Computer Science and Information Systems 11(4):1595–1615 DOI: 10.2298/CSIS131218054Q

Reliability and Data Density in High Capacity Color Barcodes ⋆

Marco Querini and Giuseppe F. Italiano

University of Rome "Tor Vergata"
via del Politecnico 1, 00133 Rome, Italy
marco.querini@uniroma2.it
italiano@disp.uniroma2.it

Abstract. 2D color barcodes have been introduced to obtain larger storage capabilities than traditional black and white barcodes. Unfortunately, the data density of color barcodes is substantially limited by the redundancy needed for correcting errors, which are due not only to geometric but also to chromatic distortions introduced by the printing and scanning process. The higher the expected error rate, the more redundancy is needed for avoiding failures in barcode reading, and thus, the lower the actual data density. Our work addresses this trade-off between reliability and data density in 2D color barcodes and aims at identifying the most effective algorithms, in terms of byte error rate and computational overhead, for decoding 2D color barcodes. In particular, we perform a thorough experimental study to identify the most suitable color classifiers for converting analog barcode cells to digital bit streams. To accomplish this task, we implemented a prototype capable of decoding 2D color barcodes by using different methods, including clustering algorithms and machine learning classifiers. We show that, even if state-of-art methods for color classification could be successfully used for decoding color barcodes in the desktop scenario, there is an emerging need for new color classification methods in the mobile scenario. In desktop scenarios, our experimental findings show that complex techniques, such as support vector machines, does not seem to pay off, as they do not achieve better accuracy in classifying color barcode cells. The lowest error rates are indeed obtained by means of clustering algorithms and probabilistic classifiers. From the computational viewpoint, classification with clustering seems to be the method of choice. In mobile scenarios, simple and efficient methods (in terms of computational time) such as the Euclidean and the K-means classifiers are not effective (in terms of error rate), while, more complex methods are effective but not efficient. Even if a few color barcode designs have been proposed in recent studies, to the best of our knowledge, there is no previous research that addresses a comparative and experimental analysis of clustering and machine learning methods for color classification in 2D color barcodes.

Keywords: color, classification, barcode.

1. Introduction

Barcodes are optical machine-readable representations of data, capable of storing digital information. Barcode data are represented as a sequence of bytes, which are then mapped

⋆ A preliminary version of this paper was presented at the Federated Conference on Computer Science and Information Systems [19].

to analog signals (in this case, barcode elements) and transmitted over a printing and scanning (Print&Scan) channel which introduces noise, distortions and interference, corrupting the transmitted signal (in this case, the barcode image after scanning). At the receiver, the distorted barcode is mapped back to bytes. Unfortunately, the received binary information is often only an estimate of the transmitted binary information. Indeed, byte errors may result due to the amount of noise encountered in the transmission. Because noise and distortions always occur in practice, as a result barcode reading algorithms have to cope necessarily with errors, and the trade-off between reliability and data density of barcodes is a significant design consideration. To cope with errors, redundancy is added by channel coding, which is a viable method to increase reliability in a noisy communication channel (which in our case is represented by the Print&Scan channel) at the price of reducing the information rate. The higher the expected number of errors and the redundancy needed for coping with it, the lower the actual data rate (in our case, the barcode data density).

Traditional barcodes, referred to as one-dimensional (1D) barcodes, represent data by varying the widths and spacings of parallel lines. The amount of digital information stored in 1D barcodes is limited and can be only increased by laying out multiple barcodes. This approach has many negative effects, however, such as enlarged barcode areas, more complex reading operations, and increased printing costs. For this reason, the barcode technology has been deploying geometric patterns (such as squares, dots, triangles, hexagons) in two dimensions: such barcodes are referred to as bidimensional (2D) codes. Both the increasing demand for higher density barcodes and the wide availability of onboard cameras in mobile devices has motivated the need for 2D color barcodes, such as the colored DataGlyphs developed at Xerox Parc [10], the High Capacity Color Barcode (HCCB) developed at Microsoft Research [15, 16], the high capacity color barcode technique proposed in [3], and HCC2D, the High Capacity Colored 2-Dimensional code [8, 17, 18]. Color barcodes generate each module of the data area with a color selected from 2^n-ary schemes (e.g., 4-ary color schemes encoding 2 $bit/module$ or 8-ary color schemes encoding 3 $bit/module$), where a module (or cell) is the atomic information unit of a 2D barcode.

Since black and white codes encode 1 $bit/module$, in principle the data density of a color barcode can be twice (4 colors) or three times (8 colors) as much as the data density of the corresponding black and white barcode. However, the actual capacity depends on the amount of redundancy added to the barcode data for correcting errors, which occur due to both geometric and chromatic distortions introduced by the Print&Scan channel. Since colors are more sensitive to the distortions introduced by the channel, the measured error rate of color barcodes can be significantly larger than the measured error rate of black and white barcodes, all other conditions being equal (i.e., when all barcodes are generated, printed and scanned under same conditions, such as module size, amount of redundancy, printing and scanning resolutions). In our experiments, under the same operating conditions, black and white QR codes had an average byte error rate of roughly 2% while their 4-color counterpart (HCC2D codes) had an average byte error rate of roughly 10%. In this framework, the higher error rates of color barcodes can be mitigated by the use of larger redundancies in the coding, which in turn may reduce substantially the higher data densities potentially offered by color barcodes, thus reducing their benefits. For instance, in order to tolerate a byte error rate of 2%, we need to reserve at least 4% of the barcode area for an error correction code (such as Reed Solomon), thus obtaining less than 96%

for its data density, while a 10% byte error rate implies that at least 20% of the barcode area must be used for error correction, reducing its data density to less than 80%.

In this paper we tackle this problem by designing and experimentally evaluating algorithms for retrieving digital data from color cells undergoing chromatic distortions (due to printing and scanning), so as to minimize their error rates. In particular, we perform an experimental study of the practical performance of several color classifiers and clusterers for converting analog barcode cells to digital bit streams. This allows to identify the most effective algorithms for decoding color barcodes in terms of their error rate and their total running times. Moreover, we investigate the trade-off between redundancy and data capacity for 2D color barcodes. This allows to optimize the data storage, addressing the need for high density barcodes (capable of storing as much information as possible in as small an area as possible).

To accomplish this task, we have developed a prototype capable of using different algorithms for color classification. We have chosen algorithms so that they are representative of general classes, such as minimum distance classifiers, clustering, decision trees, community detection, probabilistic classifiers and support vector machines. Our experimental findings show that the impact of different color classifiers on the error rate achieved in decoding can be significant. Furthermore, the use of more complex techniques, such as support vector machines, does not seem to pay off, as they do not achieve better accuracy in classifying color barcode cells. The lowest error rates are indeed obtained by means of clustering algorithms and probabilistic classifiers. From the computational viewpoint, classification with clustering seems to be the method of choice, since it is simple and it does not need time consuming training phases.

2. Related Work

To the best of our knowledge, there is little research in the literature on the color classification for 2D color barcodes. One of the first reported attempt to use color in a 2D barcode can be found in a patent by Han et al. [20], who used reference cells to provide standard colors for correct indexing. Bulan et al. [3] proposed to embed data in two different printer colorant channels via halftone-dot orientation modulation, that is, to print two colors at the same spatial location. This allows to nearly double the capacity of black and white barcodes, which is equivalent to use a 4-ary color scheme for encoding 2 $bit/module$. This work was extended in [4] by using three instead of two colorant layers and a interference mitigating design of the orientations of the three colorants to improve capacity. Microsoft HCCB uses a grid of colored triangles to encode data, using a palette of 4 or 8 colors (4-ary color or 8-ary color scheme). HCC2D, the High Capacity Colored 2-Dimensional Barcode [17] uses a grid of colored squares (using a palette of 4 or 8 colors) and has a symbol structure which builds upon a QR code [11] basis for preserving the QR robustness to distortions. Different color barcode technologies adopt different strategies for classifying colors, that is, for converting analog barcode cells to digital bit streams. For instance, the strategy adopted by the Microsoft HCCB decoder is to make use of a color palette, while Bagherinia and Manduchi [1] proposed an algorithm for decoding barcode elements in a color barcode that does not display its reference colors.

Despite the increasing interest in color barcodes, we are not aware of any previous attempt at performing a comparative analysis of the performance of different methods for

color classification in this framework, which is the main contribution of our work. Experimentation in color classification has been previously addressed in other domains such as color recognition of objects in indoor and outdoor images [5], color recognition of license plates [9] or skin color detection in face localization and tracking [12] [6], where, similarly to our problem, there is the need to discriminate among different classes of color pixels. We emphasize that results from other domains (e.g., pixel classification into skin color and non-skin color) do not necessarily carry through the classification of color cells in barcodes, because the underlying conditions are rather different. Indeed, color classification in 2D barcodes mainly focuses on classifying color cells with minimal size (e.g., thousands cells per square inch) after undergoing a printing and scanning process, while in other domains color elements have other characteristics (e.g., colors in video frames representing natural images). Furthermore, the effective decoding of color barcodes requires much more accuracy and precision than other applications considered for color classification.

3. HCC2D Barcodes

In this section, we introduce 2D color barcodes, which take advantage of colors for achieving higher data density than black and white barcodes. This is obtained at the price of coping with chromatic distortions during decoding. In order to introduce the typical decoding process of a color barcode, we describe next the HCC2D code, which will be used as a paradigmatic example throughout the rest of the paper. We remark that most of the findings reported in this paper about color classification for HCC2D codes apply to color classification for other 2D color barcodes as well.

The HCC2D code is a 2D color barcode which is made of a matrix of square color cells, whose color is selected from a color palette. Figure 1 illustrates samples of HCC2D codes with 4 and 8 colors.

Fig. 1. Samples of the High Capacity Colored 2-Dimensional code (HCC2D): (a) 4 colors and (b) 8 colors. Figure taken from [17]. (Viewed better in color).

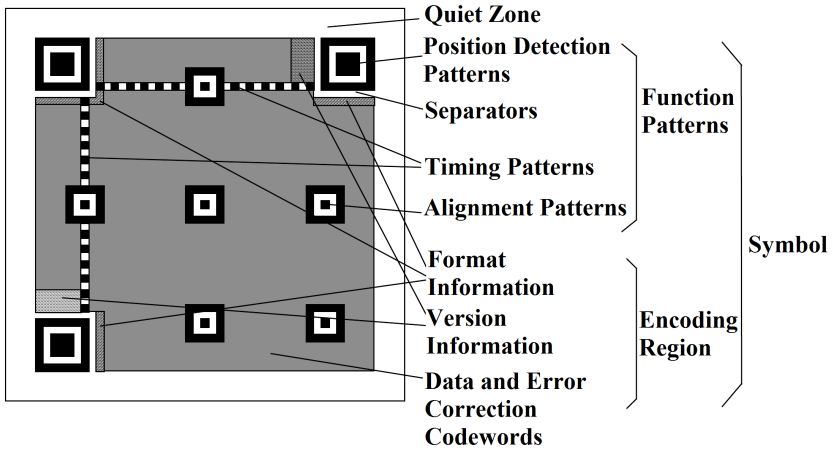

Fig. 2. Structure of generic QR codes and HCC2D codes, which inherit all function patterns of QR codes.

We have designed the HCC2D format with the main goal of increasing the data density while preserving the strong robustness to distortions of Quick Response (QR) codes. QR codes are black and white 2D barcodes designed by the Japanese corporation Denso Wave which are quite widespread among 2D barcodes, because their acquisition process appears to be strongly reliable, and are suitable for mobile environments, where this technology is rapidly gaining popularity.

The main structure of QR codes, and consequently, of HCC2D codes is illustrated in Figure 2, being composed of *Function Patterns* and *Encoding Regions*. The *Position Detection Patterns*, the *Alignment Patterns*, the *Timing Patterns*, and the *Separators for Position Detection Patterns* support the detection process in detecting the presence, the proper orientation and the correct slope of a code into an image. The *Format Information* describes the error correction level used in the code. As previously introduced, the higher the correction level, the higher the redundancy and the reliability of the barcode reading process, but the lower the actual data density rate. The *Version Information* represents the code size, that is, the amount of cells (per side) making up the code. Note that the *Version Information* alone does not determine the final print out size (expressed in $inch^2$ or cm^2), which also depends on hardware parameters, that is, on the printing resolution and on how many printer dots make up each color cell. Finally the *Data and Error Correction Codewords* contains data plus redundancy.

We designed the HCC2D code preserving all the *Function Patterns*, the *Format Information* and the *Version Information* defined in the QR code. Maintaining the structure and the position of such patterns and critical information allows the HCC2D code to preserve the strong robustness to geometric distortions of QR code. Because the retrieval of the *Format Information* and of the *Version Information* is a crucial step during the decoding phase (it may led to reading failures) and its storage requirement is small, there is no significant advantage representing it by color cells. The most important changes are gathered in the *Data and Error Correction Codewords* area. The most noticeable difference with

a QR code is that the modules belonging to the *Data and Error Correction Codewords* area are of different colors; in a HCC2D code with a palette composed of 4 colors each module is able to encode 2 $bit/module$, while 3 $bit/module$ are stored using 8 colors. Introducing colors in the *Data and Error Correction Codewords* area requires to address some issues, which are described in details in [17]. In particular, during QR code reading only the brightness information is taken into account, while HCC2D codes have to cope with chromatic distortions during the decoding phase. Since the *Encoding Region* is made of color cells, the HCC2D decoder needs to know the complete color palette in order to decode the symbol. To consider the color palette as an *a priori* shared knowledge between encoding and decoding processes is not a reliable solution; this is because chromatic distortions would not be properly taken into account, arising differently in each printed and scanned image. The processing should be adaptive to each image for better performance. To make a parallelism with black and white barcodes, QR codes compute an adaptive threshold on each image for discriminating dark and light modules, rather than using static thresholds.

In order to ensure adaptation to chromatic distortions arisen in each scanned code, we have introduced in the HCC2D code an additional field, the *Color Palette Pattern*. This is because color cells of a *Color Palette Pattern* are supposed to be distorted in the same way color cells of the *Encoding Region* are. We make use of replicated color palettes either for cluster initialization or for training machine learning classifiers. Figure 3 illustrates *Color Palette Patterns* in HCC2D codes, located at the boundaries. Note that the *Color Palette Patterns* are not too close to the three *Position Detection Patterns* areas and are far away from each other, thus ensuring that they are robust to local distortion. Furthermore, *Color Palette Patterns* take only 2 rows and 2 columns from a symbol consisting of between 21 and 177 rows and columns, and thus, the overhead is small for high density barcodes.

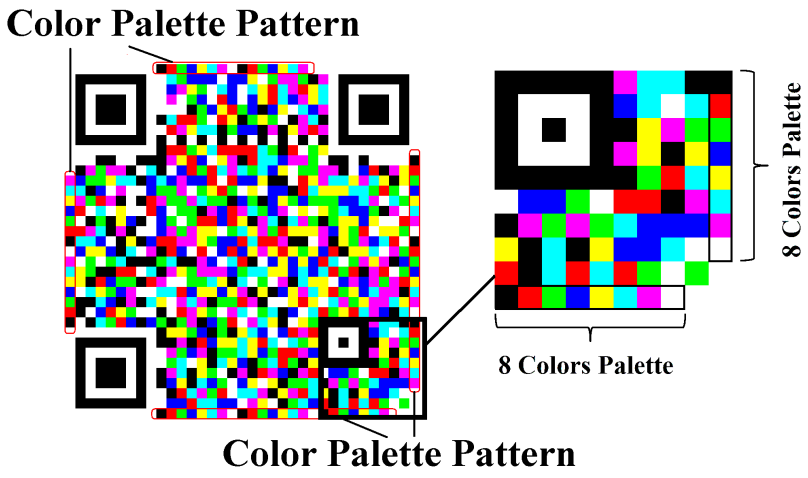

Fig. 3. The four *Color Palette Patterns* are pointed out in a HCC2D code using 8 colors. Figure taken from [17]. (Viewed better in color).

4. Color Classifiers for 2D Color Barcodes

Since the printing and scanning processes introduce chromatic distortions in color barcodes, the decoding success rate depends on the capacity to classify correctly colors of barcode cells. A barcode cell is classified correctly if its original color (before printing) and the class assigned by the classifier to the cell (after scanning) corresponds to each other. A classifier is an algorithm that distinguishes between a fixed set of classes based on labeled training examples. Algorithms reading black and white barcodes may just use a threshold to separate the two classes (that is, dark and light elements), while cells of color barcodes need to be properly classified in many classes, depending on the number of colors. We distinguish 4 or 8 classes (each representing a reference color) into which color pixels may fall, where each pixel is sampled from a cell of the 2D color barcode to decode. Each class reference color is associated with either a 2-bit sequence or a 3-bit sequence. The sequence length depends on how many bits are modulated into each barcode cell (as previously introduced, 4-ary color schemes encode 2 $bit/module$, while 8-ary color schemes encode 3 $bit/module$). Because no classifier is perfect, it is important to know whether a classifier is producing good results on real data sets.

A color classifier may have a training phase and a classifying phase. In the training phase, the classifier is provided with known samples. A known sample consists of a region in the barcode image containing the color to be learned and the corresponding label for that color. For every sample that is added during the training phase, the color classifier computes a color feature and assigns the associated class label to it. A color feature vector (to which a barcode cell is associated with) depends on the color space in which the image is encoded. Usually colors are defined in three dimensional color spaces. For instance, these could either be RGB (Red, Green and Blue) or YUV. The Y in YUV stands for "luma", that is brightness (for instance, black and white TVs decode only the Y channel of the signal). U and V provide color information and are "color difference" signals of blue minus luma (B-Y) and red minus luma (R-Y). Without loss of generality, we assume that each color feature is a represented as a three-dimensional vector in the YUV color space, because the high correlation between RGB channels and the mixing of chrominance and luminance data does not make RGB a very favorable choice for color analysis and color-based recognition algorithms [12]. When all the trained samples (color feature with a label) are added to the classifier, we get a trained color classifier. After the training phase, barcode cells are classified into their corresponding color classes. In the classifying phase, the trained classifier is used on new observations (color features without labels). The classification engine calculates color features of unlabelled samples and classifies them, by associating a label (in our case, a color class) with each unlabelled color element. Once the classification is completed, the original bitstream (which was previously encoded in the 2D barcode) can be retrieved. This is made by concatenating bits from each bit sequence associated with a barcode cell, where the mapping between a barcode cell and a bit sequence is given by the classification output. For instance, without loss of generality, assume that a 2D color barcode is encoding 2 $bit/cell$ by using 4 different reference colors (e.g., black, cyan, magenta and white). Then, assume that each reference color is mapped to a binary sequence (e.g., black is mapped to $\{11\}$, cyan to $\{10\}$, magenta to $\{01\}$ and white to $\{00\}$). Under these assumptions, a dark cell carries the bit sequence $\{11\}$ whether the cell is labelled with the black class by the color classifier. Because a percentage of color cells is always misclassified in real scenarios, bit errors

arise, and thus, the original bit stream and the decoded bit stream may differ slightly from each other.

As previously mentioned, channel coding techniques are capable of correcting errors and restoring the original bit streams, under the assumption that the redundancy introduced in the encoding phase is enough. The main problem considered here is how much redundancy is needed in order to decode successfully 2D color barcodes. This redundancy depends on many parameters, including the algorithms used for color classification. In order to estimate the most suitable redundancy rate (RR), we implemented six different methods for color classification, each of them being representative of a general class of algorithms (minimum distance classifiers, clustering, decision trees, community detection, probabilistic classifiers and support vector machines), and measured their error rates. Next, we briefly describe those algorithms.

4.1. Euclidean Distance (Minimum Distance Classifiers)

Minimum distance classifiers assign unlabelled samples to classes which minimize the distance between unlabelled data and classes in the feature space. The distance is defined as an index of similarity so that the minimum distance is identical to the maximum similarity. We used the Euclidean distance (in RGB, YUV, ...) for identifying similar colors (that is, colors with minimum distance), because it is one of the simplest and most popular distance measures. This method can be taken as a basic reference in our experiments: it is a very simple-minded method, and thus we expect all other methods to produce much lower error rates but to be much slower in their running times.

4.2. K-means (Classification using Clustering)

Clustering is the task of assigning a set of objects into groups (denoted as clusters) so that the objects in the same cluster are more similar to each other than to those in other clusters. Hence, color cells are classified once clustering is completed. Clustering itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. We used the K-means algorithm [14], which is an unsupervised learning algorithm that classifies a given data set through a certain number of clusters (exactly k clusters) fixed a priori. Using the K-means algorithm, we can exploit the *a priori* knowledge about the number of colors in color palettes, so that the algorithm generates exactly 4 or 8 clusters. In our experiments, the starting points (for centroid initialization) were taken by averaging the series of color palettes.

4.3. LMT (Decision Trees)

A decision tree is a classifier in the form of a tree structure, where each node is either a leaf node (which indicates the value of the target class) or a decision node, which specifies some test to be carried out on a single feature value, with one branch and sub-tree for each possible outcome of the test. There are a variety of algorithms for building decision trees; we used the Logistic Model Trees (LMT) [13], because they have been shown to be very accurate and compact classifiers. As in ordinary decision trees, a test on one of the attributes is associated with every internal node. Differently from ordinary decision trees, the leaves have an associated logistic regression function instead of just a class label.

4.4. Louvain (Community Detection)

The Louvain method [2] is a simple, efficient and easy-to-implement algorithm for identifying communities in large networks. It is a greedy optimization method that attempts to optimize the ”modularity” of a partition of the graph, i.e., it tries to maximize the strength of the division of a graph into clusters. Put in other terms, a high modularity in the clustering implies dense connections between nodes within clusters and sparse connections between nodes in different clusters. In order to apply the Louvain method for decoding color barcodes, we define a QR code graph as follows:

- Each QR code module represents a node.
- There is a weighted edge between each pair of nodes, where the weight is inversely proportional to the distance of the nodes in the color space.

Note that this defines a complete graph. To obtain a smaller graph, we consider only edges between nodes that are “close enough” in the color space, according to a given threshold.

4.5. Naive Bayes (Probabilistic Classifiers)

A probabilistic classifier is a function that maps an unlabelled sample to a distribution over class labels. There are a variety of probabilistic classifiers; we used the Naive Bayes algorithm because it only requires a small amount of training data to estimate the parameters. A Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. In simple terms, a Naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. For example, assume that a color cell (in YUV space) is represented by luma (Y value) and by chroma (U and V values). Even if these luma and chroma values depend on each other, a Naive Bayes classifier considers all of these properties to contribute independently to the probability that this color cell is of a given color. The Naive Bayes classifier can be trained very efficiently in a supervised learning setting, using in our case a trained set of known samples taken from the color palette patterns.

4.6. SVM (Support Vector Machines)

Support vector machines (SVM) [7] are supervised learning models, that is, machine learning tasks of inferring a function from labeled training data (in our cases, labelled color cells belonging to color palettes). A support vector machine constructs a hyperplane or set of hyperplanes in a high dimensional space (in our case, a three-dimensional color space such as RGB or YUV), which can be used for classification. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (denoted as functional margin), since in general the larger the margin the lower the error of the classifier. We used the Sequential Minimal Optimization (SMO) algorithm for training support vector machines. This is because SMO efficiently solves the optimization problem which arises during the training, avoiding the use of time-consuming numerical optimizations.

5. Experimentation

We developed a prototype for generating and acquiring HCC2D codes. Even if we restricted our experiments to HCC2D codes only, most of the results (in relative terms) can be generalized to the color classification of any other color barcode. This is due the fact that our experiments focused only on the color classification task.

Barcode reading requires a detection and a decoding phase, where color classification is only one part of the decoding phase. The risk is that errors arising in steps other than color classification may affect the experimental results in an unpredictable way. To prevent this, we proceeded as follows. Even if many factors other than the classification algorithm affect the experiment (e.g., the specific hardware involved, routines for detecting or for sampling color cells which are specific to HCC2D codes), their impact has been kept constant through the use of a common set of barcode scans as input for each classifier, along with the use of common routines for every processing step other than color classification (such as image processing, barcode detection or grid sampling routines). For this reason, even if results of our experiments (in absolute terms) depend on the hardware involved and on the HCC2D code, the relative performance of the algorithms considered seems to be of more general extent.

A metric that we adopt is the byte error rate (ByER), which is defined as the ratio between the number of incorrectly received bytes and the total number of bytes transmitted. ByER depends on characteristics of the channel such as the signal-to-noise ratio (SNR) at the receiver and on the accuracy of the "sensors". In our case, the unreliable channel is a printing and scanning channel, while the "sensors" are represented by the classification algorithms and their accuracy is our parameter of interest. We performed an experiment for computing performance statistics for byte error rates (ByER) and computational time of classifier algorithms. By computing these error rate statistics, we are able to identify the most effective color classifier and the most suitable redundancy rate (RR) for it. This allows us to optimize the data rate (DR), that is, the actual data density of color barcodes.

5.1. Experimental Set-up

We collected 100 barcode scans which make up the sample upon which performance statistics (such as the mean error rate) are computed for each of the 6 methods. Our experimental set-up is as follows:

- We collected 100 barcode scans from real-life applications. Each barcode underwent a real printing and scanning process (i.e., no artificially distorted barcodes).
- Color barcodes were printed and scanned at 600 dpi.
- The print out size of each code was 1 square inch.
- The size of each color cell was 4×4 printer dots.
- Each code was made up of 149×149 cells (out of which we had 512 known color cells to use for training classifiers and 19,720 color cells to classify).
- Each code stored 4,930 bytes in 1 square inch, considering data and redundancy from error correction.
- Each code used 4 colors for encoding 2 $bit/module$.
- Color features were expressed in the YUV space, because the explicit separation of luminance and chrominance components makes this colorspace more attractive for color analysis.

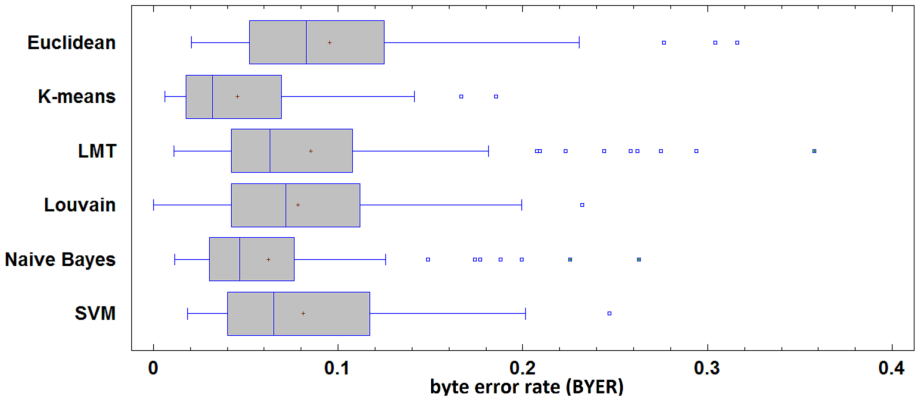

Fig. 4. Box-and-Whisker plot indicating the smallest observation, lower quartile (Q1), median (Q2), upper quartile (Q3), and largest observation. (Viewed better in color).

– Codes stored different input data so that results were not dependent on the specific barcode instance.

Each barcode image was decoded by each one of the 6 classifiers. All our experiments were run on a low-end machine equipped with OS Linux Debian 6.0 running on a 1.73 GHz Intel dual core with 2 GB RAM. Documents were printed and scanned on low cost color laser multifunction printers.

5.2. Experimental Results

We now turn to the experimental results, by starting with the analysis of the byte error rates (ByER). We remark that in the application at hand, the byte error rate (ByER) is more meaningful than the bit error rate (BER). This is due to the fact that 2D barcodes use block error correcting codes, such as Reed Solomon codes (rather than codes protecting against single bit errors), since they have to withstand accidental damages such as ink spots, affecting a contiguous portion of the barcode.

A Box-and-Whisker plot for ByER data is depicted in Figure 4. Box-and-Whisker plots are convenient way of graphically depicting groups of numerical data through their five-number summaries: the smallest observation (sample minimum), lower quartile (Q1), median (Q2), upper quartile (Q3), and largest observation (sample maximum). Quartiles are the three points that divide the data set into four equal groups, each representing a fourth of the population being sampled. The red cross indicates the mean. Outliers, which are observations that are numerically distant from the rest of the data, are depicted too. They are often indicative either of measurement error or that the population has a heavy-tailed distribution.

The shape of each distribution is asymmetric and with several strong outliers. Table 1 shows summary statistics for ByER corresponding to each of the algorithms considered. Table 1 includes measures of central tendency (i.e., the mean), variability (i.e., the standard deviation) and shape (i.e, the standard skewness and the standard kurtosis). The

Table 1. Summary statistics for ByER corresponding to each of the algorithms considered.

	Mean	99% Confidence Interval for Mean	Standard Deviation	Standard Skewness	Standard Kurtosis
Euclidean	0.0956	0.0956 ± 0.0160	0.0610	5.8994	4.8087
K-means	0.0454	0.0454 ± 0.0097	0.0369	5.7278	4.0926
LMT	0.0851	0.0851 ± 0.0177	0.0675	7.4592	6.9833
Louvain	0.0784	0.0784 ± 0.0141	0.0474	2.1610	0.8378
Naive Bayes	0.0621	0.0621 ± 0.0124	0.0473	8.0778	8.9108
SVM	0.0809	0.0809 ± 0.0131	0.0500	3.8256	0.6889

most effective algorithm (with the smallest mean and standard deviation) appears to be the K-means clustering algorithm. The ByER of K-means is 4.54% on average (this sample mean has been computed on a basis of 100 input images). To get a feeling on the sensitivity of the mean to different samples, we have computed the 99.0% confidence intervals for the mean of each ByER distribution, which are reported at the corresponding column in Table 1. The classical interpretation of these intervals is that, in repeated sampling, these intervals will contain the true mean of the population from which the data come 99.0% of the time. In practical terms, we can state with 99.0% confidence that the true K-means ByER is somewhere between 0.0357 and 0.0551. Even if these intervals assume that the population from which the sample comes can be represented by a normal distribution (which is not the case here), the confidence interval for the mean is quite robust and not very sensitive to violations of this assumption. Confidence interval for the standard deviation would be quite sensitive, and thus, they are not computed. Of particular interest here are the standardized skewness and standardized kurtosis (reported at the last columns of Table 1), which can be used to determine whether the sample comes from a normal distribution. Values of these statistics (possibly except for Louvain) are outside the range of -2 to +2, indicating significant departures from normality. Figure 5 illustrates percentiles of ByER distributions (i.e., values below which specific percentages of the data are found).

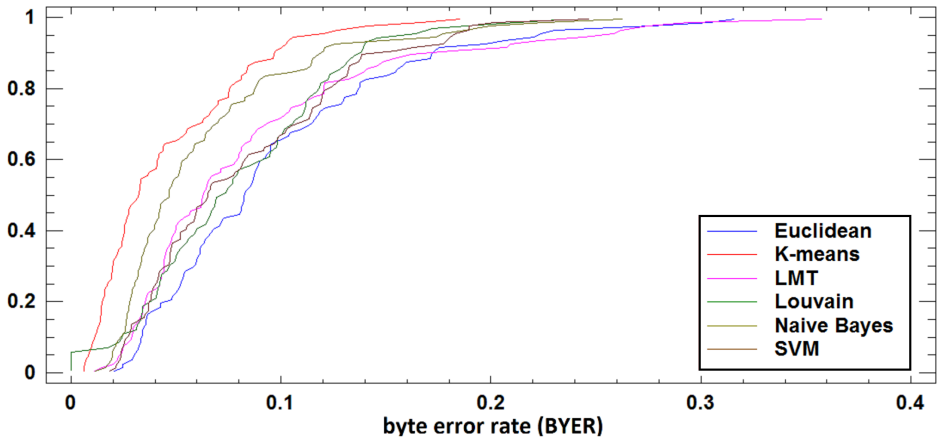

Fig. 5. Percentiles for ByER data of each algorithm. (Viewed better in color).

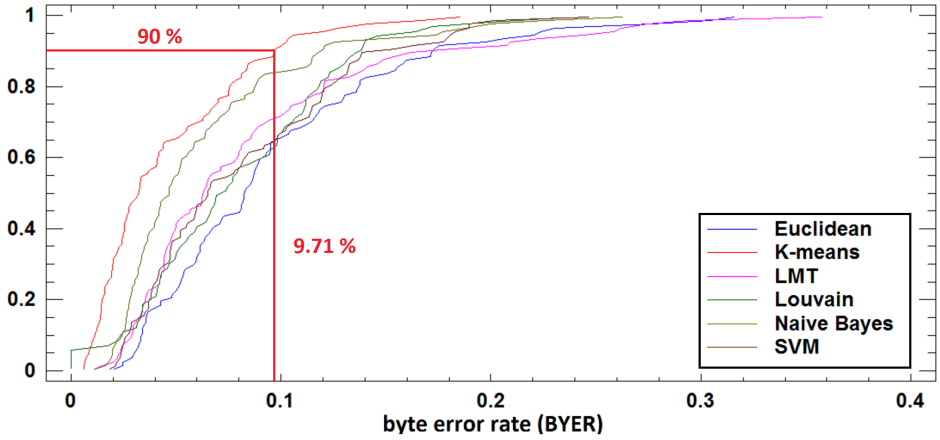

Fig. 6. Percentiles for ByER data of each algorithm. The 90-th percentile for ByER of K-means is highlighted. (Viewed better in color).

We can interpret the percentile plot as follows. Consider the 90-th percentile (the value below which 90% of the cases fall) for ByER of K-means, illustrated in Figure 6, its value being 0.0971. This means that 90 barcodes out of 100 would be decoded if the symbols were robust to ByER up to the 90-th percentile ($\approx 9.71\%$). If the K-means algorithm is used for color classification, we may state that $Prob(ByER < 9.71\%) \approx 90\%$, even if this is just an estimation of the probability on the basis of ByER data collected. If locations of errors are not known in advance, then a Reed Solomon code can correct half as many errors as the redundant symbols. For this reason, in order to achieve a success rate of 90%, the redundancy rate (RR) should be around twice the 90-th percentile ($RR \approx 19.42\%$). Data Rate (DR) is therefore reduced to 80.58% of the overall capacity. Because HCC2D codes are capable of storing 4,930 $bytes/inch^2$ (data plus redundancy), this would results in an effective data density of 3,972 $bytes/inch^2$ with a success rate of 90%. Table 2 shows this trade-off between data density and reliability (in terms of success rate) for each method. Error rates to tolerate for achieving the target success rate are illustrated for three levels (80%, 90% and 95%), along with the corresponding data rate (DR), which is expressed as ratio of data bytes to overall bytes (data plus redundancy bytes). Redundancy rate (RR) is omitted being exactly twice the ByER to tolerate (because of the Reed Solomon code). Finally, barcode data density is expressed in terms of data bytes per square inch. To assess the relative performance of the 6 algorithms considered, we did not just rely on data shown in Table 2, but we also addressed the statistical significance of our experimental results. Statistical hypothesis testing is used to determine whether an experiment conducted provides enough evidence to reject a null hypothesis. We are interested to reject the null hypothesis for which there is no (statistically significant) difference among the ByER distributions related to the 6 classifiers. We can consider ByER distributions two-by-two as paired samples. “Paired”samples means there are two measurements on each sample unit, e.g., measurements on the same subject before and after an intervention. This is the case here, because there are measurements on the same barcode scan before and after the “intervention” (i.e., substitution of the color classifier). We

Table 2. Performance as function of success rate for barcode reading.

	Error Rate to Tolerate $(ByER)$	**Effective Data Rate** (DR)	**Effective Data Density** $bytes/inch^2$
	85% Success Rate		
Euclidean	0.1547	0.6906	3,404.65
K-means	0.0837	0.8326	4,104.71
LMT	0.1399	0.7202	3,550.58
Louvain	0.1276	0.7448	3,671.86
Naive Bayes	0.1081	0.7838	3,864.13
SVM	0.1320	0.7360	3,628.48
	90% Success Rate		
Euclidean	0.1723	0.6554	3,231.12
K-means	0.0971	0.8058	3,972.59
LMT	0.1717	0.6566	3,237.03
Louvain	0.1385	0.7230	3,564.39
Naive Bayes	0.1190	0.7620	3,756.66
SVM	0.1463	0.7074	3,487.48
	95% Success Rate		
Euclidean	0.2223	0.5554	2,738.12
K-means	0.1126	0.7748	3,819.76
LMT	0.2511	0.4978	2,454.15
Louvain	0.1641	0.6718	3,311.97
Naive Bayes	0.1755	0.6490	3,199.57
SVM	0.1818	0.6364	3,137.45

have run paired tests on these samples such as the sign test and the Wilcoxon signed-rank test, for testing the null hypothesis that there is "no difference in medians" between the distributions.

The result of each test is denoted as P-value, which is the probability of obtaining a test statistic at least as extreme as the one that was actually observed, assuming that the null hypothesis is true. If the P-Value is under 0.01, the medians of the samples are significantly different at the 99.0% confidence level. For instance, running the Wilcoxon test on the Naive Bayes ByER distribution and on the K-means ByER distribution results in a P-Value of $1.03 \cdot (10^{-9}) \ll 0.01$. Running paired tests on ByER distributions taken two-by-two as paired samples, we have rejected all null hypotheses but one; we can not say anything about the performance difference between Logistic Model Trees and Support Vector Machines. Values computed on 100 barcode scans suggest that SVM has smaller average ByER than LMT, but the difference was found not to be significant (P-value resulting from the Wilcoxon signed-rank test is $0.7413 \gg 0.01$ and from the sign test is $0.6170 \gg 0.01$). We could not use more powerful statistical tests (parametric tests such

as the t-test) because the normality assumption would be violated. In summary, the results of our experiments showed that, at 99.0% confidence level, K-means outperforms Naive Bayes, which in turn outperforms Louvain, which in turn outperforms Support Vector Machines and Logistic Model Trees, which in turn outperform Euclidean classifiers.

Finally, we address the computational overhead introduced by the color classifiers considered. We stress that the overall running time is important, since in many applications a color barcode must be decoded on low-end devices, such as mobile phones or tablets. Figure 7 illustrates the Box-and-Whisker plot for computational time distributions (the sample size is 100 elements for each method). As expected, the Euclidean classifier is the simplest and fastest algorithm among the studied methods; it is capable of classifying 19,720 color cells in a few milliseconds. In our experiments, the K-means algorithm was also fast (order of milliseconds), while all other classifiers had a much higher computational overhead, as they required up to several seconds for the classification phase. The overall running time for the Louvain method is not reported in Figure 7, since its average computational time is around a minute. This is due, in part, to the overhead required to generate the QR code graphs.

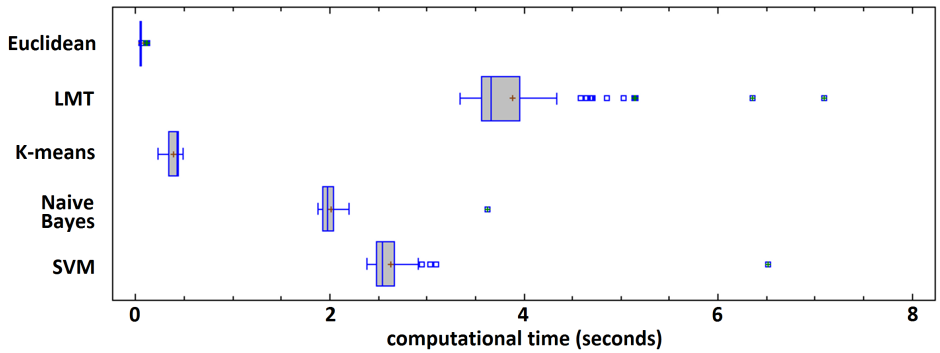

Fig. 7. Box-and-Whisker plot for computational time. (Viewed better in color).

5.3. Experiments with Mobile Phones

In this section we report the results of our experiments on mobile phones. The experimental setup is exactly the same as in our previous experiment (on desktop scanners) except for the fact that now we use mobile phones (mainly on Google Nexus 4 and Google Nexus 5) for acquiring color barcodes and that we increase the print out size to 1.5 inch per side (with a cell size of 6×6 printer dots), in order to allow the camera focus to work properly.

Table 3 reports, with exactly the same format used in Table 1, the results of our experiments with mobile phones. Beside the absolute values of error rates, which are dependent on the specific devices used for printing and scanning, it can be seen that, in any case, the choice of the classifier has a non-negligible impact on the error rate distribution. Figure 8 illustrates, similarly to Figure 4, the Box-and-Whisker plot for byte error rate in mobile environment. The first striking difference with the experiments on scanners is that

Table 3. Summary statistics for ByER data related to barcode reading by mobile phones.

	Mean	99% Confidence Interval for Mean	Standard Deviation	Standard Skewness	Standard Kurtosis
Euclidean	0.0664	0.0664 ± 0.0208	0.0792	9.4180	14.260
K-means	0.2910	0.2910 ± 0.1068	0.4065	3.4400	-2.6370
LMT	0.0456	0.0456 ± 0.0154	0.0586	14.400	35.040
Louvain	0.1147	0.1147 ± 0.0232	0.0885	7.5800	8.5200
Naive Bayes	0.0841	0.0841 ± 0.0195	0.0743	6.9220	8.5980
SVM	0.0596	0.0596 ± 0.0142	0.0543	7.5010	10.960

K-means is no longer the most effective algorithm, and it appears to degrade its performance in case of strongly non-uniform illumination of the barcodes, a problem which occurs frequently with mobile phone cameras. This is somewhat different from our preliminary experiments reported in [19], where the data set considered contained pictures taken under uniform light conditions.

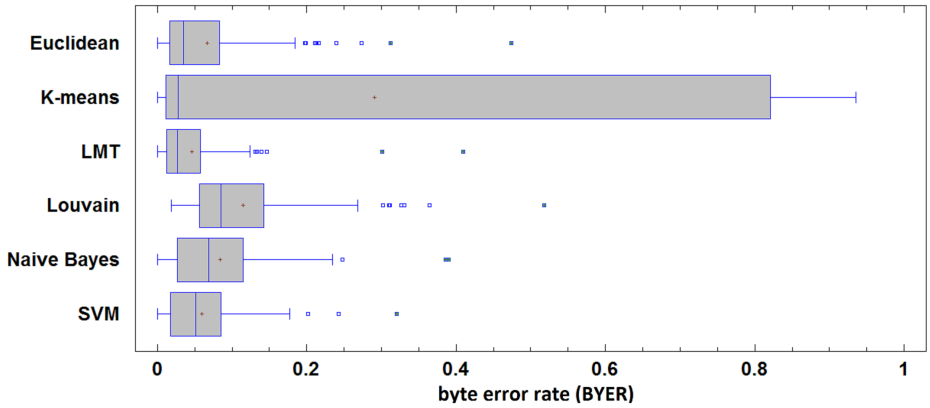

Fig. 8. Box-and-Whisker plot for ByER data related to barcode reading by mobile phones. (Viewed better in color).

To illustrate this phenomenon more in detail, we sorted the barcode images in our new data set by error rates (ByER values), and defined good barcode scans and bad barcode scans respectively as the top and the bottom 25% images in this ranking. We observed that in our sample, good and bad barcode scans differ mainly in the underlying illumination condition. Our experiments, illustrated in Figures 9 and 10, shows a sharp degradation of performance of K-means from good to bad barcode scans, while other methods appear to be less sensitive to bad inputs. The main difference with our experiments in the desktop scenario is due to the fact that desktop scanners have controlled light intensity, while pictures taken from phone cameras present a much larger variation in light conditions.

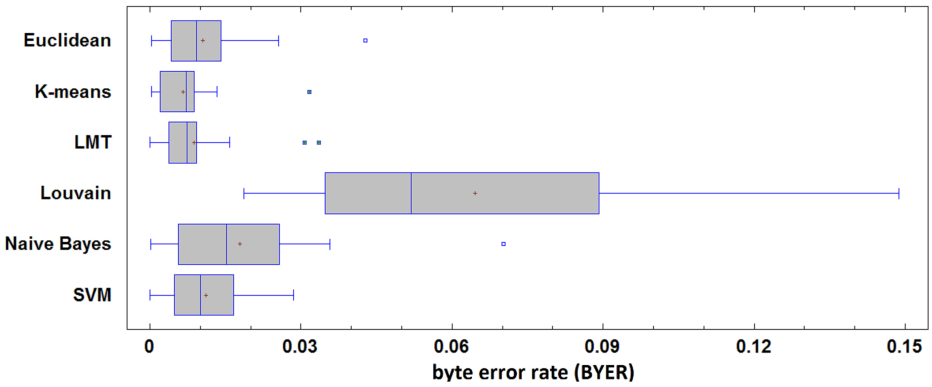

Fig. 9. Box-and-Whisker plot for ByER data (top 25 values out of 100) related to barcode reading by mobile phones. (Viewed better in color).

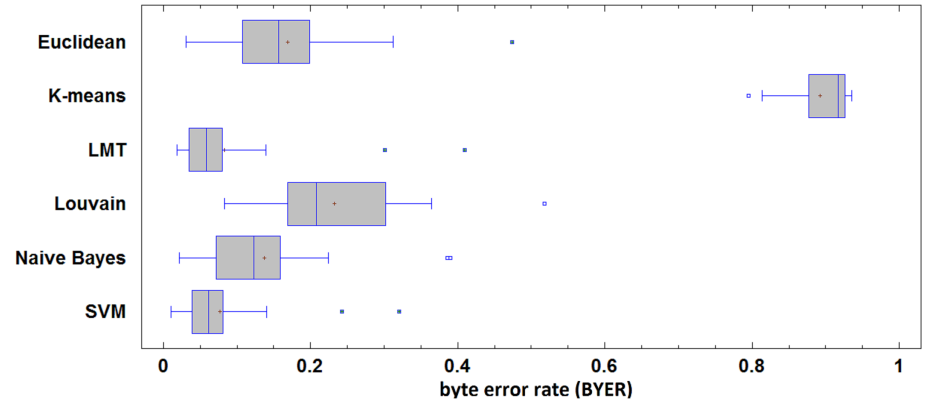

Fig. 10. Box-and-Whisker plot for ByER data (bottom 25 values out of 100) related to barcode reading by mobile phones. (Viewed better in color).

The quality difference between good and bad barcode scans in mobile scenarios is much more widened than in case of desktop scenarios (see Figures 11 and 12), and in many bad cases K-means fails to converge to an optimal solution.

Analogously to Figure 5, Figure 13 illustrates percentiles of ByER distributions. One can easily see that, differently from the other methods, K-means is able to decode successfully 68% of images in our data set, while its performance degrades substantially for the remaining 32% of images.

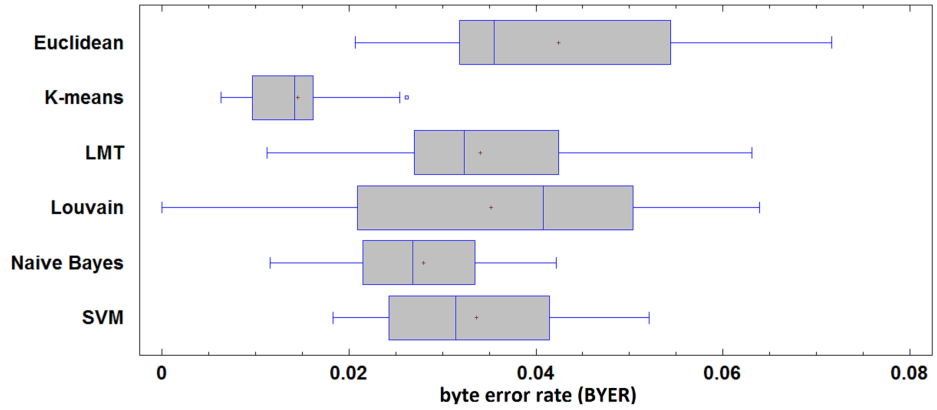

Fig. 11. Box-and-Whisker plot for ByER data (top 25 values out of 100) related to barcode reading by desktop scanners. (Viewed better in color).

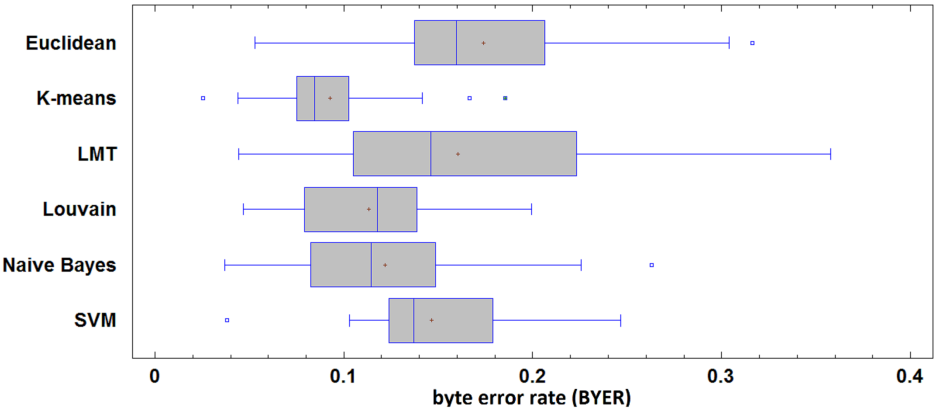

Fig. 12. Box-and-Whisker plot for ByER data (bottom 25 values out of 100) related to barcode reading by desktop scanners. (Viewed better in color).

As far as the computational performance of our algorithms is concerned, the simpler methods (Euclidean and K-means) were still able to run in order of seconds on a mobile phone architecture, while the remaining methods were too slow or too difficult to implement efficiently on mobile architectures. This raises an interesting challenge, since there does not seem to be an efficient and effective method for mobile phones. Indeed, efficient methods (such as Euclidean and K-means) appear not to be effective (in terms of error rate) while the more effective methods (such as LMT, Louvain, Naive Bayes and SVM) do not seem to be practical enough on mobile architectures.

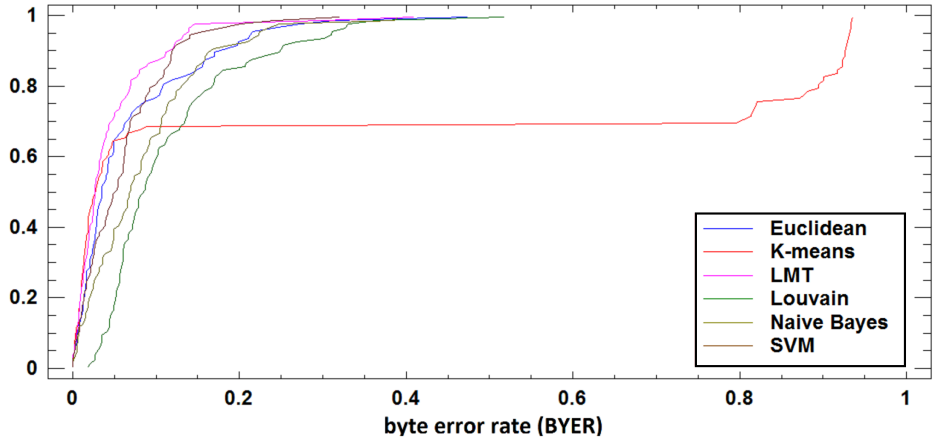

Fig. 13. Percentiles for ByER data related to barcode reading by mobile phones. (Viewed better in color).

6. Conclusions and Future Work

In this paper, we have addressed the trade-off between reliability and data density in 2D color barcodes, and reported the results of a thorough experimental study in desktop and mobile scenarios. Our experiments have shown that the impact of a color classifier on the error rate can be significant and that more complex classifiers do not necessarily achieve better accuracy in classifying color barcode cells. With the help of our study, one can identify the most suitable ways to convert analog color cells to digital bit streams.

We have shown that state-of-art methods for color classification could be successfully used for decoding color barcodes in desktop scenarios, while color classification for pictures taken from phone cameras appears to be still a challenge. Indeed, in the mobile scenario, simple and efficient methods (in terms of computational time) such as the Euclidean and the K-means classifiers are not effective (in terms of error rate), while, more complex methods are effective but not efficient.

For future work, we see a number of interesting directions where our study maybe extended, in particular for mobile devices, since there is an emerging need for new efficient classification methods, capable of addressing the problem of strong non-uniform illumination.

References

1. Bagherinia, H., Manduchi, R.: A theory of color barcodes. In: ICCV Workshops (2011)
2. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment 2008(10) (2008)
3. Bulan, O., Monga, V., Sharma, G.: High capacity color barcodes using dot orientation and color separability. In: Proceedings of Media Forensics and Security. vol. 7254. SPIE (January 2009)
4. Bulan, O., Sharma, G.: High capacity color barcodes: Per channel data encoding via orientation modulation in elliptical dot arrays. Image Processing, IEEE Transactions on 20(5), 1337–1350 (2011)

5. Buluswar, S., Draper, B.: Color recognition in outdoor images. In: Sixth International Conference on Computer Vision. pp. 171–177. IEEE (1998)
6. Chai, D., Bouzerdoum, A.: A Bayesian approach to skin color classification in YCbCr color space. In: TENCON 2000. vol. 2, pp. 421–424 (2000)
7. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning 20(3), 273–297 (1995)
8. Grillo, A., Lentini, A., Querini, M., Italiano, G.F.: High capacity colored two dimensional codes. In: Computer Science and Information Technology (IMCSIT), Proceedings of the 2010 International Multiconference on. pp. 709–716. IEEE (2010)
9. Guo, D., Chen, L., Lu, Z., Han, L.: Vehicle plate location techniques based on plate grounding-color recognition. Computer Engineering and Design 5, 023 (2003)
10. Hecht, D.: Printed embedded data graphical user interfaces. Computer 34(3)
11. ISO 18004:2006: QR Code 2005 bar code symbology specification.
12. Kakumanu, P., Makrogiannis, S., Bourbakis, N.: A survey of skin-color modeling and detection methods. Pattern Recogn. 40(3), 1106–1122 (Mar 2007)
13. Landwehr, N., Hall, M., Frank, E.: Logistic model trees. Machine Learning 59(1-2), 161–205 (2005)
14. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. vol. 1, p. 14. California, USA (1967)
15. Microsoft Research: High Capacity Color Barcodes. http://research.microsoft.com/projects/hccb/ (2013), [Online; accessed 10-December-2013]
16. Parikh, D., Jancke, G.: Localization and segmentation of a 2D high capacity color barcode. In: Proceedings of the 2008 IEEE Workshop on Applications of Computer Vision. IEEE Computer Society (2008)
17. Querini, M., Grillo, A., Lentini, A., Italiano, G.: 2D color barcodes for mobile phones. International Journal of Computer Science and Applications (IJCSA) 8(1), 136–155 (2011)
18. Querini, M., Italiano, G.: Facial recognition with 2D color barcodes. International Journal of Computer Science and Applications (IJCSA) 10(1), 78–97 (2013)
19. Querini, M., Italiano, G.F.: Color classifiers for 2D color barcodes. In: Federated Conference on Computer Science and Information Systems (FedCSIS). pp. 611–618. IEEE (2013)
20. Tack-Don, H., Cheol-Ho, C., Nam-Kyu, L., Eun-Dong, S., et al.: Machine readable code image and method of encoding and decoding the same (2006)

Marco Querini is a research associate at the department of Civil Engineering and Computer Science Engineering, University of Rome "Tor Vergata". He received the Master's degree in Computer Engineering from the University of Rome "Tor Vergata" in 2010. From the same university, he received the PhD in Computer Science and Automation Engineering in 2014, defending a thesis which presented a new system for securizing documents through their entire life cycle. His research focuses on computer security, high capacity barcodes in security applications, document securization and verification, handwritten signature verification. He is author of a number of publications in those fields, which also address challenges given by todays exploding world of mobile devices. Eng. PhD. Marco Querini is also serving as program committee member of IEEE conferences such as MMAP 2014.

Giuseppe F. Italiano is a Professor of Computer Science at University of Rome Tor Vergata in Italy, where he was also the Department Chair from 2004 to 2012, Chair of the Evaluation Committee from 2002 to 2008, and is currently Vice-Rector for Quality,

Evaluation and Performance. Prof. Italianos research focuses on the design, analysis, implementation and experimental evaluation of algorithms and data structures. In particular, he is interested in several research areas, including algorithm engineering, combinatorial algorithms, computer security, graph algorithms and string algorithms. In those areas, he has published over 200 papers in journals and conference proceedings, is the inventor of few US patents, and has co-founded two technology startups. Prof. Italiano has been serving as Editor-in-Chief and Associate Editor in several journals in theoretical computer science. His research has been funded in part by several EU projects, industrial contracts and by the Italian Ministry of Education, University and Research.

Received: December 18, 2013; Accepted: May 26, 2014.