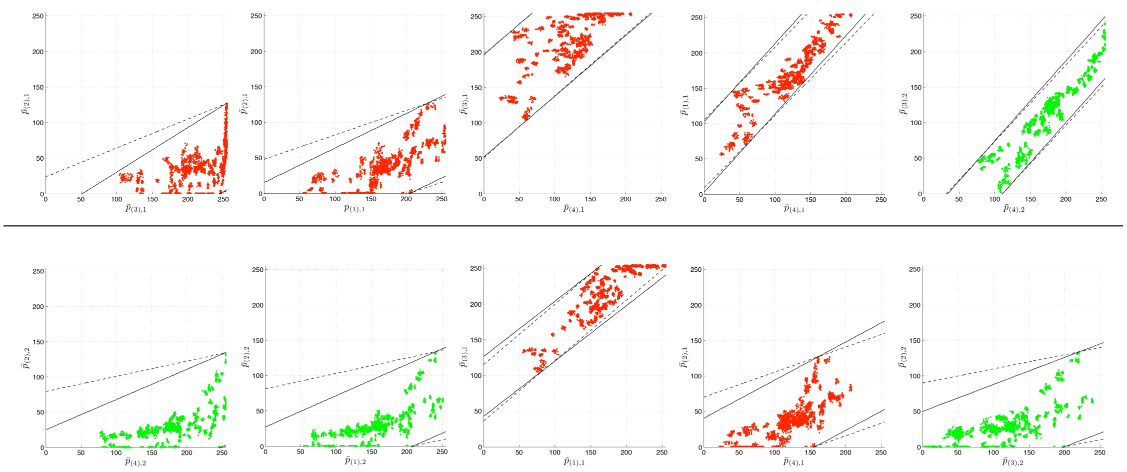


Figure 2.11: The first ten scatterplots of color probe points from our training data, restricted to the planes $P_{(s_1,s_2),k}$. The gamma-corrected data $(\bar{p}_{(s_1),k}, \bar{p}_{(s_2),k})$ are shown on top of the linearized data $(p_{(s_1),k}, p_{(s_2),k})$. The scatterplots are ordered according to the increasing false positive rate P_F of the elemental detectors as shown in Fig. 2.8. The 'classification strips' are shown with different line types depending on whether saturated points were removed before computing the eigenvectors $\{q_k\}$ (dashed line) or not (solid line). $s = 1$: White surface. $s = 2$: Black surface. $s = 3$: Orange surface. $s = 4$: Green surface. $k = 1$: Red channel. $k = 2$: Green channel. $k = 3$: Blue channel.

of a square black border encircling a grayscale pattern that contains the ID of the marker. An improved version, the ARTag [26], still uses the square black border but replaces its interior with a digital pattern of 36 bits.

For our comparative study, we used the open source implementation of the ARToolkit Library for Windows maintained by the University of Washington[1]. The detection algorithm works by first binarizing a greyscale image using a fixed threshold.

[1]www.hitl.washington.edu/artoolkit

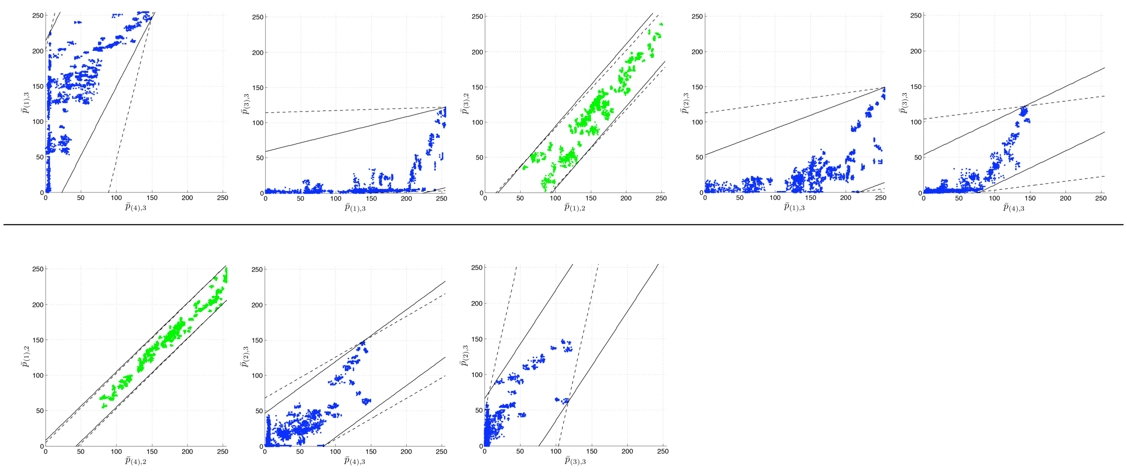


Figure 2.12: The last eight scatterplots of color probe points from our training data, restricted to the planes $P_{(s_1,s_2),k}$ (see caption of Fig. 2.11).

The connected components of the binarized image are then computed, and their edges and corners are extracted. The vertices of the detected outer black square are used to compute the homography mapping the square onto its image in the camera. (Note that, as shown in Fig. 2.2, a similar operation can be performed with our color marker, by relying on the segmentation described in Sec. 2.3.) The interior of the black square border is then analyzed to extract the marker's ID.

In our experiments, we considered various realistic situations including different viewing distances and angles, illumination conditions, motion blur, and occlusions. Note that we are interested in the *detection* performance, and not in the ID computation. Hence, our results are only in terms of *detection rate*: we never checked whether the ID computed by the ARToolKit algorithm was correct or not. In each test, a color marker and an ARToolKit marker were placed side by side on a vertical surface, and

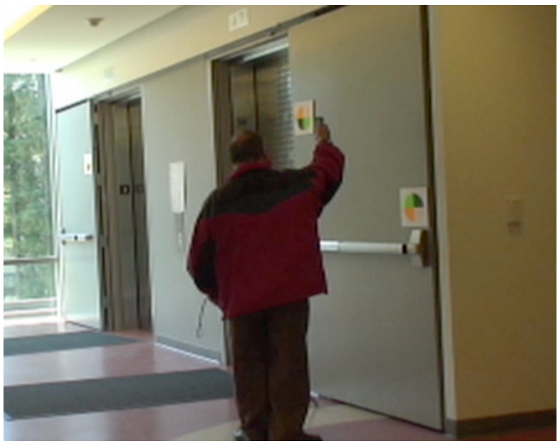


Figure 2.13: Example of use of our color marker as a wayfinding system for blind persons ([49]).

images were taken by a Nokia N95 cell phone (at 640 × 480 resolution) for further processing on a laptop computer. This solution allowed us to compare the two algorithms on the same computing platform. The diameter of the color marker (15 cm) was set to be equal to the side length of the ARToolKit marker. There was never more than one color marker and one ARToolKit marker visible in the same image.

The detection rates for the various experiments described in the following are shown in Table 2.1, along with the average computational time (on the laptop) per frame for both types of markers.

2.7.1 Experiments

2.7.1.1 Multiple camera placements

In this experiment, images of the marker pair were taken from angles of 0, 30 and 60 degrees (with respect to the normal to the markers' surface), at 6 equispaced distances from 0.6 to 3.6 meters. Two different illumination conditions were considered. The camera locations and sample images are shown in Fig. 2.14. Under the first illuminant, the color marker was detected from any location, while the ARToolKit was not detected at distances beyond 2.4 meters. Under the second illuminant, the color marker was detected in all but one location. The ARToolKit marker was not detected in 4 locations.

2.7.1.2 Motion blur - Bright light

24 images were taken of the marker pairs under a bright illuminant from various distances, while the camera was moving. Camera motion should always be expected with mobile vision applications; these experiments are meant to study the robustness of the detection algorithms under the ensuing motion blur. The color marker was detected all 24 times, while the ARToolKit marker was detected 20 times. Examples are shown in Fig. 2.15.

2.7.1.3 Motion blur - Dim light

In this case, 58 images of the markers under dim light were taken while the camera was moving. In low light conditions the camera is forced to increase exposure time and sensor gain. This gives rise to motion blur and noise, both clearly noticeable in the examples of Fig. 2.16. Under this challenging condition, the color marker was detected 47 times, while the ARToolKit marker was never detected, due to the fact that the fixed threshold was too high for correct binarization.

2.7.1.4 Occlusions

7 images were taken with both markers being partly occluded by a surface. This situation may occur, for example, when one is searching for a marker in a crowded scene, with other persons impeding view of part of the marker. The color marker was detected in all but one case (in which it was actually detected, but segmentation was not successful - see Fig. 2.17). The ARToolKit marker was detected only once in these experiments.

2.7.1.5 False positives

No instances of false positives were recorded using the color marker, even when the background contained a variety of colors. Sporadic episodes of false positives occurred with the ARToolKit marker (see e.g. Fig. 2.18).

2.7.1.6 Processing time

The average processing times for the two algorithms, computed on the laptop computer used for the experiments (Intel Pentium Dual CPU T3200 at 2 GHz with 3 GB RAM) for images with size of 640×480 pixels, are shown in Table 2.1. Two different versions of the color marker detection algorithm were implemented: one in which a specific ID marker was searched for, and one that considered all possible 24 color permutations as described in Sec. 2.5.5.

For the single-ID color marker, the processing time is lower than for the ARToolKit detection when there are no markers visible in the scene. When a marker is visible and detected, the color marker takes a slightly larger computational time. Note that these computational times include segmentation (described in Sec. 2.3). When a marker is detected, segmentation accounts for about 16% of the computational cost.

The situation is very different when the marker ID is not known in advance. In this case, as explained in Sec. 2.5.5, a much larger number of tests is required for each pixel, leading to an increase in processing time by a factor of 16.

2.7.2 Discussion

The basic conclusion from these experiments is that, for the same marker surface area, color markers can be detected far more robustly and at a wider range of distances than ARToolKit markers. The detection speed is comparable in the two cases when a specific ID marker is searched for. However, if all 24 marker IDs are considered during search using the color permutation technique of Sec. 2.5.5, color markers require a much larger (16 times) computational time than ARToolKit markers. A careful comparative analysis of the two marker types and detection algorithm can shed light on these performance differences.

Detection of an ARToolKit marker hinges on successful binarization of the marker's outer edge. Since the marker's outer edge is black on a white background, binarization is in most cases attainable using a fixed threshold. The ability to use a fixed threshold is vital for computational efficiency, a factor that is especially important with power-constrained mobile platforms and when high image resolution is used. Unfortunately, as seen in the experiments with dim ambient light (Sec. 2.7.1.3), a fixed threshold may lead to gross errors in some situations. Adaptive binarization ([70]) would most likely improve results, but at a heavier computational cost. This is, in fact, one of the principal advantage of using color markers: robust detection of carefully chosen color patterns is achievable with very few operations per pixel under any illuminant.

The outer black border of the ARToolKit marker needs to be resolved at a high enough resolution to enable geometric analysis. This places a heavy constraint on the maximum distance at which the marker can be detected. An advantage of the color marker is that it does not require geometric processing for detection: as long as the probe is contained in the marker's image (see Fig. 2.1), detection can be achieved.

The reason for the dramatic increase of computational time when the color marker's ID is not known in advance is that ID identification for permuted-color markers is embedded in the search process at the pixel level. In the case of ARToolKit markers, the ID information is inside the marker, while detection only uses the outside border. A similar solution would be impossible with our color marker: the whole surface of the color marker must be used for the color patches, since the probe's vertices may fall on different points of the marker's image depending on the viewing distance. It should be noted, however, that the maximum distance at which the pattern inside an ARToolKit marker can be decoded is likely to be smaller than the maximum distance at which the outer border of the marker can be detected, and that the decoding process is likely to be affected by motion blur (see e.g Fig. 2.15).

Different solutions for embedding ID information in a color marker could be considered, such as using a color or grayscale pattern in an outer ring around the marker.

	DETECTION RATE	
	Color Marker	ARToolKit Marker
Various placements Illuminant 1	**18/18**	12/18
Various placements Illuminant 2	**17/18**	14/18
Motion blur Bright light	**24/24**	20/24
Motion blur Dim light	**47/58**	0/58
Partially occluded	**6/7**	1/7

	PROCESSING TIME (ms)	
	Color Marker	ARToolKit Marker
Individual ID No visible markers	**2.6**	3.2
Individual ID Visible markers	3.9	**3.7**
Multiple IDs No visible markers	58.9	**3.2**
Multiple IDs Visible markers	58.9	**3.7**

Table 2.1: Comparative results in terms of detection rate and processing time for the tests considered in Sec. 2.7.

In particular, it was shown recently that up to 7 bits of information can be embedded reliably within a single color patch ([73]). Thus, a selected set of just a few color patches around the marker could convey enough ID information for most practical purposes.

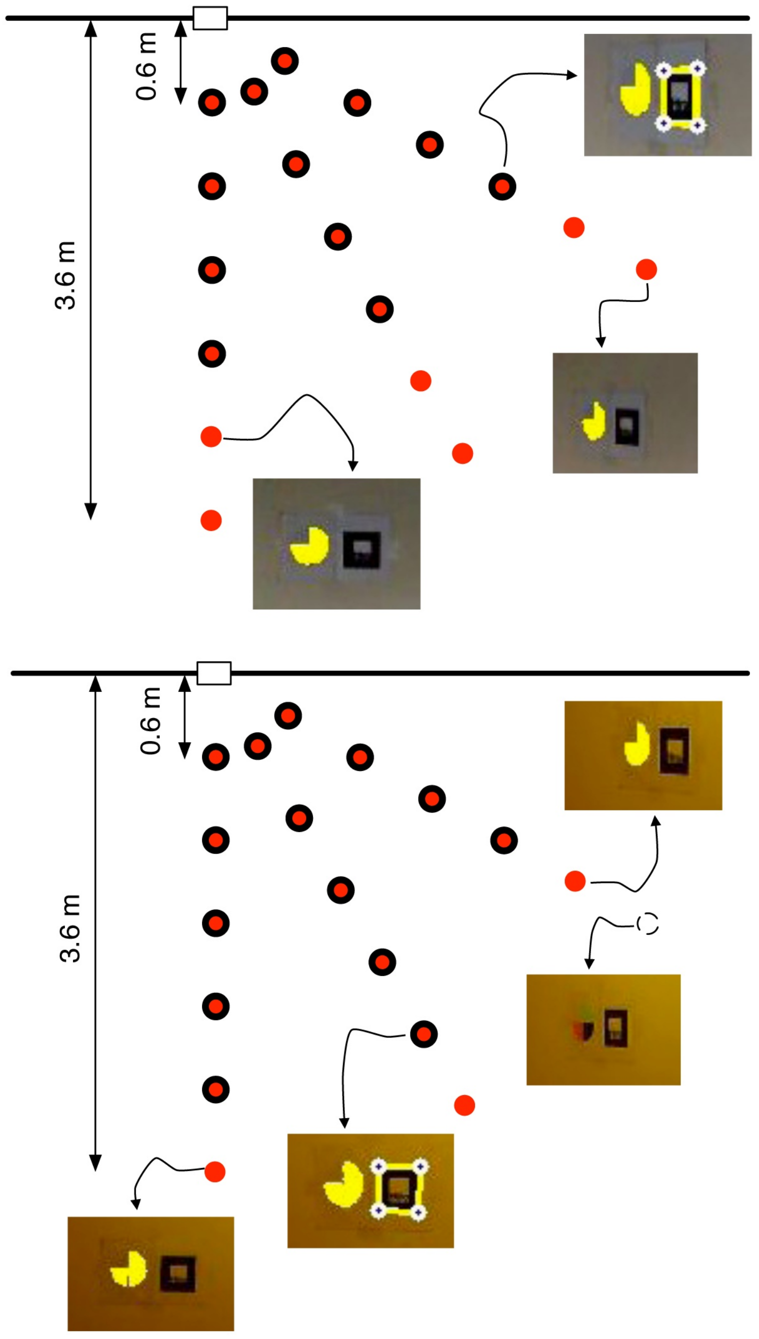


Figure 2.14: Comparative detection experiments using the color marker and the ARToolKit marker for two different illuminants (Sec. 2.7.1.1). The diameter of the color marker (set to 15 cm) was equal to the side length of the ARToolKit marker. The markers were placed side by side on a wall, at a location shown by a small rectangle in the bird-eye view. Images were taken with a cell phone camera placed in the locations shown by the circles. Locations in which the color marker was successfully detected are marked in red. A thick black border indicates successful detection of the ARToolKit marker. The image crop-outs show samples of correct and missed detection. Successful detection is indicated by the yellow area on a color marker (which shows the result of segmentation, as described in Sec. 2.3) or by the yellow edges on a ARToolKit marker.

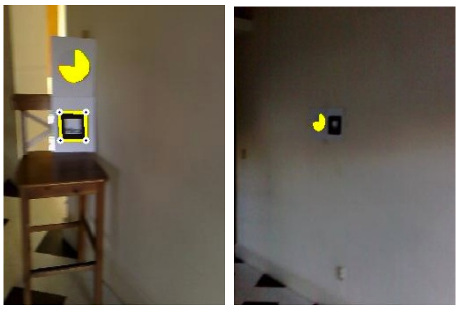


Figure 2.15: Correct and missed detection examples for the *Motion blur - Bright light* experiments (Sec. 2.7.1.2).

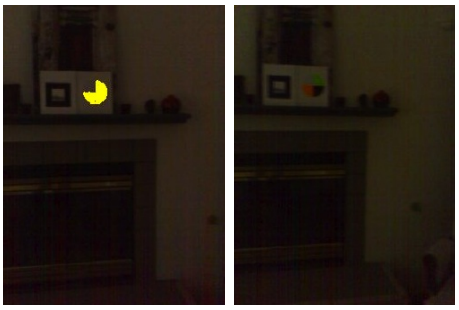


Figure 2.16: Correct and missed detection examples for the *Motion blur - Dim light* experiments (Sec. 2.7.1.3).

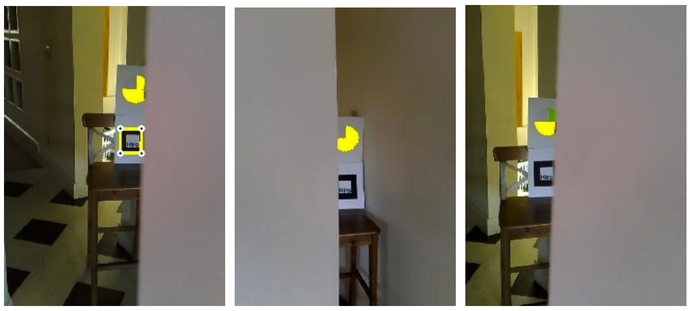


Figure 2.17: Correct and missed detection examples for the *Occlusions* experiments (Sec. 2.7.1.4).

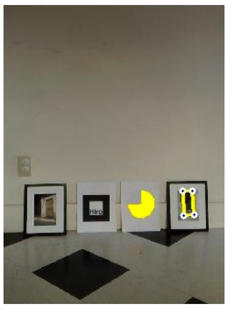


Figure 2.18: An example of false positive and missed detection for the ARToolKit marker.

2.8 Conclusions

We introduced a new detection algorithm that is suitable for multi-color, pie-shaped markers. This is a crucial component in a cell phone-based system that uses environmental labeling for blind wayfinding. The proposed algorithm is very light and has excellent performance in terms of detection rate and false alarm rate. The algorithm is implemented as a cascade of elemental detectors, each one of which operates on only two color values from a probe in the same color channel. The elemental detectors are derived based on a diagonal rendering model. One interesting result of our study (for which we provide formal justification) is that using the original, gamma-corrected data gives better results than using linearized data. In addition, we have introduced a very simple method for selecting surfaces for our color markers that have good Lambertian characteristics, and thus minimize the risk of mis-detection due to specular reflection.

Compared with a popular grayscale marker (ARToolKit), our color markers enable more robust detection in various realistic conditions for a similar processing time. However, the modality used by the ARToolKit (and other similar marker such as the

ARTag) for embedding ID information allows for faster decoding than the simple approach of color permutation proposed for the color markers. We are currently exploring new strategies for encoding ID information using a grayscale or color pattern at the outer edge of the color marker.

Part II

Color Information Decoding

Chapter 3

Subspace-based Color Barcode Elements Decoding

3.1 Introduction

This contribution takes a different approach than mainstream methods, and considers color barcodes that can be decoded under multiple illuminants without the need to display a color palette or reference colors. Our strategy is to consider groups of k color patches (with $k \geq 2$), and model their evolution due to changing illuminant using a low-dimensional linear subspace. Thus, each group of k colors (a *barcode element*) is represented by one such subspace. When a group of k color patches is observed, our algorithm does not attempt to decode each color independently. Instead, the whole group is decoded by finding the nearest subspace in a dataset. We show experimentally that this algorithm relatively enables information rate (average number of bits per bar) with very low probability of incorrect detection. In particular, our results show that this algorithm has the potential to convey more information for the same area than technology such as HCCB that requires display of the reference colors for correct decoding. Our study borrows the idea of statistical modeling of joint color changes from the work on color eigenflows by Miller and Tieu [51]. However, rather than trying to represent the variation of *all* printable colors as a function of illuminant, we concentrate on the variation of small groups of printed colors.

3.2 Information Rate

We assume that the patches in a color barcode are built from a set $\mathcal{C}_N$ of N *reference patches*[1] A length k *barcode element* is an ordered set of k reference patches extracted from $\mathcal{C}_N$. For reasons discussed in Sec. 3.4, we assume that the patches in a barcode element are selected without replacement (i.e., all color patches in a barcode element are different from each other). We also assume that only a subset $\mathcal{B}_{k,x}$ of the set of all possible length k barcode elements $\mathcal{B}_k$ can be used to build a barcode, where x denotes the proportion of elements of $\mathcal{B}_k$ in $\mathcal{B}_{k,x}$ (with $0 < x \leq 1$).

A *barcode* is the juxtaposition (in any spatial pattern) of n barcode elements, resulting in $K = nk$ bars. The *information rate* R of a barcode[2] (measured in bits per bar) is defined by the logarithm base 2 of the number of different symbol that can be

[1]Note that the words "patch" and "bar" are used interchangeably in this chapter to mean a region with uniform color.

[2]Note that in communication theory, "information rate" usually represents the average entropy per symbol. Our definition assumes that all symbols are equally likely.

represented by the barcode, divided by the number of bars:

$$R = \frac{1}{k} \log_2 \frac{xN!}{(N-k)!} \tag{3.1}$$

Note that if $k \ll N$, the following approximation holds:

$$R \approx \log_2 N + \frac{\log_2 x}{k} \tag{3.2}$$

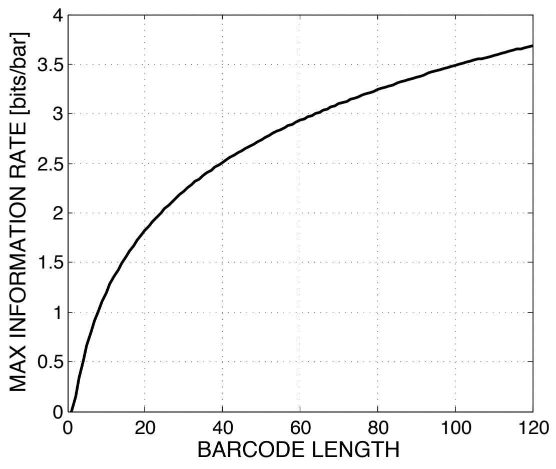


Figure 3.1: The maximum information rate $\bar{R}_{\max}(K)$ for a barcode that displays its reference colors as a function of the barcode length K.

The goal of a barcode decoder is to infer the index of each observed barcode element in $\mathcal{B}_{k,x}$. The mainstream approach to color barcode decoding (e.g. [34, 53]) assumes that N patches in the barcode are reserved to display the reference colors. This allows for faithful decoding by comparing the color of the patches in the barcode against the displayed reference colors. However, this solution comes at the cost of reduced information rate, since the N reference patches cannot be used to encode information.

The information rate in this case is (assuming that each patch is decoded independently):

$$\bar{R}(K,N) = \left(1 - \frac{N}{K}\right)\log_2 N \tag{3.3}$$

It is instructive to compute the maximum information rate achievable with this system as a function of the barcode length K (shown in Fig. 3.1 for $K \leq 120$):

$$\bar{R}_{\max}(K) = \max_N \bar{R}(K,N) \tag{3.4}$$

For example, the maximum information rate of a length 60 barcode that displays its reference colors is of about 3 bits/bar, meaning that this barcode cannot carry more than 180 bits. This value of information rate is obtained for N=16 reference colors. Increasing the number of reference colors decreases the information rate for this barcode length, as the gain due to the higher number of symbols that can be represented by each bar is undermined by the fact that fewer patches are available for carrying information.

Along with the information rate, it is important to consider the error rate. Let $P_E(k,x)$ be the probability of decoding error (incorrect identification) of a generic barcode element in $\mathcal{B}_{k,x}$. The probability of decoding error for the whole barcode, assuming that decoding errors for the individual barcode elements in the barcode are statistically independent events, is:

$$P_E(K,k,x) = 1 - (1 - P_E(k,x))^{K/k} \tag{3.5}$$

Note that, though a convenient working hypothesis, the assumption of independent decoding error may not hold true in all situations, and should be tested experimentally.

Some qualitative considerations can be drawn from (3.1) and (3.5). The information rate R grows linearly with $\log_2 N$ (as long as $k \ll N$) and with $\log_2 x$. Increasing

k increases the information rate (note that the second term in the r.h.s. of (3.1) is negative), and this increase is all the more noticeable for small values of x. For example, if $x = 0.02$, then increasing k from 3 to 4 adds almost 0.5 bits per bar. For what concerns the probability of incorrect decoding, it grows with the number of bars K in the barcode. The dependence of $P_E(k, x)$ on k and x is more complex, and is the object of the work described in the next sections.

3.3 Barcode Element Decoding

3.3.1 The Dimensionality of Joint Color Spaces

Decoding a barcode element means finding its index in $\mathcal{B}_{k,x}$ based on the observed colors $\mathbf{c} = [c_1, \ldots, c_k]^T$ of its patches, where $c_i = [c_i^R, c_i^G, c_i^B]^T$ is an (R,G,B) color vector. As well known, a variation of the illuminant spectrum determines a variation of the perceived colors. A popular model to describe the observed color of a Lambertian surface [45] assumes that the spectra of the surface reflectances and of the illuminants live in finite-dimensional spaces of dimension N_{ref} and N_{ill} respectively. Thus, the observed color of a surface s under a given illuminant is equal to

$$c(v) = \Phi_s v \tag{3.6}$$

where v is a vector of length N_{ill} containing the coefficients of the illuminant with respect to the chosen basis, and Φ_s is a full-rank $3 \times N_{\text{ill}}$ matrix whose entries are a function of the illumination and reflectance basis vectors as well as of the spectral sensitivities of the camera. Note that, since $N_{\text{ill}} \geq 3$ in general, the rank of Φ_s is 3, making the decoding of an individual color c hopeless without some prior knowledge of the scene or of the illuminant. If, however, *multiple* colors seen under the same illuminant are decoded at once, the task is less daunting. For example, consider the vector $\mathbf{c}(v)$ formed by the colors in the barcode elements as defined above. Then $\mathbf{c}(v) = \Phi v$ with

$$\Phi^T = [\Phi_1^T | \dots | \Phi_k^T] \tag{3.7}$$

where

$$\text{rank}(\Phi) = \min(3k, N_{\text{ill}}) \tag{3.8}$$

Hence, the vector $\mathbf{c} \in \mathbb{R}^{3k}$ is constrained to live in a subspace $\mathcal{S}$ of dimension of at most N_{ill}. This observation is critical for our decoding algorithm, as discussed next.

3.3.2 Joint Subspace Generation

Our decoding algorithm begins by modeling the subspaces $\mathcal{S}(i)$ of all length k barcode elements. The use of linear subspaces as class models is based on the assumption that the color vector $\mathbf{c}$ distribution in each barcode element class lies approximately on a lower-dimensional subspace. We have considered two approaches to build these subspace. In the first case, each barcode element is seen under a wide variety of illuminants, and a subspace of suitable dimension M is built from these observations via Principal Component Analysis. In practice, one only needs to take images under multiple illuminants of the N reference patches, build vectors $\mathbf{c}$ from these images for each barcode element, and compute the SVD of the resulting matrix. This procedure produces accurate subspace modeling; in practice, however, it may be unwieldy, because each subspace depends on the color patches as well as on the characteristics of the camera used to observe them. This means that the whole training procedure (including taking pictures of the patches under different illuminants) would have to be repeated each time a different camera is used or the colors are printed with a different printer. We thus consider a second approach to subspace modeling, which, albeit less accurate, enables a much simple training procedure. This approach relies on the diagonal (or Von Kries) model of color changes [28], which assumes that each color channel changes as a result of an illuminant change by a multiplicative factor that depends on the illuminant but not on the reflectant:

$$c_s(v_2) = D_{v_1 \to v_2} c_s(v_1) \tag{3.9}$$

where $D_{v_1 \to v_2}$ is a diagonal matrix. It is easy to see that, in this case, $c(v) = \Phi v$ with

$$\Phi^T = \begin{bmatrix} c_1^R & 0 & 0 & c_2^R & 0 & 0 & c_3^R & \dots \\ 0 & c_1^G & 0 & 0 & c_2^G & 0 & 0 & \dots \\ 0 & 0 & c_1^B & 0 & 0 & c_2^B & 0 & \dots \end{bmatrix} \tag{3.10}$$

The three columns of Φ form an orthogonal basis of the subspace. In practice, calibrating the algorithm for a new camera or a new set of reference colors can be performed simply by taking a single picture of all N reference patches *under any illuminant.* This allows the system to immediately generate a basis for each barcode element by building the corresponding matrix Φ.

3.3.3 Nearest Subspace Decoding

Given the observed color vector $\mathbf{c}$ of a barcode element, we decode it by assigning it to the subspace $\mathcal{S}(i)$ that has the minimum distance to $\mathbf{c}$ (where the distance of $\mathbf{c}$ to $\mathcal{S}(i)$ is defined in the usual way by the Euclidean distance between $\mathbf{c}$ and its projection onto $\mathcal{S}(i)$). Nearest subspace search is a common technique in Computer Vision. It was shown by Basri *et al.* [11] that it is possible to map subspace $\mathcal{S}$ and query item $\mathbf{c}$ to points in $\mathbb{R}^{d'}$ for some d', in such a way that the Euclidean distance between these two points increases monotonically with the distance of $\mathbf{c}$ to $\mathcal{S}$, thereby enabling the use standard nearest neighborhood techniques (e.g. k-d trees) for barcode element decoding.

3.3.4 Barcode Elements Subset Selection

Nearest subspace decoding produces a certain probability of error P_E, defined as the average probability of decoding error over all barcode elements. Note that P_E contributes to the probability of decoding error for a barcode formed by n barcode elements as by (3.5). If P_E is larger than desired, one may reduce the cardinality of the subspace elements by only selecting a subset $\mathcal{B}_{k,x}$ of $\mathcal{B}_k$. Intuitively, a smaller set provides fewer opportunities for misclassification, at the cost of reduced information rate.

Selection of a proportion x of barcode elements that minimizes the associated P_E is computationally very expensive. In particular, if one barcode element is removed from $\mathcal{B}_k$, the new empirical probability of error needs to be recomputed for all barcode elements. We have considered several techniques to reduce the complexity associated with subset selection. One possible approach is to use subspace distance as an indicator of the probability that two barcodes could be confused with each other. We adopt the following definition of distance between two subspaces $\mathcal{S}_1$ and $\mathcal{S}_2$ with dimension d_1 and d_2, respectively [65]:

$$\mathrm{dist}^2(\mathcal{S}_1, \mathcal{S}_2) = \max(d_1, d_2) - \|\Phi_1^T \Phi_2\|_F \tag{3.11}$$

where Φ_1, Φ_2 are any orthonormal basis matrices of $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively, and $\|\cdot\|_F$ represents the Frobenius norm. Fig. 3.2 shows the effect of pairwise subspace distance on the probability of error. More precisely, we considered all barcode elements that could be built with $N = 24$ patches and k bars (see Sec. 3.4 for details about our experimental dataset). We then estimated (via cross-validation over multiple illuminants) the probability that the barcode element i is incorrectly decoded as j with $i \neq j$. Obviously, the sum of all these probabilities, divided by the number of barcode elements, gives the probability of incorrect decoding P_E. To build the plot in Fig. 3.2, we ordered all length k barcode element pairs according to decreasing distance. Then we computed the cumulative sum of all probabilities of decoding i as j, divided by the number of barcodes. The plot clearly shows that the contribution to the overall probability of decoding error P_E is due for the most part to the barcode element pairs that are closest to each other. This suggests that barcode element subset selection could be accomplished based on pairwise subspace distance. For example, for small k, we adopt the following greedy strategy. Start from a barcode element chosen at random. At each iteration, add to the subset

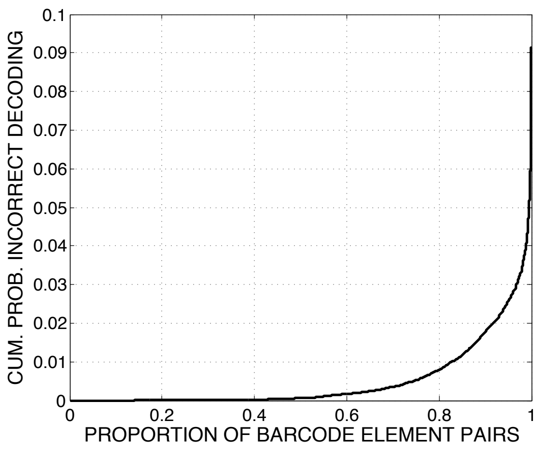


Figure 3.2: The cumulative probability of incorrect decoding as a function of the proportion of ordered barcode element pairs considered for $k = 3$ (see text). All subspaces have dimension of 2.

the barcode element that maximizes the minimum subspace distance to all barcode elements already selected. For larger values of k, even this procedure may become too computationally intensive, and we resort to a simpler strategy. First, we compute the probability of incorrect decoding for each barcode element. Then, we build the subset from the barcode elements that have the smallest probability of incorrect decoding.

3.4 Experiments

The reference color patches for our experiments were selected from a checkerboard of 512 colors, uniformly sampled in (R,G,B) color space, printed on paper by a regular printer. Images were taken of the checkerboard with a Canon EOS 350D camera in raw (CR2) format under 69 different lighting conditions (including direct sunlight,

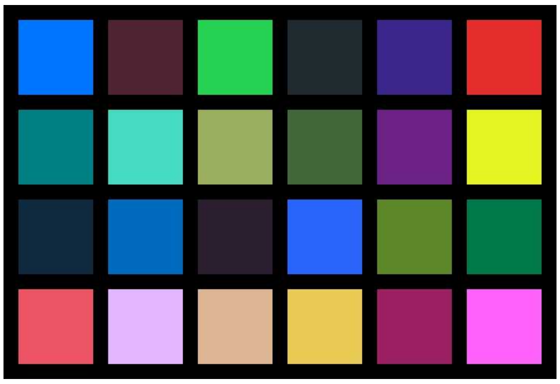


Figure 3.3: The 24 color patches selected for our tests. The first two rows contain the colors for the tests with N=12.

diffuse skylight with overcast sky or under cast shadow, and various types of artificial light). We selected ten representative illuminants by k-means clustering of the observed color values of a white patch in the set.

We then selected two sets of reference color patches, $\mathcal{C}_{24}$ and $\mathcal{C}_{12} \subset \mathcal{C}_{24}$ for $N = 24$ and $N = 12$ respectively, using the greedy strategy introduced in Sec. 3.3.4. The selected colors are shown in Fig. 3.3. Synthetic images of all barcode elements for k ranging between two and five were built from the average color values of the images of the reference color patches seen under the ten representative illuminants (where all patches forming a barcode elements were seen under the same illuminant). For each set of length k barcode elements, we extracted subsets with various proportion x as discussed in Sec. 3.3.4. For each such subset, we computed the probability of incorrect decoding $P_E(k, x)$ as follows. We ran five rounds of cross-validation, each time picking five illuminants at random, learning the subspaces for each barcode element considered based on its images under these illuminants, and testing each barcode element in turn on one of the remaining illuminants, randomly chosen. We counted the number of times any barcode element was incorrectly decoded, and divided the result by the number of cross-validation rounds (five) and by the number of barcode elements in the subset (equal to $x \cdot 24!/(24-k)!$). We also tested the decoding algorithm based on the diagonal model discussed in Sec. 3.3.2. In this case, the color subspaces were built from observation of the reference colors under just one illuminant. We ran five rounds of cross-validation, each time selecting one illuminant at random (without repetition), training our model on such illuminant and testing it with barcode elements seen under another randomly chosen illuminant.

Fig. 3.4 shows the number of barcode elements (selected for $k = 5$ and $x = 0.003$ using the selection algorithm discussed at the end of Sec. 3.3.2) containing each one of the 24 reference colors in $\mathcal{C}_{24}$. Note that 22 reference colors are chosen with comparable probability; one color had much higher probability of being selected, while another color was selected much less often than the others.

Fig. 3.5 shows the probability of incorrect decoding P_E for barcode length k between two and five and for subspace dimension M between one and four (see Sec. 3.3.2). Note that, for each k, there is an optimal value of the subspace dimension M, and that the error increases for larger values of M. This may be due to the fact that increasing the subspace dimension M may lead to overfitting and poor generalization. In the experiments presented below, we chose to use $M = 2$ for $k = 2, 3$ and $M = 3$ for $k = 4, 5$.

Fig. 3.6 shows the probability of decoding error $P_E(k, x)$ for a generic barcode element of length k between 2 and 5, using $N = 12$ and $N = 24$ reference colors, and for

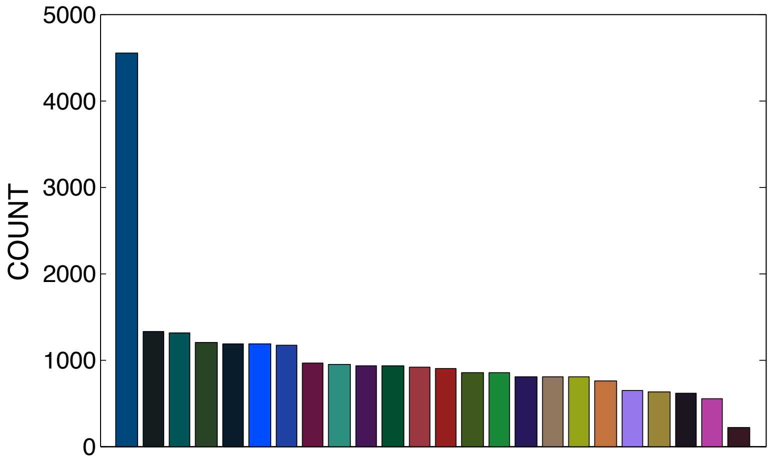


Figure 3.4: For each one of the 24 reference colors, the plot shows the number of barcode elements selected with $k = 5$ and $x = 0.003$ containing that color.

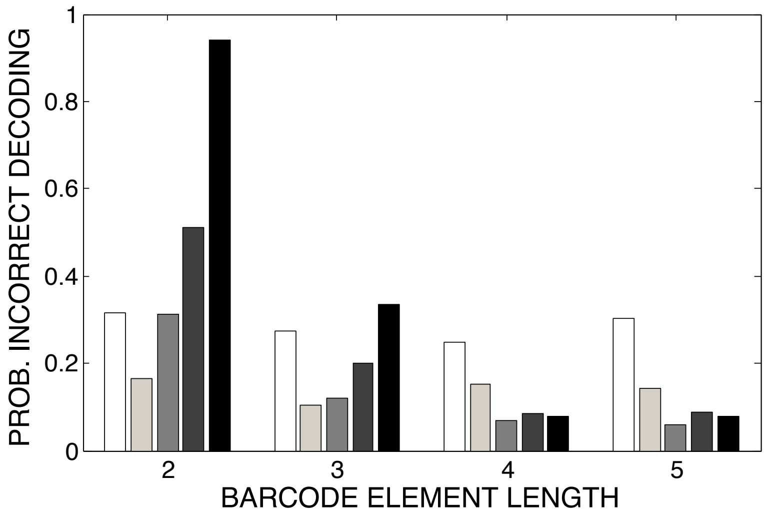
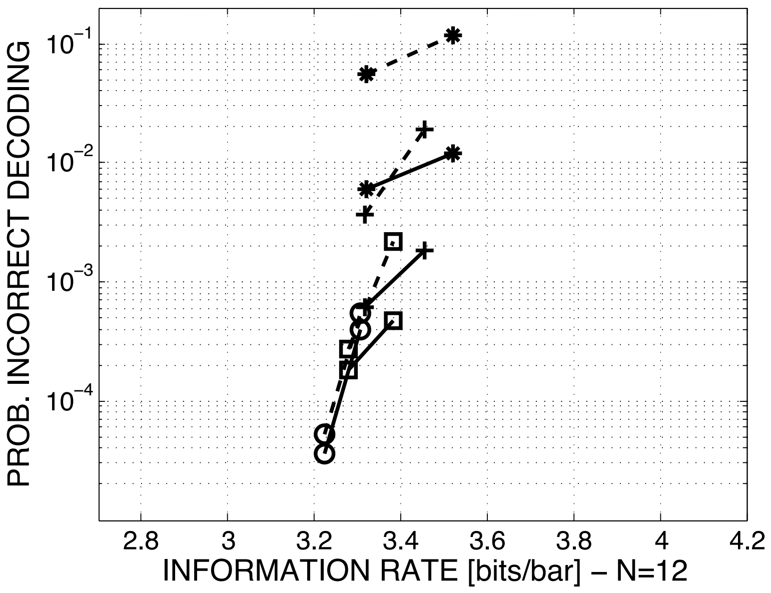


Figure 3.5: The probability P_E of incorrect decoding as a function of the barcode element length k and subspace dimension M from 1 (white bars) to 5 (black bars).

various values of the subset proportion x, as a function of the resulting information rate (as by (3.1)). Both types of subspace modeling (via PCA or via the diagonal model) are considered. Fig. 3.7 shows the probability of decoding error $P_E(K, k, x)$ for a K length barcode, computed using Eq. (3.5) for $K = 60$, 120, and 240.

The main observation that can be drawn from these results is that it is possible to reach relatively high information rate with very low error rate. For example, PCA-based subspace modeling for $k = 5$ and $N = 24$ results in a probability less than 0.001 of incorrect decoding of a length 60 barcode, while allowing one to encode information at a rate of about 3.8 bits per bar. To put this result in context, let us recall from Fig. 3.1 that the maximum information rate of a length 60 barcode that displays its reference colors is of less than 3 bits per bar. Thus, our system allows one to pack about 0.8 additional bits per bar (or 48 bits overall) in a length 60 barcode with very low decoding error probability. One can easily infer from Eq. (3.1) that an information rate of 3.8 bits/bar for a length 5 barcode element and $N = 24$ colors is achieved for $x = 0.103$, and thus decoding each barcode element requires finding the nearest subspace in a database of 524,288 elements.

The effect of the number of reference colors N, barcode element length k, and subspace modeling algorithm on the resulting information rate and decoding error probability are clear from Fig. 3.6 and 3.7. Increasing the number of reference colors N allows one to achieve higher values of information rate. Longer barcode elements result in lower decoding error probability for the same information rate. For this reason, we do not allow color repetition in a barcode element. Repeating a color in a length k



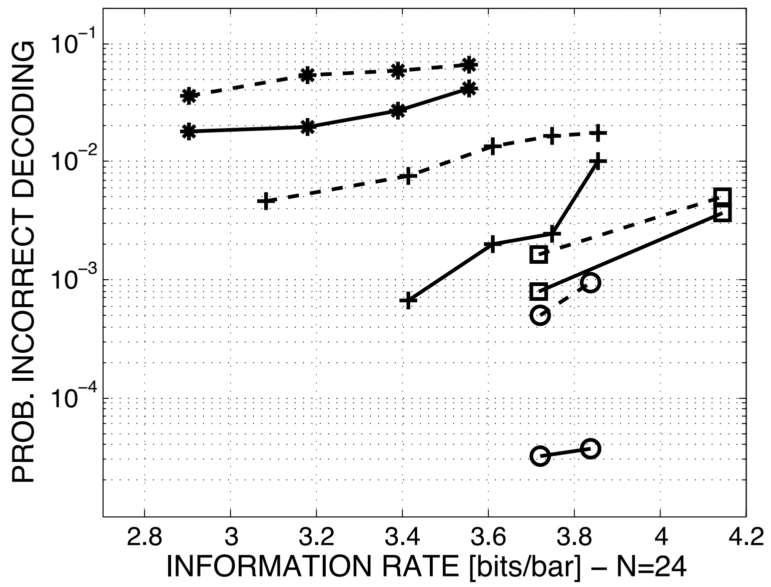
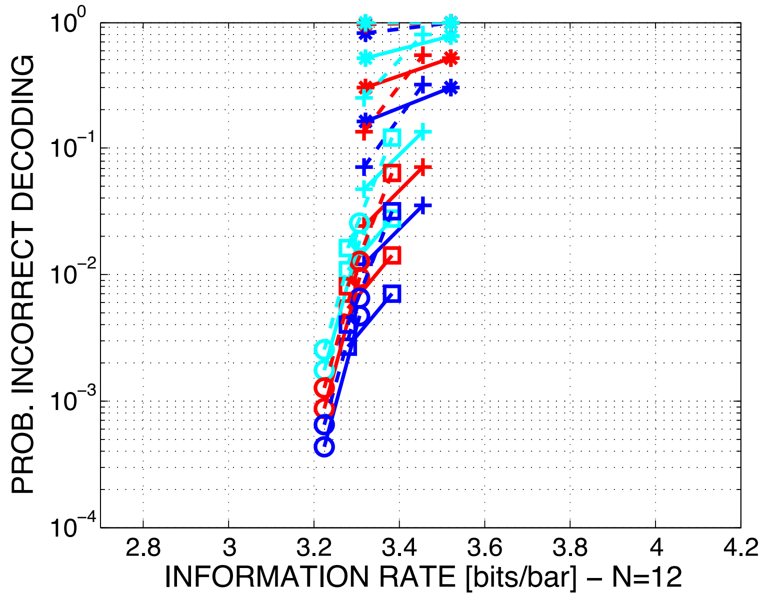


Figure 3.6: The probability incorrect detection $P_E(k, x)$ (on a logarithmic scale) for length k barcode elements versus the information rate R (3.1). For each value of the barcode element length k, a variable number of subset size proportions x were tested. '$*$': k=2; '+': k=3; '□': k=4; '○': k=5. Continuous line: subspaces learnt via PCA over five illuminants. Dashed line: diagonal model (4.6).



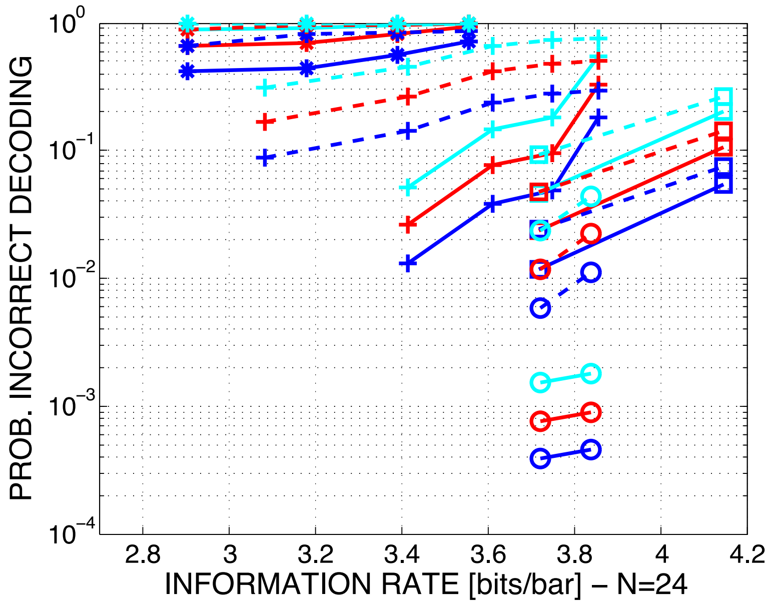


Figure 3.7: The probability incorrect detection $P_E(K, k, x)$(on a logarithmic scale) for a length K barcode element formed by multiple length k barcode elements versus the information rate (3.1). For each value of the barcode element length k, a variable number of subset size proportions x were tested. '$*$': k=2; '+': k=3; '□': k=4; '○': k=5. Blue line: K=60; red line: K=120; cyan line: K=240. Continuous line: subspaces learnt via PCA over five illuminants. Dashed line: diagonal model (4.6).

barcode is equivalent (for what concerns decoding) to include $\mathcal{B}_{k-1}$ in $\mathcal{B}_k$, leading to a substantial increase of the decoding error for $\mathcal{B}_k$. Using the diagonal model to build joint color subspaces (which, as discussed in Sec. 3.3.2, allows for a fast calibration procedure) leads to an increase of the decoding error rate by a factor of 10. Even so, for moderately long barcodes (e.g. $K = 60$ bars), the decoding error probability remains low (the probability of decoding error for a barcode with 60 bars is equal to 0.01 at $R = 3.8$ bits per bar).

3.5 Conclusions

We have proposed a new algorithm for decoding barcode elements in a color barcode that does not display its reference colors. Our experiments have shown that, by carefully selecting a subset of barcode elements, it is possible to achieve good information rate at low decoding error probability. Thus, this approach represents an efficient alternative to mainstream barcode technology that requires display of the reference colors, thereby limiting the effective information rate.

More research work is needed to compare the decoding error rate achieved by our system with other sources of error in practical situations (for example, errors due to blur-induced color mixing from two nearby patches or to printed color drift and fading). As mentioned in Sec. 3.3.3, nearest neighbor search techniques can be used for finding the closest subspace to a color vector by means of a mapping into a higher dimensional space. We also address the issue of fast decoding in chapter 5.

Chapter 4

Subspace-based Color Barcode Decoding Using Reference Colors

4.1 Introduction

In this contribution, our focus is a more efficient and faster solution for decoding color information in a barcode. We consider color barcodes that can be decoded under multiple illuminants with small number of reference patches of known colors, seen under the same illuminant as the color to be decoded. The use of reference color patches enables robust decoding with relatively low computational complexity which is suitable for implementation on a low powered mobile device.

Displaying the reference colors in the barcode enables simple decoding strategies. For example, one may compare each color patch to the reference colors, and select the reference color that is closest to the color of the patch. At the same time, displaying all colors in the palette may be counterproductive, in terms of information rate, when large palettes are used [53]. In other words, for large palette size N, the savings produced by a large variety of color palette are offset by the need to display all colors in the palette. Based on this observation, we propose the use of fairly large palettes with a limited number of reference colors displayed in the barcode. Rather than comparing a color patch to a reference color, we modeled the joint color variation of the patch and of the reference colors under varying illuminant by a low-dimensional linear space. These subspaces (one per each color in the palette) can be learned offline with training images taken under multiple illuminants. When decoding a barcode image, each patch is analyzed individually, together with the reference colors. Decoding the patch color becomes a problem of associating the vector formed by the patch color and the reference colors to the closest subspace.

Our linear model is built under the assumption of Lambertian surface reflectance, and thus is liable to failure when substantial specular reflection is present in the image. Since the barcode material (e.g. printed paper) is hardly Lambertian, a specular component is to be expected when the barcode is viewed from an angle. We extended our model assuming Lambertian reflectance to explicitly account for specular reflection. We use the dichromatic model [60] to describe the appearance of a surface under specular reflection, and show how this can be included in our subspace-based decoding approach, which is augmented based on the observed color of a white patch. The experimental results on images taken under a wide variety of illuminants and viewing angles show a substantial improvement (in terms of reduced decoding error rate) with respect to the original system that assumed Lambertian surface reflectance. In quantitative terms, we show that, by using a palette with $N = 20$ colors and 4 reference colors displayed in the barcode, we are able to encode a 128-bit message using 34 patches overall with 0 decoding errors in our test set. Compared to the 4-color HCCB standard that displays all 4 colors (and thus requires 68 patches to encode the same message), we achieve a reduction of the barcode size by one half.

This contribution builds on our work described in chapter 4. However, rather than considering groups of k color patches, we concentrate on the variation of one color patch and r reference patches as a group of color patches to model their evolution due to changing illuminant.

4.2 Background and Definitions

A color barcode is created from a set $\mathcal{C}_N \in \mathcal{C}$ of N colors (*palette*) and a set $\mathcal{C}_r \in \mathcal{C}$ of r reference colors. A color barcode of length $K = n + r$ is defined as the arrangement of n color patches, selected from the palette $\mathcal{C}_N$ and used for information encoding, and the r reference colors of $\mathcal{C}_r$, in any spatial configuration. Decoding the bar code means assigning the color of each one of the n information-carrying patches to the index of the corresponding color in the palette $\mathcal{C}_N$. As with standard color barcodes (e.g. HCCB), we assume that the position of the reference colors in the barcode is known. In this work we showed that by carefully modeling the joint color variation as a function of the illuminant, it is possible to use $r < N$ reference colors and still obtain good decoding performance. In fact, we don't even constrain the set of reference colors $\mathcal{C}_r$ to be a subset of the color palette $\mathcal{C}_N$.

As defined in chapter 4, the information rate R of a barcode (reference colors are not used) is defined by the logarithm base 2 of the number of different symbol that can be represented by the barcode and measured in bits per bar. The information rate for a barcode of K bars and r reference patches is defined as:

$$R(K, r) = \left(1 - \frac{r}{K}\right) \log_2 N \tag{4.1}$$

Since each color patch carries $\log_2 N$ bits of information, the barcode carries $n \log_2 N$ bits. In order to encode B bits, one needs this many color patches:

$$K = n + r = \lceil B / \log_2 N \rceil + r \tag{4.2}$$

Increasing the palette size N and reducing the number r of reference colors decreases the size K of the barcode, resulting in higher *information rate*. For example, Fig. 4.1 shows the minimum length K of a color barcode that encodes a message of $B = 128$ bits as a function of the size of the color palette N and of the number of reference colors r. (Note that the HCCB standard with $N = r = 4$ would require $K = 68$ color patches for the same 128-bit message.) For a fixed number r of reference colors, the barcode length K decreases monotonically with the size of the color palettes N. In contrast, if the whole palette is represented by the reference colors ($r = N$), the plot of K at $N = 13$ ($K = 48$), after which adding colors to the palette becomes counterproductive.

While the plot in Fig. 4.1 suggests that large palettes with few reference colors lead to high information rate, it hides the fact that increasing the palette size typically

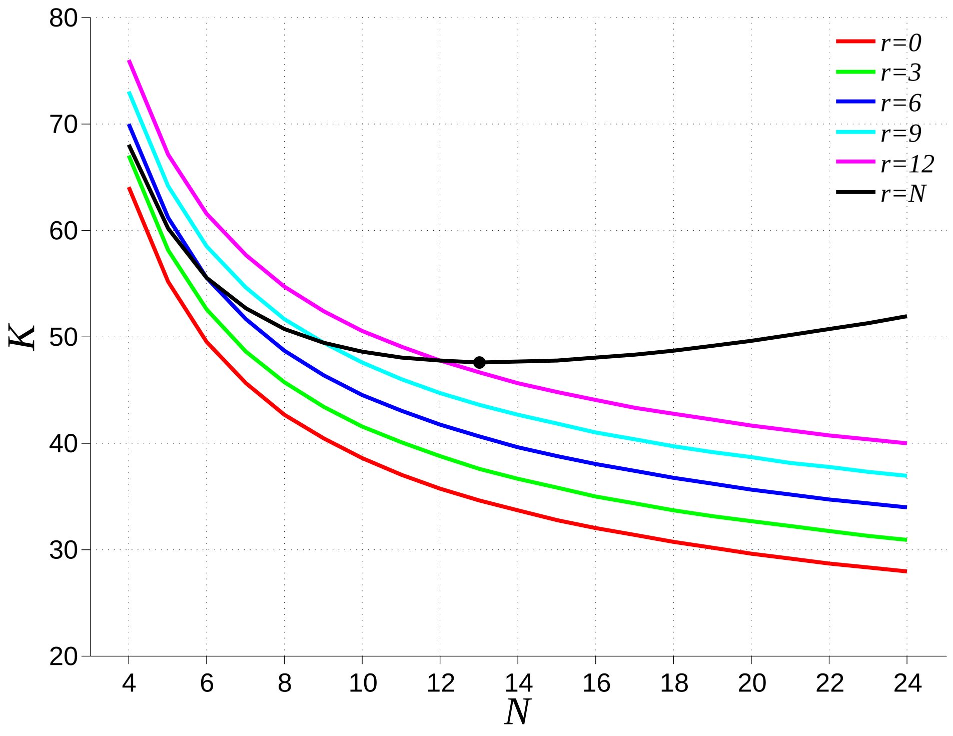


Figure 4.1: The number of patches required to encode 128 bits versus the number of available colors. The minimum number of colors to display the palette is 13 (circle).

results in larger decoding error rates, which must be offset by adding more reference colors. Let $P_E(N, r)$ is the probability of decoding error (that is, of misclassifying the color of a patch) for a given palette $\mathcal{C}_N$ and a given set of reference colors $\mathcal{C}_r$. Assuming that decoding errors are statistically independent events, the *decoding error rate*, that is, the probability of decoding error for the barcode (i.e., of decoding at least one color patch incorrectly) is equal to:

$$P_E(N, r, K) = 1 - (1 - P_E(N, r))^{K-r} \tag{4.3}$$

One may expect $P_E(N, r)$ to increase with increasing N (larger palette) and decrease with increasing r (more reference colors). This is verified experimentally in Sec. 4.4. This relation establishes a design trade-off between decoding error probability and information rate, mediated by the parameters N and r.

4.3 Color Barcode Decoding

4.3.1 The Lambertian Case

Consider a patch colored with the i-th color in the palette $\mathcal{C}_N$. The observed color $\mathbf{c}_i = [c_{i,R}, c_{i,G}, c_{i,B}]^T$ of this patch will vary as the illuminant changes (e.g., from sunlight to artificial light), making color identification difficult. The key observation we made in [7] is that the *joint* color variation of a set of color patches, all under the same illumination, is bound by a linear constrain. For example, consider the $3(r + 1)$–dimensional vector $\mathbf{e}_i = [\mathbf{c}_i, \mathbf{d}_1, \ldots, \mathbf{d}_r]^T$ which includes the colors of the (known) reference patches $\{\mathbf{d}_j\}$, with $\mathbf{d}_j = [d_{j,R}, d_{j,G}, d_{j,B}]^T$. If the surfaces are Lambertian, and assuming that the illuminant spectra live in a finite-dimensional subspace of dimension N_{ill}, then the vector $\mathbf{e}_i$ must live in a linear subspace $\mathcal{S}_i$ of dimension equal to $\min(3(r+1), N_{\text{ill}})$, which can be considered equal to N_{ill} for $r \geq 1$ [37, 63, 45]. Formally:

$$\mathbf{e}_i = \mathbf{\Phi}_i \mathbf{v} \tag{4.4}$$

where $\mathbf{v}$ is a N_{ill}–vector that represents the illuminant, and $\mathbf{\Phi}_i$ is a full-rank matrix that characterizes the reflectivity of the patch surface[1]. The observed color $\mathbf{c}_i$ of a single surface under a given illuminant is equal to $\mathbf{c}_i = \mathbf{\Phi}_{\mathbf{c}_i}\mathbf{v}$. The decoding of $\mathbf{c}_i$ would be very difficult if no reference color is used. This is due to the fact that $N_{\text{ill}} \geq 3$ in general ([24], [54]), and the rank of $\mathbf{\Phi}_{\mathbf{c}_i}$ would be 3 which makes $rank(\mathbf{\Phi}_{\mathbf{c}_i})$ small relative to N_{ill}. If, however, a color along with multiple reference colors seen under the same illuminant is decoded at once, the probability of correct decoding will be higher due to $\mathbf{e}_i \in \mathbb{R}^{3(1+r)}$ and $rank(\mathbf{\Phi}_i) = min(3(1+r), N_{\text{ill})}$. The vector $\mathbf{e}_i$ is constrained to live in a subspace $\mathcal{S}_i$ of dimension of at most N_{ill}. It is useful to define the *dimensionality ratio (DR)* as the ratio between the dimension of the embedding subspace $\mathcal{S}_i$ and the dimension of $\mathbf{e}_i$:

$$\text{DR} = \frac{N_{\text{ill}}}{3(r+1)} \tag{4.5}$$

This suggests a simple algorithm for decoding a generic color patch $\mathbf{c}$: (1) Build the vector $\mathbf{e}$ (by juxtaposing the observed color $\mathbf{c}$ with the observed colors of the reference patches in the same barcode); (2) Find the subspace $\mathcal{S}_i$ that is closest to the vector $\mathbf{e}$; (3) Decode $\mathbf{c}$ as i. Intuitively, the smaller the dimensionality ratio DR (itself an decreasing function of r), the higher the robustness of this decoding algorithm with

[1]Note that $\mathbf{\Phi}_i$ is also a function of the illumination and reflectance basis vectors as well as of the spectral sensitivities of the camera assuming Lambertian characteristics of surfaces.

respect to noise. This formalizes the intuitive notion that more reference colors should ensure lower decoding error rates. The subspaces $\mathcal{S}_i$ for $1 \leq i \leq N$ can be learnt from observation of the colors in $\mathcal{C}_N \cup \mathcal{C}_r$ under a wide variety of illuminants. If multiple pictures of the color patches under different illuminants are impractical or impossible to obtain, one may "constrain" the embedding subspaces $\mathcal{S}_i$ by means of the diagonal (von Kries) model of color change [28]. The diagonal model assumes that each color channel changes as a result of an illuminant change by a multiplicative factor. Indeed, under the diagonal color model, the matrix $\mathbf{\Phi}_i$ can be written as

$$\mathbf{\Phi}_i^T = \begin{bmatrix} c_{i,R} & 0 & 0 & d_{1,R} & 0 & 0 & d_{2,R} & \cdots \\ 0 & c_{i,G} & 0 & 0 & d_{1,G} & 0 & 0 & \cdots \\ 0 & 0 & c_{i,B} & 0 & 0 & d_{1,B} & 0 & \cdots \end{bmatrix} \tag{4.6}$$

It is easy to see that this matrix can be learnt from observation of the colors under just one illumination. However, the resulting decoding error rate are typically higher than with the "unconstrained" subspace approach.

4.3.2 The General Case

In the real world, surfaces are rarely Lambertian, and the reflected light should be expected to contain a specular component. The amount of this specular component depends on the surface characteristics and on the joint illumination/viewing geometry. For color barcodes printed on paper, the specular component can be quite noticeable [6].

The dichromatic reflection model [61] for RGB pixel values is defined as follows:

$$\mathbf{c} = m^{(b)}\mathbf{c}^{(b)} + m^{(i)}\mathbf{c}^{(i)} \tag{4.7}$$

The dichromatic model states that the observed color $\mathbf{c}$ of a surface is the sum of two colors, $\mathbf{c}^{(b)}$ (*body reflection*) and $\mathbf{c}^{(i)}$ (*interface reflection*), weighted by coefficients $m^{(b)}$ and $m^{(i)}$. The term $m^{(b)}$ and $m^{(i)}$ depend only on the viewing angle, illumination direction, and surface orientation. $\mathbf{c}^{(b)}$ is the color of the body (or diffuse) reflection of the material at a given pixel. $\mathbf{c}^{(i)}$ is the color of the interface reflection which can represent the illuminat color. If we assume that the spectral power distribution of the specular reflection is similar to the spectral power distribution of the incident light, then the approximate Lambertian reflection of a bright patch (e.g. white patch), which reflects the maximum intensity of illuminant possibly for each color component, represents the illuminant color. Thus, the dichromatic reflection model predicts that the specular reflection "steers" the color of the surface towards the color of a white surface seen under the same illuminant. This observation suggests that if the barcode contains a white reference patch, the color of this white patch may be used to "remove" the specular component from the color of other patches, provided that one can somehow estimate the coefficient $m^{(i)}$ at each patch. In practice, it is reasonable to assume that $m^{(b)} \leq 1$ and $m^{(i)} \geq 0$ for all illumination and viewing directions. It is also assumed that $m^{(b)}$ is relatively insensitive to changes in viewpoint (and thus can be safely set it to 1). In contrast, $m^{(i)}$ is highly viewpoint dependent.

Formally, we can model the color of the vector $\mathbf{e}$ defined in the previous section as follows:

$$\mathbf{e}_i = \mathbf{\Phi}_i \mathbf{v} + \mathbf{W}\mathbf{m}^{(i)} \tag{4.8}$$

with

$$\mathbf{W} = \mathbf{I} \otimes \mathbf{w} = \begin{bmatrix} \mathbf{w} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{w} & \mathbf{0} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{4.9}$$

where $\mathbf{I}$ is the $(r+1) \times (r+1)$ identity matrix, $\otimes$ represents the Kronecker product, $\mathbf{w}$ is the observed color of the white reference patch, and $\mathbf{m}^{(i)}$ is a $(r+1)$–vector containing the interface reflection coefficients for all patches in $\mathbf{e}$. This suggests that the subspace approach used for the Lambertian case could be extended to the general case with specularities, owing to the observed white patch. However, it should be noticed that the presence of specular reflection increases the dimensionality[2] of the embedding space $\mathcal{S}$ to $N_{\text{ill}} + r$. With respect to the Lambertian case, the dimensionality ratio DR is thus increased by a factor of $1 + r/N_{\text{ill}}$, making decoding harder.

In order to keep the dimensionality ratio under control, in this work we assume that $m^{(i)}$ is constant across patches for a fixed illuminant. This simplifying assumption can be partly justified by the fact that, for a small sized planar barcode, the viewing geometry can be considered approximately constant for all patches. Hence, assuming constant $m^{(i)}$ across patches means neglecting the difference between interface reflection coefficients for the different patches in the barcode. This approximation allows us to rewrite Eq. (4.8) as follows:

$$\mathbf{e}_i = \begin{bmatrix} \mathbf{\Phi}_i \mid \mathbf{V} \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ m^{(i)} \end{bmatrix} \tag{4.10}$$

[2]Note that the white patch is assumed to be part of the reference colors. For this patch, the specular reflection component is immaterial. This is the reason why the dimension of the embedding space is $N_{\text{ill}} + r$ rather than $N_{\text{ill}} + r + 1$.

with $\mathbf{V} = [\mathbf{w}^T, \dots, \mathbf{w}^T]^T$ obtained from the observed color of the white reference patch. With this simplification, the dimensionality ratio DR is only increased by a factor of $1+1/N_{\text{ill}}$ with respect to the Lambertian case. Note that the matrix $\mathbf{\Phi}_i$ is computed from training data in the absence of specular reflection. In practice, this can be achieved by ensuring that, when taking training images, the color patches lie on a plane orthogonal to the camera's optical axis. Also note that, in order to compute the distance of the vector $\mathbf{e}$ to a subspace $\mathcal{S}_i$, it is useful to first derive an orthogonal column basis for $[\mathbf{\Phi}_i|\mathbf{V}]$, which can be achieved via QR decomposition by considering the first $N_{\text{ill}} + 1$ columns of $\mathbf{Q}$ matrix.

4.3.3 Reference Color and Subspace Selection

The reference colors can be sampled from the $\mathcal{C} - \mathcal{C}_N$ colors that are not used for the color palette $\mathcal{C}_N$. (Note that choosing reference colors from the palette would *increase* the dimensionality ratio DR from $N_{\text{ill}}/3(r+1)$ to $N_{\text{ill}}/3r$, making decoding harder.) We used a greedy recursive strategy for jointly selecting r reference colors (with $1 \leq r \leq 5$) and the dimensionality of the embedding subspaces $\{\mathcal{S}_i\}$, which was kept constant across subspaces for given r. Reference patches are added one at a time.

To produce a certain $P_E(N, r)$ smaller than desired, one needs to find reference colors within a set of colors. Selection of r number of reference colors that minimizes the associated $P_E(N, r)$ is computationally very expensive. In particular, the probability of incorrect decoding for all combinations of r reference colors from a large set of colors needs to be evaluated. The suboptimal recursive greedy technique reduces the complexity associated with r reference colors selection. Our approach is to select one reference color at a time to compute the $P_E(N, r)$. We select the reference colors from the set $\mathcal{C} - \mathcal{C}_N$ that minimizes the $P_E(N, r)$. Given the current set of r reference colors, all possible remaining $\mathcal{C} - \mathcal{C}_N - r$ colors and subspace dimensions from 1 to 5 are tested using cross-validation over multiple illuminants, and the marginal error rates $P_E(N, r)$ are computed. The reference color and subspace dimension (for subspace-based approach) that minimize the decoding error rate are selected and added to the set. For $r = 1, 2$ the algorithm chose an embedding subspace dimension 3, while for $r = 3, 4, 5$ the chosen subspace dimension was of 4. The subspace dimension for diagonal model is 3 (Eq. (4.6)). The white patch was then added to the reference colors when the dichromatic model is used. Fig. 4.2 shows the probability of incorrect decoding $P_E(N, r)$ for the number of reference colors r between one and five and the subspace dimension between one and five. In Fig. 4.2, the probability of incorrect decoding is very high if the subspace dimension is 5 and $r = 1$. This may be due to the fact that increasing the subspace dimension may lead to over fitting and poor generalization.

4.4 Experiments

We ran a number of experiments with color checkerboards printed on paper with a regular color printer. Images were taken of the checkerboards with a Canon EOS 350D camera in raw (CR2) format with a resolution of 3474×2314 pixels and 12 bits per color channel. The raw format captures as closely as possible the radiometric

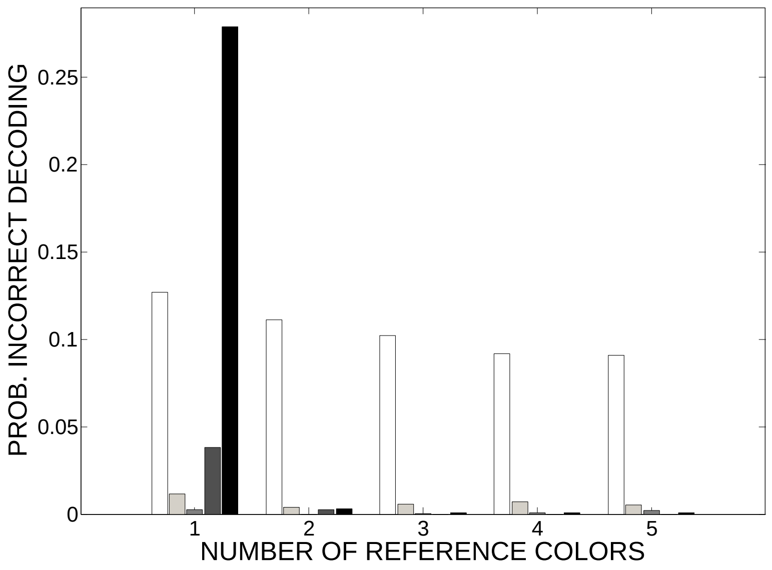


Figure 4.2: The probability $P_E(N, r)$ of incorrect decoding as a function of the number of reference colors r and subspace dimension from 1 (white) to 5 (black).

characteristics of the scene which is the physical information about the light intensity and color of the scene. The future sensors of mobile phones will be capable of capturing images in raw RGB format.

4.4.1 Training Set and Model Construction

For our experiments, we printed a colorchecker with 125 colors on paper by a regular printer, uniformly sampled in (R, G, B) color space. Images were taken of this colorchecker from a a constant distance of about 1.5 meters, with the camera's optical axis orthogonal to the checkerboard to minimize specular reflections, under 32 different illumination conditions including indoor and outdoor under direct sunlight, diffuse skylight with overcast sky, cloudy sky, or under cast shadow, and various types of artificial light. The color values within each patch were averaged together to reduce noise.

We select the color palettes $\mathcal{C}_{12} \subset \mathcal{C}_{16} \subset \mathcal{C}_{20} \subset \mathcal{C}_{24}$. For each illuminant, we clustered the colors of the patches using k-means with 24 clusters. We then selected the 24 cluster centers with highest occurrences among all illuminants. We repeated the same procedure to select the colors of the palettes for $N=$ 20, 16 and 12, each time starting from the palette chosen in the previous step. To make sure that the k-means clustering selects 24 distinct colors, we run k-means 10 times with different starting point to select N colors with highest occurrences in all runs.

The color of reference patches were selected from $125 - 24$ colors as described in Sec. 4.3.3 for subspace-based method and diagonal model. The colors for reference

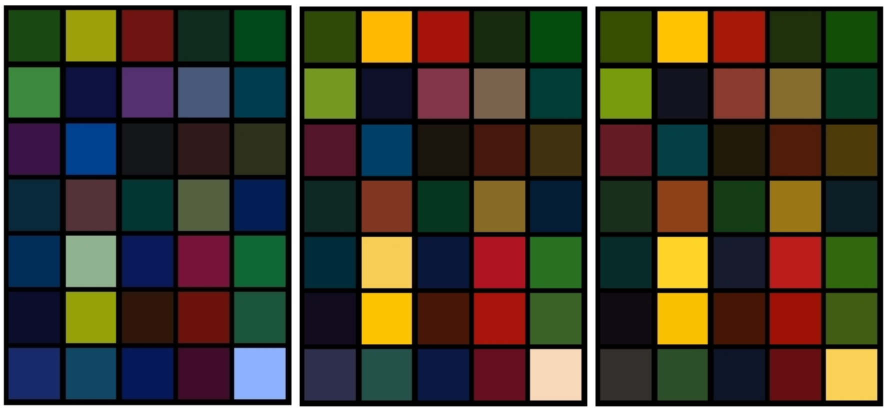


Figure 4.3: A collage created with the chosen palette colors and reference colors, seen under three different illuminant spectra. The patches are distributed in such a way that, in lexicographic order, the first N patches form $\mathcal{C}_N$ for $N = 12, 16, 20, 24$. Following are the 5 reference colors chosen for the "unconstrained" subspace decoding algorithm, followed by the 5 reference patches chosen using the diagonal color model. The last patch is the white patch.

colors are different than the $N = 24$ colors since it is desired to build each vector $\mathbf{e}$ with distinctive colors rather than allowing color repetition in the vector $\mathbf{e}$, and reduce the dimensionality ratio DR. Figure 4.3 shows selected 24 colors and 5 reference colors seen under three different illuminants.

Along with the palette and the reference colors, we computed the embedding subspaces (represented by the matrices $\{\mathbf{\Phi}_i\}$) for all combinations of palette size N and number of reference patches r, using data from all 32 illumination conditions. Additionally, we learnt the matrices $\{\mathbf{\Phi}_i\}$ using the diagonal model (4.6) for all combinations (N, r). However, since these matrices can be learnt from just one image, we created 32 versions of each $\mathbf{\Phi}_i$, one per illumination condition. The matrices $\{\mathbf{\Phi}_i\}$ are used in (4.10).

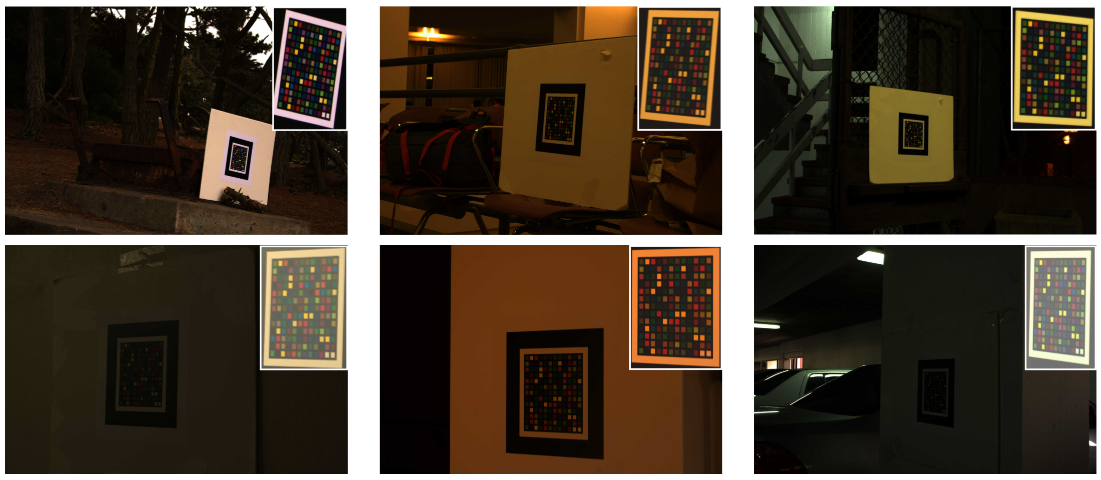


Figure 4.4: Examples of images with the "test" color checkerboard, used in our experimental tests. The inset at each picture shows the brightness-rescaled, zoomed-in checkerboard detail.

4.4.2 Test Set and Results

We evaluated the performance of our proposed decoding algorithms using a 13×12 "test" color checkerboard with size of 16.5×15 cm, printed with the same color printer used for the "training" checkerboard. The first six pairs of rows each contain all 24 colors in the palette, in random order. The last row contains the five reference colors selected for the unconstrained subspace model, followed by the five reference colors chosen for the diagonal model and by two white patches. In order to facilitate automatic checkerboard detection and patch localization in our pictures, we printed a thick black edge outside the pattern, outlining a visible white frame (see Fig. 4.4). (This design was inspired by the ARToolKit marker concept [39].) Of course, in a real application, a smaller frame (or no frame at all) would have to be used.

We took 100 images from the test checkerboard under multiple illumination conditions, multiple viewing angles (ranging from -45 to 45 degrees with respect to the normal to the checkerboard surface), and multiple distances (1 to 5 meters). Figure 4.4 shows some examples of our test images. Each color patch was automatically localized, and color values within the central area of the patch were averaged together to reduce noise (resulting in one color value per patch).

We evaluated the marginal error rate $P_E(N, r)$ for a combination of design choices: (a) using the unconstrained vs. the diagonal subspace model (4.6); (b) using the Lambertian reflection model (4.4) vs. the dichromatic reflection model (4.10). For each design choice, we considered all combinations of parameters[3] N and r. For each pair (N, r) we used the associated matrices $\{\mathbf{\Phi}_i\}$ learnt from the "training" checkerboard as discussed above.

When using the unconstrained subspace model, the error rate $P_E(N, r)$ was given by the total number of color patches in the "test" checkerboard that were incorrectly decoded, divided by the number of images (100) and by the number of color patches in the colorchecker ($N \times 6$). The computation of the error rate using the diagonal model (4.6) is slightly different, as in this case there are 32 different versions of each matrix $\mathbf{\Phi}_i$, one per illumination condition. We tested all such matrices, and computed the error rate as the total number of color patches in the "test" checkerboard that were incorrectly decoded, divided by the number of images (100), by the number of color patches in the colorchecker ($N \times 6$), and by the number of illumination conditions in

[3]Note that, when using the dichromatic model, we added the white patch to the sets of reference colors used for the Lambertian reflection model, resulting in a number of reference patches larger by one.

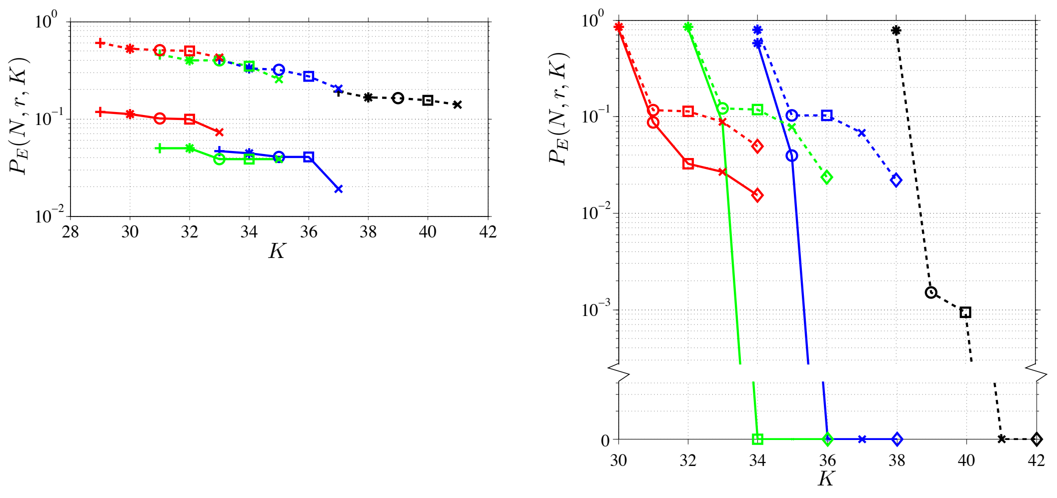


Figure 4.5: The probability $P_E(N, r, K)$ of decoding error for a 128-bit message as a function of the total barcode length K, the palette size N, the number of reference colors r, and the embedding subspace type. Black: $N = 12$; Blue: $N = 16$; Green: $N = 20$; Red: $N = 24$. '+': $r = 1$; '$*$': $r = 2$; '$\circ$': $r = 3$; '$\Box$': $r = 4$; '$\times$': $r = 5$; '$\Diamond$': $r = 6$. Solid line: unconstrained embedding subspace; Dotted line: diagonal model (4.6). Left: Assuming Lambertian reflectance (4.4), with r ranging from 1 to 5. Right: Using the dichromatic reflection model (4.10), with r ranging from 2 to 6 (the white patch was added to the chosen set of reference colors.)

the training dataset (32).

The resulting error rates $P_E(N, r, K)$ for a message with $B = 128$ bits are shown in Fig. 4.5. As expected, the error rate decreases with increasing barcode length K. The diagonal model is also shown to perform poorly compared to the unconstrained model. Using the dichromatic model results in improved performance for large enough r. Indeed, for $N = 20$ and $r = 4$, we achieve 0 error rate for $K = 34$ in our data set. This is a very promising result, considering that the same parameters yield an error rate of

0.07 using the Lambertian reflection model. To put this result in context, consider that, as discussed in Sec. 4.2, a system that represents all the palette colors in the reference set ($N = r$) requires a barcode of length K equal to at least 48 (for $N = 13$) to encode 128 bits. By using a smaller number of color patches and the dichromatic model, our algorithm is able to pack the same amount of information in a barcode that is 30% smaller. With respect to the HCCB system with $N = r = 4$ (which requires $K = 68$ patches for a 128-bit message), our algorithm allows for reduction of the barcode size by half.

When using the diagonal model, an error rate of less than 0.001 is obtained only for $K \geq 40$. In this case, there is a smaller (but still significant) gain in terms of information rate with respect to the case $N = r$. As discussed earlier, the practical advantage of the diagonal model is that it requires only one picture of the color pattern, rather than multiple pictures under a variety of illumination condition as needed by the unconstrained subspace model.

4.5 Conclusions

We have introduced a new algorithm for color barcode decoding in presence of specular reflection. Our experiments have shown that, by selecting up to 24 colors and a small number of reference colors, it is possible to achieve higher information rate than with mainstream color barcode decoding methods while ensuring low decoding error rates. Future research will consider other sources of error such as due to blur-induced color mixing from two nearby patches or to color barcodes printed from different printers.

Chapter 5

Color Barcode Decoding in Presence of Blur-induced Color Mixing

5.1 Introduction

Color barcodes accessible by mobile devices become popular as an inexpensive computing tool for information encoding. A mainstream approach to ensure robust decoding is to use a color palette of barcode colors printed with the barcode. The color palette contains all reference colors used to generate a barcode. Since the variation of a patch color and of the corresponding reference color can be assumed to be consistent due to changes such as device used, printer, and media, the patches of the barcode can be decoded robustly. For example, the HCCB method [53] clusters the colors, and assigns each cluster to one of the reference colors in the palette using the minimum distance. One challenge facing such color clustering methods is that the color distribution of one class can overlap significantly with color distributions of other classes when a large number of colors is used. Blur-induced color mixing from neighboring color patches changes the color of patches. Therefore, such clustering techniques may not work well for these situations.

Decoding a color barcode using mobile devices is even more challenging. In mobile device applications, the printed color barcode is captured from different distances and angles. The barcode images are more often out of focus since the device may focus in different part of the image when the image is captured from relatively long distance. Motion blur caused by camera shake while capturing an image contributes to decoding error. In addition, barcode localization in the captured image can cause inaccurate patch extraction in the barcode. Any or a combination of these factors could lead to incorrect decoding.

None of previous decoding approaches have addressed color barcode decoding in presence of blur-induced color mixing from neighboring color patches. In this contribution, we introduce a new technique for decoding the color information in presence of blur-induced color mixing using a small number of colors ensuring high information density. The color mixing could be caused by incorrect focus, uniform or non-uniform motion, long-distance image capture, inaccurate barcode boundary identification, and perspective projection. Decoding error rate using mainstream methods in presence of color mixing maybe high since each measured patch color is a linear combination of the colors of the patch itself and its neighboring patches. Rather than decoding individual patches using a clustering method, our iterative algorithm decodes the colors of all patches across the barcode image by minimizing the overall observation error.

5.2 Color Barcode and Color Mixing Problem

In our color barcode decoding approach, the patches in a color barcode are created from N reference colors. A color barcode can be defined as a spatial pattern of n color patches for information encoding and a color palette of rN reference color patches, resulting in a $K = n + rN$ element barcode.

Figure 5.1 shows an example of a color barcode with triangular color patches. This barcode is made of $m \times k$ quarter square triangles (QST). A QST is a square made up of four triangles (left, top, right, bottom) whose colors are selected from N reference colors as shown in Figure 1 (top). In our decoding method, we assume that N reference colors are displayed with the barcode as $N\times$ QST. Each triangular color patch is surrounded by three other triangular patches with the same or different colors.

When a barcode is observed, a robust decoding can be achieved if the distance between N clusters resulted from patch colors of the barcode is significantly large. Then a patch can be decoded by comparing its measured color to one of the measured reference colors in the palette using minimum distance. In contrast, when a color barcode is observed in presence of blur, then the measured color does not represent the color of the patch. In this case, the measured color is a mixed color resulted from the linear combination of the patch color itself and the colors of its neighboring patches based on the assumption that the device optical system is linear. Figure 5.2 shows examples of

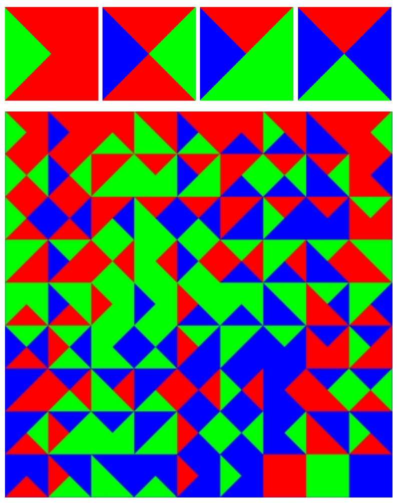


Figure 5.1: Top: Examples of quarter square triangles (QST). Bottom: Example of a color barcode made of different QST.

a color barcode imaged from multiple distances and viewing angles, and blur-induced color mixing from neighboring patches.

In case of short camera-barcode distance (assuming that images are taken at frontoparallel view) with correct focus and no motion, color mixing occurs mostly along patch boundaries, thus leaving the non-boundary pixels unaffected. In this case, the central patch pixels can be used for decoding. However, the color mixing can occur in entire patch area in presence of blur. In general, color mixing can be caused by following factors or a combination of these factors. These factors include

1. Incorrect focus: incorrect focus is caused by a shift along the optical axis away from the plane of best focus. This reduces the sharpness of the barcode image by mixing colors of patches with their neighboring patch colors.

2. Motion blur: the barcode image will appear blurred or smeared along the direction of relative camera motion with respect to the barcode, thus causes color mixing.

3. Camera-barcode distance: barcodes within a certain range of distances from the camera will appear in-focus in the barcode image. Barcodes further than this range will begin to appear noticeably blurred. This has to do with camera depth of field. The shallower the depth of field, the more quickly the blurring happens with increasing distance.

4. Inaccurate barcode boundary identification: the edges obtained from barcode images are normally affected by blur. This leads to inaccurate barcode boundary identification and patch extraction. In this case, the measured color of an extracted patch is resulted by mixing the color of the patch itself and the colors of its neighboring patches.

5. Perspective projection: barcode images and their patches under perspective projection look distorted. This means that size of each patch is different in the barcode image. Therefore, the amount of blur caused by incorrect focus, motion, or long camera-barcode distance will not be uniform across a barcode image.

Clustering techniques and minimum distance classifier may not perform well in presence of color mixing due to overlap of color distributions of one class with other classes. Figure 5.3 illustrates examples of barcode images taken at different distances. These images have been taken from relative close distances with correct focus and no motion. Each cluster is built based on averaging the central patch pixels. No overlap

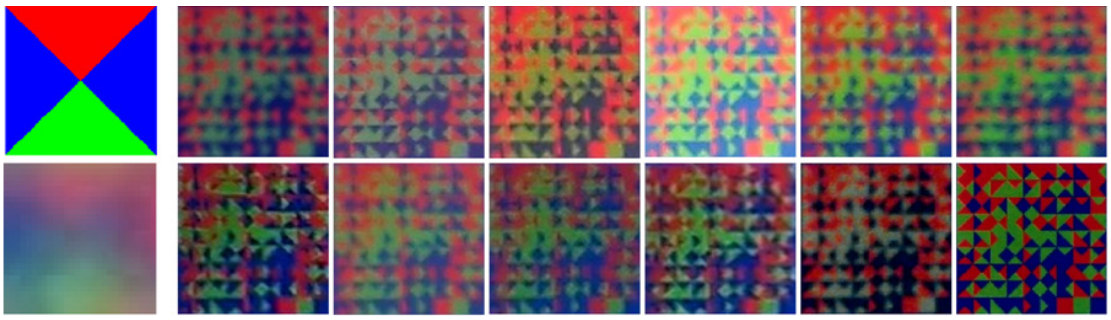


Figure 5.2: Left: An example for a QST and its image extracted from a barcode image. Right: examples of a color barcode imaged from multiple distances and viewing angles after perspective projection correction.

between the clusters can be seen in RGB space. Figure 5.4 shows barcode images in presence of blur caused by long camera-barcode distance, incorrect focus, or camera shake. The color mixing occured in a subset of patches which creates color clusters with a significant overlap that can lead to higher probability of decoding error if a minimum distance classifier is used.

The advantage of using triangular color patches is that each patch has only three neighboring patches and a linear mixing of only four colors is created as a result of blur. Selection of patch shapes with larger number of edges may lead to higher decoding error rate due to mixing of more colors from neighboring patches.

We use small number of colors in this project due to the fact that using large number of colors causes significant overlps between color distributions of classes in presence of blur-induced color mixing. In addition, the probability that more patches within a barcode have at least one neighbor with the same color as the patch itself is higher using small number of colors. This leads to more robust decoding since more neighboring patches have the same color. Therefore, it is beneficial to use small number of colors.

5.3 Decoding Method

Let the 3-dimensional vector $\bar{\mathbf{c}}_i = \left[\bar{c}_i^R, \bar{c}_i^G, \bar{c}_i^B\right]^T$ be the measured color of the triangular patch i with (R,G,B) color vector resulted by linear color mixing. When a color barcode is observed in presence of blur, our observation model is such that the measured color at each triangular patch ($\bar{\mathbf{c}}_i$) is a linear combination of the patch color itself ($\mathbf{c}_i$) and the colors of its three neighboring patches ($\mathbf{c}_{i,1}, \mathbf{c}_{i,2}, \mathbf{c}_{i,3}$) with unknown coefficients as

$$\bar{\mathbf{c}}_i = w\mathbf{c}_i + w_1\mathbf{c}_{i,1} + w_2\mathbf{c}_{i,2} + w_3\mathbf{c}_{i,3} \tag{5.1}$$

$$w = 1 - (w_1 + w_2 + w_3) \tag{5.2}$$

where $\mathbf{v} = [w, w_1, w_2, w_3]^T$ are the linear coefficients. The index 1 for a triangular patch is associated with the horizontal or vertical edge depending on the patch position in a QST. The indices 2 and 3 are associated with two other edges clock-wise starting from 1 followed by 2 and 3. The linear coefficients are calculated differently based on the position of a triangular patch in a QST. Let $\mathbf{w} = [w_l, w_t, w_r, w_b]^T$ be the amount of blur induced by left, top, right and bottom sides of the barcode. In case of square patches we could use $\mathbf{w}$ as the linear coefficients associated with four neighboring patches. In case of triangular patches the linear coefficient associated with each edge is calculated as the average of two elements of $\mathbf{w}$ depending on the direction of edge normal components as

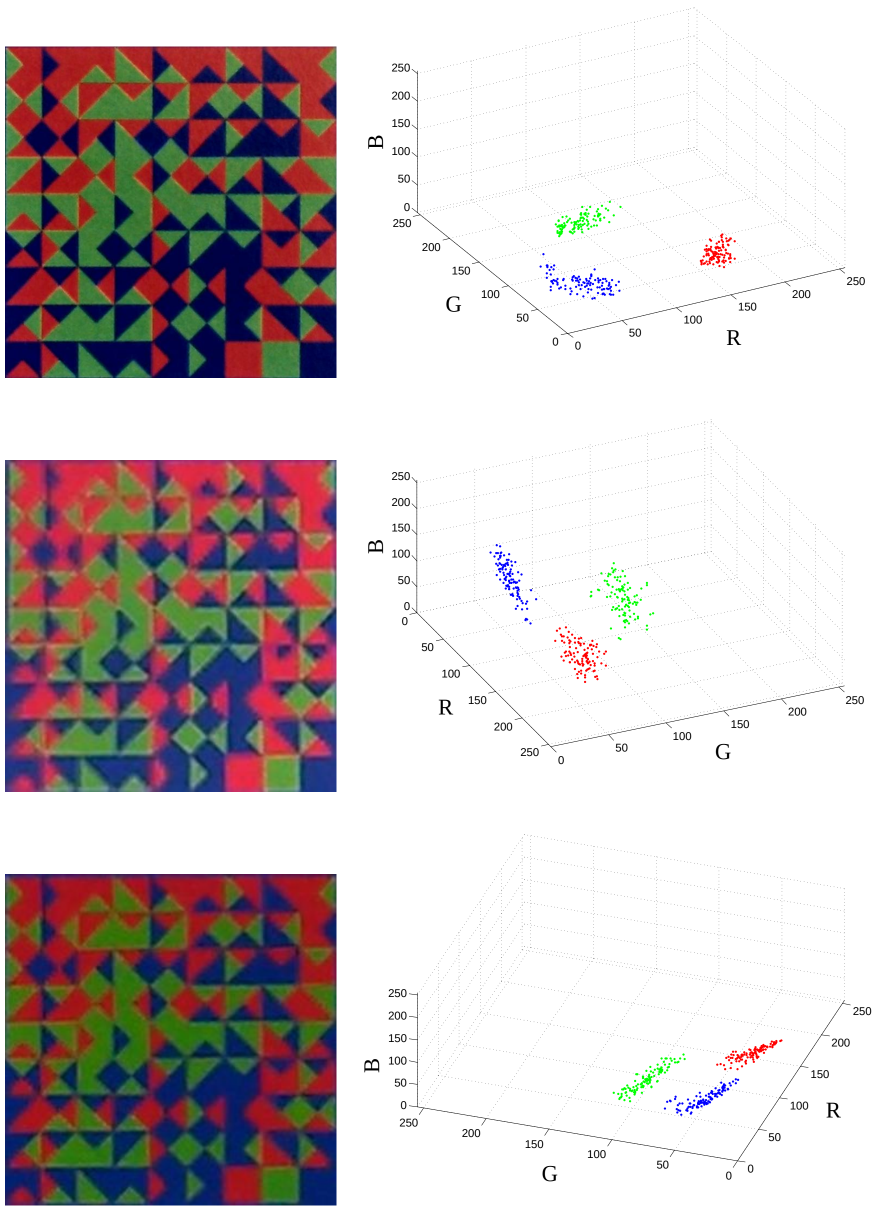


Figure 5.3: Examples of barcode images and corresponding average patch colors for barcode images with correct focus.

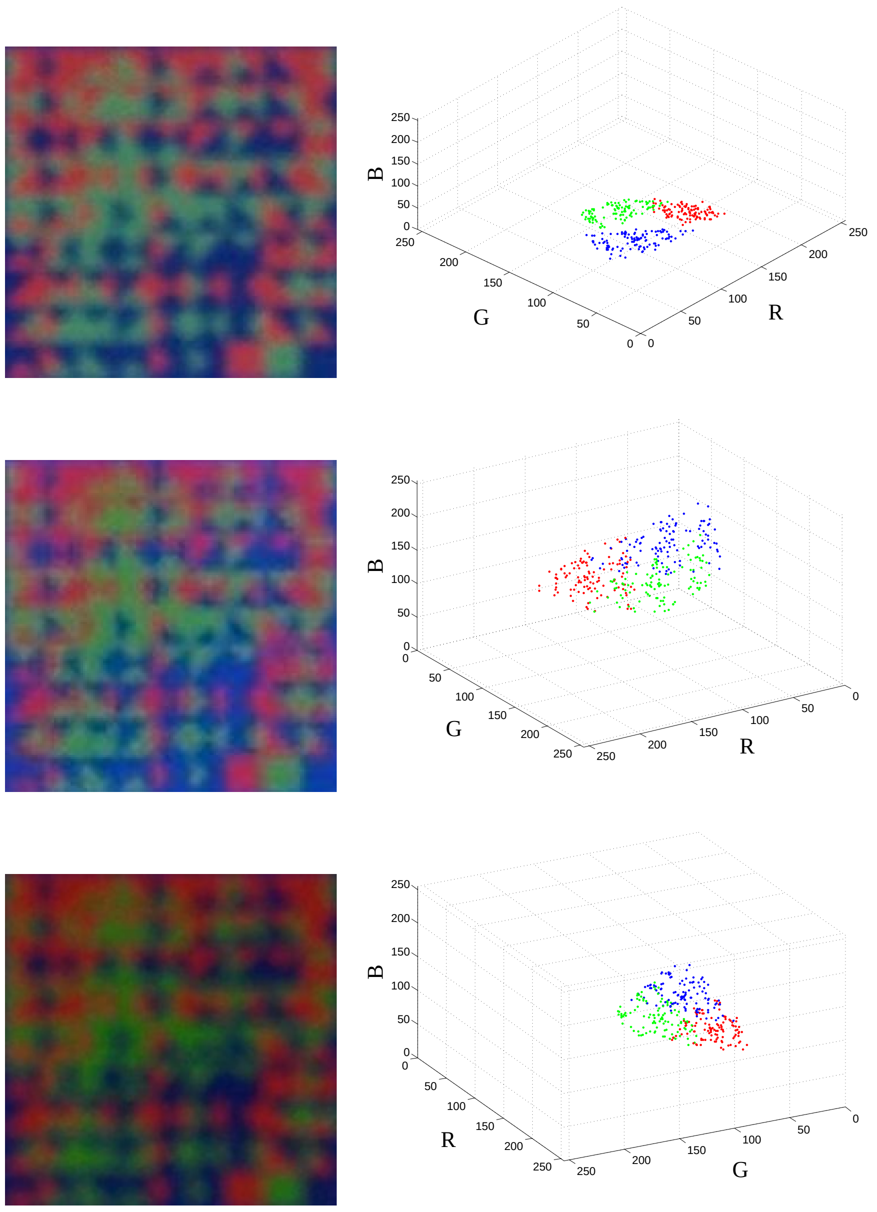


Figure 5.4: Examples of barcode images and corresponding average patch colors in presence of blur.

described in Equation 5.3 for left, top, right, and bottom triangular patches in a QST.

$$\begin{array}{cccc} & w_1 & w_2 & w_3 \\ \text{left} & w_l & (w_r + w_t)/2 & (w_r + w_b)/2 \\ \text{top} & w_t & (w_r + w_b)/2 & (w_l + w_b)/2 \\ \text{right} & w_r & (w_l + w_b)/2 & (w_l + w_t)/2 \\ \text{bottom} & w_b & (w_l + w_t)/2 & (w_r + w_t)/2 \end{array} \tag{5.3}$$

Our assumption is that the amount of blur induced in different directions by $\mathbf{w}$ are less than certain value and constrained to be uniform across the barcode image if the blur is assumed to be uniform as well.

In our decoding approach, rather than decoding the individual patches independently, we decode the colors of the patches that best justifies observed colors of all patches. We know that each patch affects the colors of all its neighboring patches. For this reason, we cannot determine the color of an individual patch without considering all patches in the barcode. An iterative technique that decodes the colors of all patches across the barcode image by minimizing the overall observation error (decomposed in a sum of individual observation errors) can solve this decoding problem. The following iterative algorithm summarizes the decoding a color barcode:

1. Determine the colors $\mathbf{e}_k$ (for $k = 1 \ldots N$) of the color palette and measure the color $\bar{\mathbf{c}}_i$ of individual patches.

2. Initialize the color $\mathbf{c}_i$ of the patch i by assigning a color from the palette with

index k that minimizes the distance to $\bar{\mathbf{c}}_i$

$$\min_k \|\bar{\mathbf{c}}_i - \mathbf{e}_k\| \tag{5.4}$$

3. Select a set of values for $\mathbf{w}$ and compute the linear coefficients $\mathbf{v}$ using Eq. 5.3 and 5.2.

4. Compute the new color of each patch using Eq. 5.1

$$\hat{\mathbf{c}}_i = (\mathbf{c}_i - (w_1\mathbf{c}_{i,1} + w_2\mathbf{c}_{i,2} + w_3\mathbf{c}_{i,3}))/w \tag{5.5}$$

5. Update the color $\mathbf{c}_i$ of the patch i by assigning a color from the palette with index k that minimizes the distance to $\hat{\mathbf{c}}_i$

$$\min_k \|\hat{\mathbf{c}}_i - \mathbf{e}_k\| \tag{5.6}$$

6. Compute the residual error $h = \hat{\mathbf{c}}_i - \mathbf{c}_i$.

7. Repeat steps 4 to 6 until no changes in the residual error h observed.

8. Repeat steps 3 to 7 for a new set of values for $\mathbf{w}$.

9. Select $\mathbf{w}$ and the colors assigned to the patches such that the sum of the squared residuals $\sum_j h_j^2$ is minimized.

One possible solution for this optimization problem is exhausive search. Exhausive search systematically enumerates all possible candidates for the solution. The candidate that minimizes the sum of the squared residulas at setp 9 is the solution. The exhausive search visits all grid points in a bounded region. The main problem with this approach is that it uses a step size to enumerate which makes it hard to decide how fine the grid should be. If the grid is too large, then the optimum solution may be missed. If the grid is too small, computational cost explodes exponentially as a grid with M points in one dimension will have M^D points in D dimensions. Therefore, exhausitive search can be used when the solution is bounded and a larger step size to enumerate possible candidates is possible. In our case, a large step size may not lead to an optimal solution since the amount of color mixing from one barcode image to another can be different. Using alternative solvers such as global optimization tools may lead to faster and more accurate solution. The global optimization methods we considered for our algorithm include simulated annealing (SA), pattern search (PS), and genetic algorithm (GA). A summary of these methods can be found in Appendix.

5.4 Experiments

5.4.1 Data Sets

For our experiments, we created a color barcode of 3.2×3.2 cm^2 with $N = 3$. This barcode has 324 patches. The last three QST represent the color palette of the barcode. We printed this test barcode on a paper by a regular printer. Our camera is a 8 Megapixel smart phone camera (DROID RARZR M) with JPEG image format. The user has no control over the automated non-linear (e.g. gamma) and white-point correction applied by the camera. We took 715 images from our barcode under realistic illumination conditions, viewing angles (ranging from $\pm 30°$ with respect to the normal

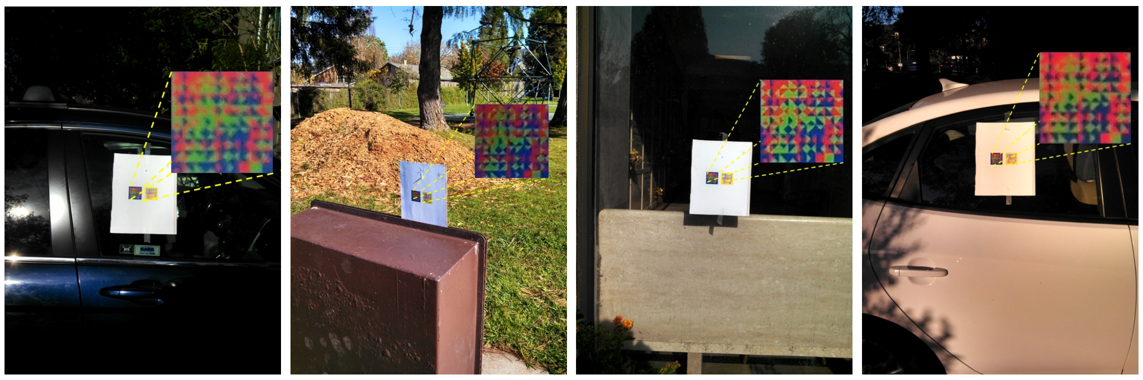


Figure 5.5: Examples of barcode images after perspective projection correction taken at multiple distances and viewing angle.

to the barcode surface), and multiple distances ranging approximately between 55 to 140 cm. The quality of barcode images can vary from one image to another due to motion, incorrect focus, and camera-barcode distance, viewing angle, JEPG format, and automated corrections by the camera (Figure 5.5).

The number of images taken in fronto-parallel view with correct focus is 351 and in presence of blur is 105. The number of images under perspective projection with correct focus is 224 and in presence of blur is 35. We extracted the barcodes from the images by selecting a rectangular region of interest manually containing the barcode. Canny edge detection is used to identify the edges of the barcode which can be used to undo the perspective projection of the barcode image to a square shaped barcode. We extracted each patch based on the knowledge about the pixel dimension of the barcode image and the number of the patches arrangement in the barcode.

The choice of N colors to build the barcode is relatively trivial. We have chosen the colors of the palette as Red, Green, and Blue. These colors have the largest distance from each other in the RGB color space. Red, Green, and Blue are used as the primary for additive combination of colors, as in superimposing of projected lights or in CRT displays as well.

5.4.2 Global Optimization Methods

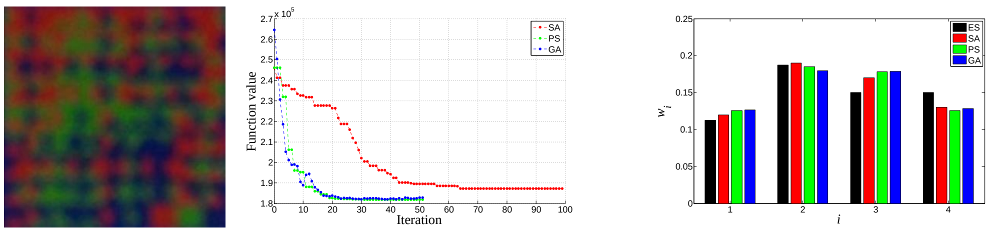
We also evaluate our decoding algorithm using three different global optimization methods. The objective function is the function describe in Sec. 5.3 is the function that we want to minimize. Global Optimization algorithms attempt to find the global minimum of the objective function. These methods works well with objective functions that are not differentiable, or are not even continuous. They do not require any information about the gradient of the objective function.

Figure 5.6 shows six barcode images with their objective function values of the best point at each iteration. The solution $\mathbf{w}$ for each method is also shown. The number of objective function evaluation at each interartion for each method is different. Typically, the objective function values drop rapidly at the early iterations and then level off approaching the optimal value.

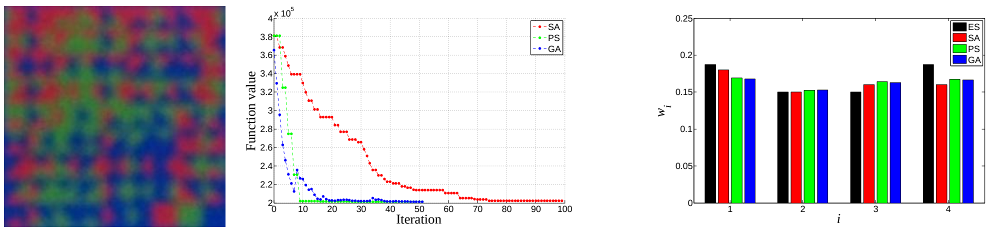
Table 5.2 shows the number of patches decoded incorrectly for each method. Typically, all three methods converge quickly with uniform distribution of w_i as expected if the barcode image is in-focus and taken from a short distance as shown in image f). All optimization methods including exhaustive serach produces less number of patches decoded incorrectly than the minimum distance method.

Figure 5.6: Left: various color barcodes. Middle: objective functions vs. number of iterations. Right: linear coeffiecents for various solvers.

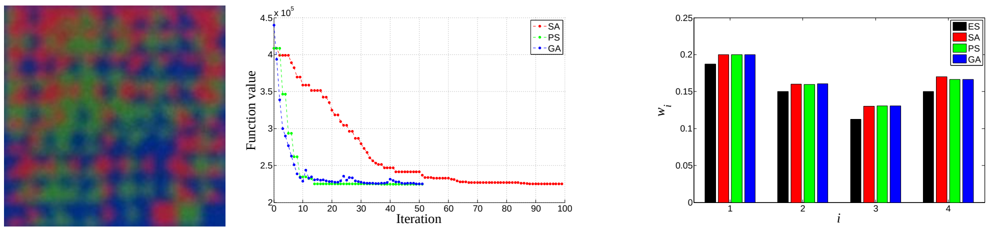
a)



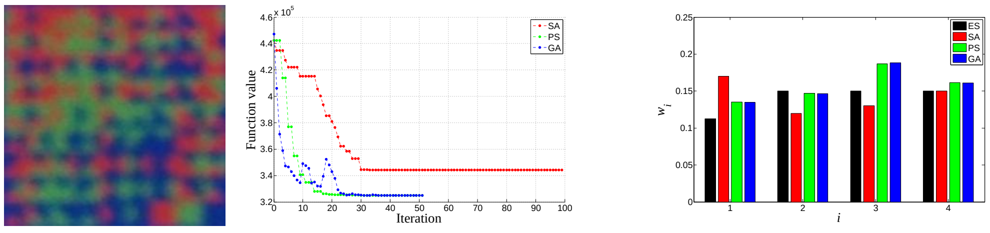
b)



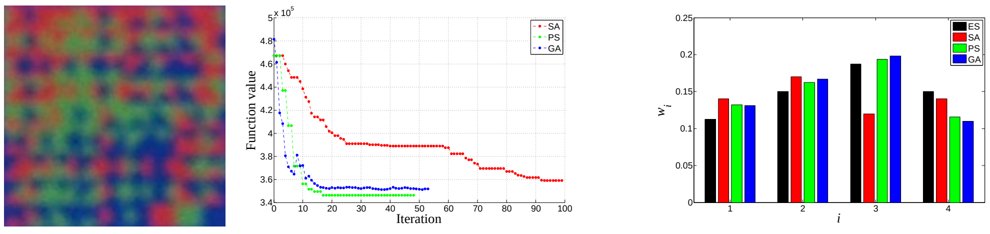
c)



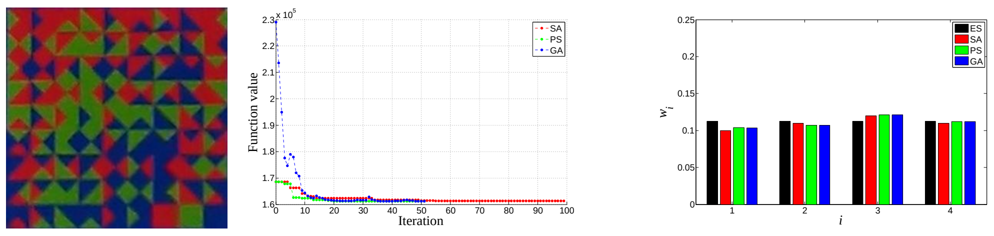
d)



e)



f)



	Patch Size [pixel]	ES	SA	PS	GA	MD
a)	**64**	3	2	2	2	19
b)	**49**	0	0	0	0	11
c)	**49**	0	0	0	0	24
d)	**42**	9	8	6	6	26
e)	**42**	7	10	4	5	25
f)	**64**	0	0	0	0	0

Table 5.2: Comparative results in terms of number of patches decoded incorrectly for the barcode images in Figure 5.6. The barcode has a total of 324 patches with 12 reference color patches. Methods: exhaustive search (ES), simulated annealing (SA), pattern search (PS), genetic algorithm (GA), minimum distace (MD).

5.4.3 Performance Evaluation

We evaluated our algorithm using exhaustive search, three global optimization methods, and the minimum distance method using all 715 barcode images. We used the average patch pixels in our methods and the average of 3×3 central patch pixels in the minimum distance method to compare with the reference colors in the palette. Using all pixels in the patch would lead to much higher decoding error in minimum distance method due to mix of colors on the patch boarders. In contrast using 3×3 central patch pixels in our method would lead to higher decoding error since the average color of central patch pixels does not represent the mix color resulted from the color of the patch itself and the colors of its three neighboring patches.

To determine the search range in $\mathbf{w}$, we run the algorithm on 100 images randomly selected from our data set. We selected a range of $[0, 0.5]$ for each of elements of $\mathbf{w}$ with a step size of 0.0333 to reduce the computation time due to a 4 degrees of freedom. Then, we select the range of $\mathbf{w}$ to $[0, 0.2]$ for our methods by determining the min and max of all elements of $\mathbf{w}$ found for these 100 images. We chose the same step size of 0.0333 for exhausive search method, which tries 2401 different $\mathbf{w}$ to find the optimum solution.

The implementation of our algorithm is in Matlab. We use the Matlab Global Optimization Toolbox for simulaed annealing, pattern search and genetic algorithm. An optimized and real-time implementation of a chosen optimization method on a cellphone is left for future work. Based on our observation, the number of iterations at step 7 described in Sec. 5.3 varies between 1 and 6 with an average of 3 iterations when we use exhaustive search method. We used a laptop (Intel i5-2520M CPU @ 2.50GHz) to measure the execution time to compute the $\mathbf{w}$ for our barcode. The execution time for exhasitive search, simulated annealing, pattern search, and genetic algorithm are approximately 73, 11, 9, and 49 seconds respectively using our Matlab implementation.

To evaluate and compare the performance of our algorithm using different solvers with the minimum distance method, we define the empirical probability of decoding error $P(p)$ for each patch size p (number of pixels per patch) as described in chapter 4. $P(p)$ is defined as the number of color patches, which were incorrectly decoded, divided by the number of the barcode images (with the same patch size), and by the number of color patches in the barcode (312). Figure 5.7 shows the empirical probability of decoding error vs. the number of pixels per patch.

As described in chapter 5, the probability of decoding error for B bits, assuming the decoding errors for the individual colors in the barcode are statistically independent events, is:

$$P_B(B, p) = 1 - (1 - P(p))^n \tag{5.7}$$

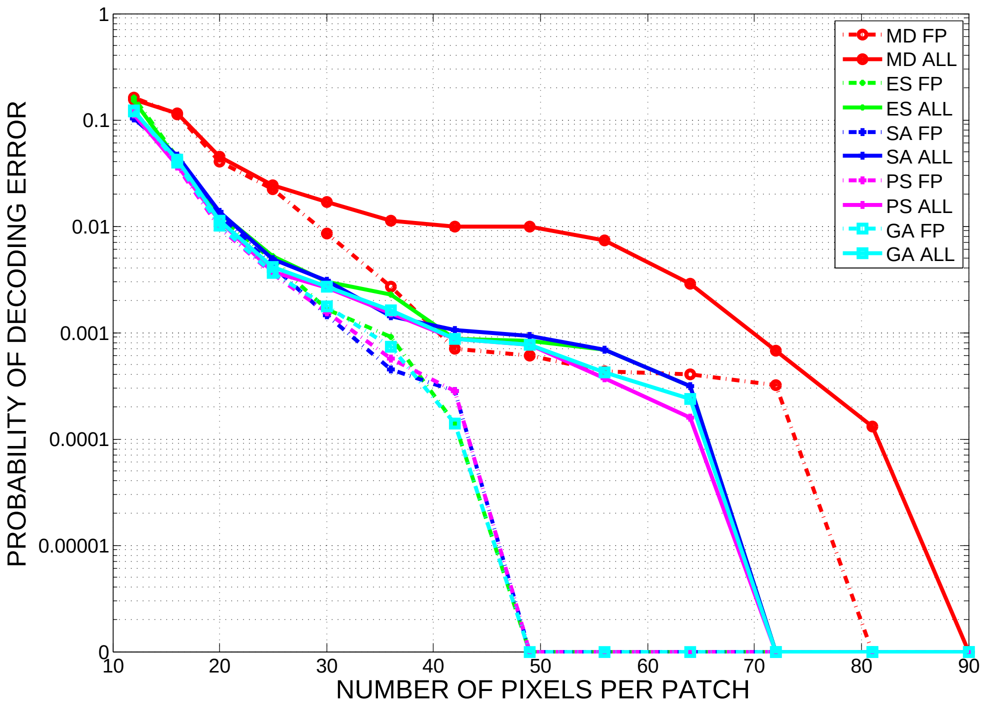
$$n = \lceil B / \log_2 N \rceil \tag{5.8}$$

The probability of decoding error increases with the number of bits. Figure 5.8 shows the probability of decoding error of $n = 81$ color patches (using $N = 3$) that represents $B = 128$ bits (which represents an Internet Protocol Version 6 address (IPv6)) vs. the approximate camera-barcode distance. The approximate camera-barcode distance can be determined by the assumption that the ratio of the barcode height on the sensor and the real barcode height is the same as the ratio of the focal length and the distance to the barcode. Therefore, the approximate camera-barcode distance can be formulated as

$$d = f \frac{bH}{sG} \tag{5.9}$$

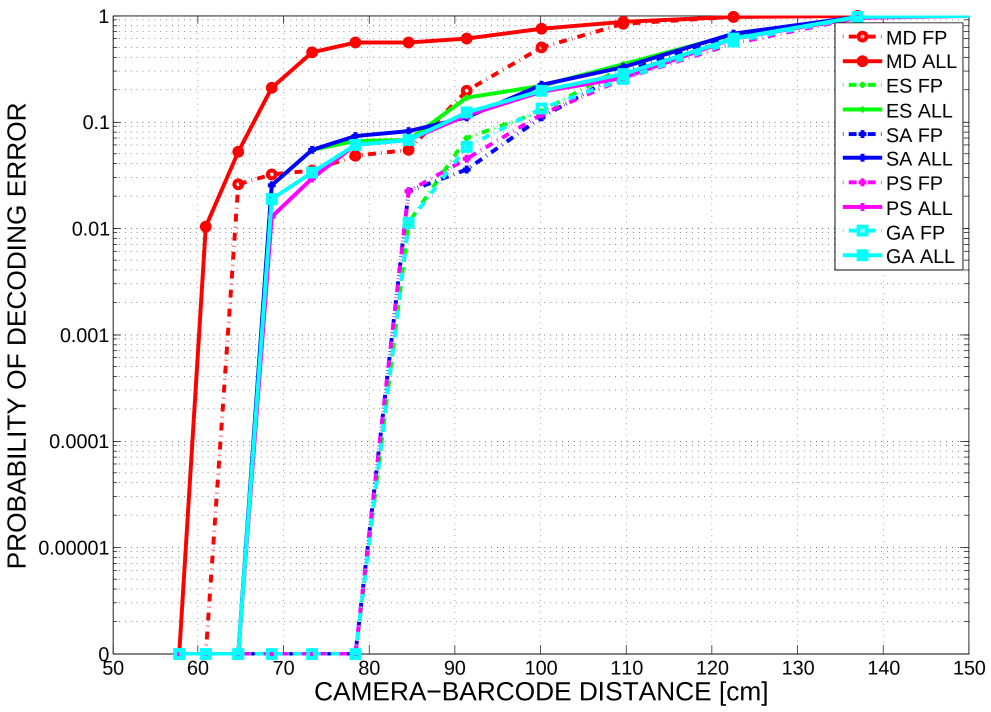
where f is the focal length of the camera, b is the physical height of the barcode, s is the sensor height, G is the barcode height in the image in pixels, and H is the image height in pixels.

The basic conclusion from our results (Figure 5.7 and 5.8) is that, for the same



c

Figure 5.7: The empirical probability of decoding error vs. the number of pixels per patch. Dashed line: images taken in fronto-parallel view with correct focus. Solid line: all images in the data set used. Methods: minimum distace (MD), exhaustive search (ES), simulated annealing (SA), pattern search (PS), genetic algorithm (GA).



c

Figure 5.8: The probability of decoding error of 81 patches ($B = 128$ bits) vs. the approximate barcode-camera distance. Dashed line: images taken in fronto-parallel view with correct focus. Solid line: all images in the data set used. Methods: minimum distace (MD), exhaustive search (ES), simulated annealing (SA), pattern search (PS), genetic algorithm (GA).

color barcode, our algorithm can decode far more robustly in presence of blur than the minimum distance method. The performance of all three optimization methods and the exhaustive search is overall similar. However, the pattern search method takes less time to decode our color barcode and shows slightly better performance in presence of blur.

As described in previous chapter, one way to increase the information density within a given barcode space is to increase the number of colors. This requires larger number of pixels per patch (short camera-barcode distance) to ensure robust decoding. Our method described in Sec. 5.3 is able to decode the same barcode with smaller patch size which means that more patches can be packed in the same barcode space, thus increasing the information density.

To put our results in context, the minimum number of pixels per patch to decode robustly (with zero probability error as shwon in Figure 5.7) is 81 in minimum distance method and 49 in our method when the barcode is captured from a fronto-parallel view with correct focus. This indicates that a color barcode system that uses our decoding approach requires 40% less space to encode 128 bits using only $N = 3$ colors. A color barcode system that uses the minimum distance method with $N = 7$ reference colors can encode 128 bits in the same space as our decoding method. Note that the color palette of multiple copies of $N = 7$ colors occupies more space on a barcode and cannot be used to encode information.

In more difficult situation, where the barcode image is captured in presence of blur and/or from a viewing angle ranging $\pm 30^{\circ}$, a robust decoding can be achieved with 90 pixels per patch using minimum distance method and 72 pixels per patch using our method. In this case, our method requires 20% less space to encode 128 bits. The minimum distance method using $N = 4$ reference colors could encode 128 bits in the same space as our decoding method.

5.5 Conclusions

We have presented a new algorithm for color barcode decoding that can handle barcode images in presence of blur using a smart phone camera images. Rather than clustering the colors of the barcode and decoding the patches using minimum distance method, our iterative algorithm decodes the colors of all patches across the barcode by minimizing the overall observation error. The global optimization methods such pattern search and simulated annealing find the optimal solution faster than exhaustive search and genetic algorithm. These two algorithms may be considered for real time implementation on a cell phone in future.

Our decoding approach enables higher information rate in a given space by allwoing smaller patch size compared to mainstream methods. We show that our method enables to encode 20% to 40% more information in presence of color mixing using only three colors for the same barcode area than the technologies using minimum distance method.

Chapter 6

Future Work

In this thesis, analysis, and justification for novel and robust algorithms, applicable to color target detection and color information access problems are developed. Experiments on real data demonstrate the effectiveness of the presented methods, and its practicality for real applications. The author hopes that the work presented here serves as a way for the future researchers in this field of research. In this section, we outline a few of the open questions related to the research presented in this thesis. In particular, we offer possible extensions to some of the chapters.

- In Chapter 2, we introduced a new detection algorithm that is suitable for multi-color, pie-shaped markers. The proposed algorithm is computationally very light and has excellent performance in terms of detection rate and false alarm rate. Compared with a popular grayscale marker (ARToolKit), our color markers enable more robust detection in various realistic conditions for a similar processing time. One possible extension to our detection algorithm is to improve the performance of the algorithm in low light conditions. This may be a crucial requirment in a cell phone-based system that uses environmental labeling for blind wayfinding in an environment with low light conditions.

- In Chapter 4, we introduced a new color barcode decoding approach that requires few reference colors attached to the color barcode and can handle the presence of specular reflection. Our experiments have shown that, by carefully selecting a set of reference colors, it is possible to achieve a high information rate at a low probability of decoding error with relatively low computational complexity. The robustness of our algorithm has not been proven when the barcodes are printed by different printers. Therefore, one important extension for our method is to ensure robust decoding of color barcodes that are printed from different printers.

- In Chapter 5, we have presented a new algorithm for color barcode decoding that can handle barcode images in presence of uniform blur across the barcode image using only three reference colors ensuring higher information density than mainstream approaches. One possible extension to our algorithm is robust decoding in presence of non-uniform blur across the barcode image. This could improve the decoding performance when barcode images are under perspective projection or barcodes printed on non-planar surfaces. Investigation of real time implmentation considering three global optimization methods would be a possible extension to our algorithm as well.

Bibliography

[1] ARToolKitPlus for pose tracking on mobile devices, author=Wagner D, Schmalstieg D, booktitle=Proc. Computer Vision Winter Workshop, year=2007.

[2] ISO 18004:2006. Qr code 2005 bar code symbology specification.

[3] Mark A. Abramson. *Pattern Search Filter Algorithms for Mixed Variable General Constrained Optimization Problems.* PhD thesis, Department of Computational and Applied Mathematics, Rice University, August 2002.

[4] Mark A Abramson, Charles Audet, John E Dennis Jr, and Sébastien Le Digabel. OrthoMADS: A deterministic MADS instance with orthogonal directions. *SIAM Journal on Optimization*, 20(2):948–966, 2009.

[5] Charles Audet and John E Dennis Jr. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903, 2003.

[6] Homayoun Bagherinia and Roberto Manduchi. Robust real-time detection of multicolor markers on a cell phone. *Journal of real-time image processing*, 2011.

[7] Homayoun Bagherinia and Roberto Manduchi. A theory of color barcodes. In *In*

IEEE Color and Photometry in Computer Vision (CPCV 2011) Workshop, pages 806–813. IEEE, 2011.

[8] Homayoun Bagherinia and Roberto Manduchi. High information rate and efficient color barcode decoding. In *In International Workshop on Color and Photometry in Computer Vision (CPCV 2012)*, pages 482–491. Springer, 2012.

[9] Homayoun Bagherinia and Roberto Manduchi. Color barcode decoding in the presence of specular reflection. In *In 4th Color and Photometry in Computer Vision (CPCV 2014) Workshop*. Springer, 2014.

[10] Homayoun Bagherinia and Roberto Manduchi. A novel approach for color barcode decoding using smart phones. In *International Conference on Image Processing (ICIP 2014)*. IEEE, 2014.

[11] R. Basri, T. Hassner, and L. Zelnik-Manor. Approximate nearest subspace search with applications to pattern recognition. In *Proc. IEEE Computer Vision and Pattern Recognition*, 2007.

[12] 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.

[13] David H. Brainard and William T. Freeman. Bayesian color constancy. *Journal of the Optical Society of America A*, 14:1393–1411, 1997.

[14] O. Bulan, V. Monga, and G. Sharma. High capacity color barcodes using dot

orientation and color separability. In *Proc. SPIE-IS&T Electronic Imaging*, volume 7254, 2009.

[15] Xiangrong Chen and Alan L Yuille. Detecting and reading text in natural scenes. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. IEEE, 2004.

[16] Xiangrong Chen and Alan L Yuille. A time-efficient cascade for real-time object detection: With applications for the visually impaired. In *Proceedings of the IEEE Workshop on Computer Vision for the Visually Impaired, IEEE Computer Society*, page 28, Washiington, DC, USA, 2005. IEEE.

[17] Youngkwan Cho and Ulrich Neumann. A multi-ring color fiducial system and an intensity-invariant detection method for scalable fiducial-tracking augmented reality. In *VRAIS '98: Proceedings of the Virtual Reality Annual International Symposium, IEEE Computer Society*. Citeseer, 1998.

[18] David Claus and Andrew W Fitzgibbon. Reliable fiducial detection in natural scenes. In *Computer Vision-ECCV 2004*, pages 469–480. Springer, 2004.

[19] Andrew R Conn, Nicholas IM Gould, and Philippe Toint. A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–572, 1991.

[20] Andrew R Conn, Nicholas IM Gould, and Philippe Toint. A globally convergent

augmented lagrangian algorithm for optimization with general constraints and simple bounds. *Mathematics of Computation*, 66(217):261–288, 1997.

[21] Andrew R Conn, Nicholas IM Gould, and Philippe Toint. A globally convergent augmented lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds. *Mathematics of Computation*, 66(217):261–288, 1997.

[22] James Coughlan and Roberto Manduchi. Color targets: Fiducials to help visually impaired people find their way by camera phone. *EURASIP Journal on Image and Video Processing*, 2007, 2007.

[23] James Coughlan and Roberto Manduchi. Functional assessment of a camera phone-based wayfinding system operated by blind and visually impaired users. *International Journal on Artificial Intelligence Tools*, 18(03):379–397, 2009.

[24] ER Dixon. Spectral distribution of australian daylight. *JOSA*, 68(4):437–450, 1978.

[25] Hany Farid. Blind inverse gamma correction. *Image Processing, IEEE Transactions on*, 10(10):1428–1433, 2001.

[26] Mark Fiala. ARTag, an improved marker system based on ARToolkit. Technical Report NRC: 47419, National Research Council Canada, 2004.

[27] G.D. Finlayson, M.S. Drew, and B.V. Funt. Diagonal transforms suffice for color constancy. In *Proceedings of Fourth International Conference on Computer Vision*, pages 164–171, 1993.

[28] Graham D Finlayson, Mark S Drew, and Brian V Funt. Color constancy: generalized diagonal transforms suffice. *J Opt Soc Am A*, 11(11):3011–3019, 1994.

[29] Graham D. Finlayson, Steven D. Hordley, and Paul M. Hubel. Color by correlation: A simple, unifying framework for color constancy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23:1209–1221, November 2001.

[30] D. A. Forsyth. A novel algorithm for color constancy. *Int. J. Comput. Vision*, 5:5–36, September 1990.

[31] Orazio Gallo and Roberto Manduchi. Reading 1d barcodes with mobile phones using deformable templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2010. In press.

[32] David E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, 1989.

[33] A. Grillo, A. Lentini, M. Querini, and G.F. Italiano. High capacity colored two dimensional codes. In *Proc. Int. Multiconf. on Comp. Science Inf. Tech.*, 2010.

[34] T. Han, C. Cheong, N. Lee, and E. Shin. Machine readable code image and method of encoding and decoding the same. U.S. Patent 7020327, 2000.

[35] Glenn Healey and David Slater. Global color constancy: recognition of objects by use of illumination-invariant properties of color distributions. *J Opt Soc Am A*, 11(11):3003–3010, 1994. `http://josaa.osa.org/abstract.cfm?URI=josaa-11-11-3003`.

[36] Lester Ingber. Adaptive simulated annealing (asa): Lessons learned. *Invited paper to a special issue of the Polish Journal Control and Cybernetics on "Simulated Annealing Applied to Combinatorial Optimization."*, 1995. Available from `http://www.ingber.com/asa96_lessons.ps.gz`.

[37] D.B. Judd, D.L. MacAdam, G. Wyszecki, H.W. Budde, H.R. Condit, S.T. Henderson, and J.L. Simonds. Spectral distribution of typical daylight as a function of correlated color temperature. *JOSA*, 54(8):1031–1040, 1964.

[38] H. Kato, K.T. Tan, and D. Chai. Novel colour selection scheme for 2D barcode. In *Proc. International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS 2009)*, 2009.

[39] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on*, pages 85–94. IEEE, 1999.

[40] Blanding B May R Kato H, Billinghurst M. Ar-toolkit. Technical report, Hiroshima City University, 1999.

[41] Tamara G Kolda, Robert Michael Lewis, and Virginia Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM review*, 45(3):385–482, 2003.

[42] Tamara G Kolda, Robert Michael Lewis, and Virginia Torczon. A generating set

direct search augmented lagrangian algorithm for optimization with a combination of general and linear constraints. Technical Report SAND2006-5315, 2006.

[43] Wonwoo Lee and Woontack Woo. Real-time color correction for marker-based augmented reality applications. In *International Workshop on Ubiquitous Virtual Reality*, pages 32–25, 2009.

[44] Robert Michael Lewis, Anne Shepherd, and Virginia Torczon. Implementing generating set search methods for linearly constrained minimization. *SIAM Journal on Scientific Computing*, 29(6):2507–2530, 2007.

[45] Laurence T. Maloney and Brian A. Wandell. Color constancy: a method for recovering surface spectral reflectance. In Martin A. Fischler and Oscar Firschein, editors, *Readings in computer vision: issues, problems, principles, and paradigms*, pages 293–297. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[46] L.T. Maloney. Evaluation of linear models of surface spectral reflectance with small numbers of parameters. *J. Opt. Soc. Amer.-A*, (3(10)):1673–1683, 1986.

[47] Wandell B Maloney L. A computational model of color constancy. *Journal of the Optical Society of America*, 1(1):29–33, 1986.

[48] Corey Manders and Steve Mann. True images: A calibration technique to reproduce images as recorded. In *Multimedia, 2006. ISM'06. Eighth IEEE International Symposium on*, pages 712–715. IEEE, 2006. `http://doi.ieeecomputersociety.org/10.1109/ISM.2006.155`.

[49] Roberto Manduchi, Sri Kurniawan, and Homayoun Bagherinia. Blind guidance using mobile computer vision: a usability study. In *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility*, pages 241–242. ACM, 2010.

[50] David H Marimont and Brian A Wandell. Linear models of surface and illuminant spectra. *JOSA A*, 9(11):1905–1913, 1992.

[51] Erik Miller and Kinh Tieu. Color eigenflows: Statistical modeling of joint color changes. In *Proc. of International Conference on Computer Vision(ICCV)*, volume 1, pages 607–614, 2001.

[52] Leonid Naimark and Eric Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, page 27. IEEE Computer Society, 2002.

[53] Devi Parikh and Gavin Jancke. Localization and segmentation of a 2d high capacity color barcode. In *Applications of Computer Vision, 2008. WACV 2008. IEEE Workshop on*, pages 1–6. IEEE, 2008. `http://doi.ieeecomputersociety.org/10.1109/WACV.2008.4544033`.

[54] Jussi PS Parkkinen, J Hallikainen, and T Jaaskelainen. Characteristic spectra of munsell colors. *JOSA A*, 6(2):318–322, 1989.

[55] Songwen Pei, Guobo Li, and Baifeng Wu. Codec system design for continuous color

barcode symbols. In *Proceedings of the 2008 IEEE 8th International Conference on Computer and Information Technology Workshops*, pages 539–544, Washington, DC, USA, 2008. IEEE Computer Society.

[56] Marco Querini and Giuseppe F Italiano. Color classifiers for 2d color barcodes. In *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, pages 611–618. IEEE, 2013.

[57] Michael Rohs and Beat Gfeller. *Advances in Pervasive Computing*, chapter Using camera-equipped mobile phones for interacting with real-world objects, pages 265–271. 2004.

[58] E. Sali and D.M. Lax. Color bar code system. U.S. Patent 7210631, 2006.

[59] Daniel Scharstein and Amy J Briggs. Real-time recognition of self-similar landmarks. *Image and Vision Computing*, 19(11):763–772, 1999.

[60] Steven A Shafer. Using color to separate reflection components. *Color Research & Application*, 10(4):210–218, 1985.

[61] Steven A Shafer. Using color to separate reflection components. *Color Research & Application*, 10(4):210–218, 1985.

[62] Khurram Shafique and Mubarak Shah. Estimation of the radiometric response functions of a color camera from differently illuminated images. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 4, pages 2339–2342. IEEE, 2004. {10.1109/ICIP.2004.1421569}.

[63] D. Slater and G. Healey. What is the spectral dimensionality of illumination functions in outdoor scenes? In *Proc. CVPR.* IEEE, 1998.

[64] Chen DT Garrett WF Livingston MA State A, Hirota G. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 429–438. ACM, 1996.

[65] X. Sun and QianSheng Cheng. On subspace distance. In *Proc. ICIAR*, pages 81–89, 2006.

[66] J. Swartz T. Pavlidis and Y. Wang. Fundamentals of bar code information theory. *Computer*, 23(4):74–86, 1990.

[67] Gabriel Takacs, Vijay Chandrasekhar, Natasha Gelfand, Yingen Xiong, Wei-Chao Chen, Thanos Bismpigiannis, Radek Grzeszczuk, Kari Pulli, and Bernd Girod. Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 427–434. ACM, 2008.

[68] K.T. Tan, S.K. Ong, and D. Chai. JPEG color barcode images analysis: A camera phone capture channel model with auto-focus. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 2(4), 2009.

[69] O. Tatsuya and M. Kazuhiro. Layered two-dimensional code, creation method thereof, and read method. United States Patent Application 20090166418, 2006.

[70] Øivind Due Trier and Torfinn Taxt. Evaluation of binarization methods for document images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):312–315, 1995. `http://doi.ieeecomputersociety.org/10.1109/34.368197`.

[71] Yanghai Tsin, Robert T Collins, Visvanathan Ramesh, and Takeo Kanade. Bayesian color constancy for outdoor object recognition. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–1132. IEEE, 2001. `http://doi.ieeecomputersociety.org/10.1109/CVPR.2001.990658`.

[72] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. IEEE, 2001.

[73] Fan Wang and Roberto Manduchi. Color-constant information embedding. In *Color and Reflectance in Imaging and Computer Vision Workshop 2010*. Springer, 2010.

[74] Jingtao Wang, Shumin Zhai, and John Canny. Camera phone based motion sensing: interaction techniques, applications and performance study. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 101–110. ACM, 2006.

Appendix A

Color Image Formation and Bilinear Method

In this appendix, we review the color image formation and bilinear method [71]. For a Lambertian surface, the measured intensity ρ_c of channel $c \in [r, g, b]$ is

$$\rho_c = g \int_\lambda f_c(\lambda) s(\lambda) l(\lambda) d\lambda \tag{A.1}$$

g is the scaling factor. $f_c(\lambda)$ is the camera sensitivity function of channel c; $s(\lambda)$ and $l(\lambda)$ denote the reflectance and light spectrums respectively. λ is the wavelength. The equivalent vector representation of this equation is formulated as follows by discretizing the reflectance s, light spectrum l and camera sensitivity function f_c into N samples as column vectors

$$\rho_c = g l^T D(f_c) s \tag{A.2}$$

where $D(f_c)$ is a $N \times N$ diagonal matrix with f_c as diagonal elements. The finite dimensional linear models for both reflectance [46, 54] and illumination spectrum [37, 63], assuming the reflectance and iluumination spectrum are spanned by the column spaces of the matrices B_s and B_l respectively. The reflectance and iluumination spectrum can be reformulated as

$$\begin{aligned} s &= B_s w \\ l &= B_l v \end{aligned} \tag{A.3}$$

where $w \in \mathbb{R}^{n_w}$ and $v \in \mathbb{R}^{n_v}$ are coefficient vectors with much lower dimensionality than N. In previous research, it has been shown that both the natural light spectrum [37, 63] and reflectance [46, 54] can be accurately approximated with such low dimensional subspaces. The bilinear model for color image formation for color channel c is formulated as follows

$$\begin{aligned} \rho_c &= v^T g B_l^T D(f_c) B_s w \\ \rho_c &= v^T \phi_c \end{aligned} \tag{A.4}$$

The bilinear model for three color channels is

$$\rho = \Phi v \tag{A.5}$$

Appendix B

Simulated Annealing

In this appendix, we review the simulated annealing algorithm. The details of the simulated annealing (SA) algorithm can be found in [36]. Basically, SA models the process of heating a material followed by lowering slowly the temperature to decrease defects, thus minimizing the system energy. A new point is randomly generated at each iteration of the SA algorithm. The distance of the new point from the current point, or the extent of the search, is based on a probability distribution with a scale proportional to the temperature. The algorithm accepts all new points that lower the objective function, but also, with a certain probability, points that raise the objective function. By accepting points that raise the objective function, the algorithm avoids being trapped in local minima, and is able to explore globally for more possible solutions. An annealing schedule is selected to systematically decrease the temperature as the algorithm proceeds. As the temperature decreases, the algorithm reduces the extent of its search to converge to a minimum. The temperature is a parameter that affects the distance of a trial point from the current point and the probability of accepting a trial point with higher objective function value. Temperature decreases with a paced as the algorithm proceeds. The faster the temperature decrease, the shorter the execution time is, but the probabiltiy of finding an optimal solution decreases. The temperature is a function of the Annealing Parameter, which is a proxy for the iteration number. Annealing controlls the temperature to ensure that an optimal solution reaches. Reannealing increases the temperature after the algorithm accepts a certain number of new points, and starts the search again at the higher temperature to avoid the algorithm trapped in local minima. The algorithm selects the distance of a new point from the current point by a uniform distribution with a a step length that is equal to the current temperature. The algorithm determines whether the new point is better than the current point. If the new point is better than the current point, it becomes the next point. If the new point is worse than the current point, the algorithm can still make it the next point. The algorithm accepts a worse point based on an acceptance function:

$$\frac{1}{1+\exp^{\frac{\Delta E}{\max(T)}}} \tag{B.1}$$

where ΔE is the difference between the new objective and old objective function values and T is the current tempreture. Smaller acceptance probability is achieved by smaller temperature. A larger ΔE leads to smaller acceptance probability. The algorithm decreases the temperature and stores the best point found so far. The following function is used to update the temperature:

$$T = T_0 \times 0.95^k \tag{B.2}$$

where T_0 is the initial temprature, and k is the iteration number until reannealing. Reannealing sets the k value to lower value than the iteration number, therefore the temperature increases. k is estimated based on the value of estimated gradients of the objective function in each dimension as follows:

$$k = \log\left(\frac{T_0 \times \max(s_i)}{T \times s_d}\right) \tag{B.3}$$

where T_0 is the initial temprature, T is the current annealing, s_d is the gradient of objective function in direction d times difference of bounds in direction d.

Appendix C

Pattern Search

In this appendix, we review the pattern search algorithm. A suffiecent number of litrature can be found about pattern search (PS) [5, 21, 3, 4, 41, 42, 44]. A pattern search algorithm searches a set of points around the current point, selecting the point that makes the objective function lower than the value at the current point. There are variety of pattern search algorithms such as the generalized pattern search, the generating set search, and the mesh adaptive search. We consider the generalized pattern search for our experiements. Other pattern search variations could be considered in future work. Basically, the algorithm computes a series of points that approach an optimal solution. It searches a mesh (a set of points around the current point) at each step. The mesh is created by including the current point to a scalar multiple of a set of vectors called a pattern. If the algorithm evaluates a point in the mesh that lowers the objective function than the current point, then the new point replaces the new point at the next step of the algorithm. The algorithm uses fixed direction vectors that the pattern search algorithm uses to determine which points to search at each iteration. The algorithm uses a set of vectors v_i as a pattern to evaluate which points to search at each iteration. The set v_i is created based on the number of independent variables in the objective function, and the positive basis set. The positive basis sets in the algorithm is the maximal basis with 2N vectors. The group of vectors that create the pattern are fixed-direction vectors. In our case, there are four independent variables in the minimization problem, the default for a 2N positive basis consists of the following pattern vectors:

$$\begin{aligned} v_1 = [1\,0\,0\,0] \;\; v_2 = [0\,1\,0\,0] \;\; v_3 = [0\,0\,1\,0] \;\; v_4 = [0\,0\,0\,1] \\ v_5 = [-1\,0\,0\,0] \;\; v_6 = [0\,-1\,0\,0] \;\; v_7 = [0\,0\,-1\,0] \;\; v_8 = [0\,0\,0\,-1] \end{aligned} \tag{C.1}$$

PS searches a point among a set of points that minimizes the objective function at each iteration. This point becomes the current point if its objective function value is less than that of the current point. A set of points is created based on generating a set of vectors w_i by multiplying each pattern vector v_i by a scalar α and adding the w_i to the current point. The scalar α is multiplied by a number greater than one (e.g. 2) if the point replaces the current point and by a number smaller than one (e.g. 0.5) if the current point remains unchanged. The algorithm halts the search when the distance from the previous point to the current point is less than a value, or α is less than a tolerance value, or the difference between the objective function value at the previous point and objective function value at the current point is less than a value.

Appendix D

Genetic Algorithm

In this appendix, we review the genetic algorithm. There are sufficient number of lituratue about Genetic Algorithms [32, 19, 20]. In this section, we briefly describe the genetic algorithm we used to solve the minimization problem described in Sec. 5.3. The genetic algorithm (GA) is a method based on natural selection similar to the biological evolution process. The genetic algorithm generates a population of points and constantly modifies the current population. Basically, at each interation, the GA randomly selects points from the current population as parents to produce the children of next generation. The population gradually evolves over successive generations whose best point approaches an optimal solution. At each iteration, the genetic algorithm creates the next generation from the current population based on selection of parents, combining two parents, and random changes to individual parents. A population has high diversity if the average distance between points within the population is large. The genetic algorithm uses diversity to search a larger region. The best fitness value (objective function value) for a population is the smallest fitness value for any point in the population. The algorithm begins by generating an initial population randomly. Then, at each iteration, the algorithm uses the current population to create the children of the next generation. The algorithm selects a group of points (parents) in the current population. The algorithm usually selects points with better fitness values as parents. The GA generates three types of children for the next generation including elite, crossover, and mutation. Elite children are the points in the current generation with the best fitness values. These points automatically survive to the next generation. Crossover children are generated based on combination of a pair of parents in the current population. The crossover procedure randomly selects an entry from one of the two parents vectors and assigns it to the same coordinate of the child vector. Mutation children are generated based on introducing random changes to a single parent. This process is repeated until a termination condition has been reached. For instance, the algorithm terminates if a solution is found that satisfies minimum criteria.