Graphic Code:
Creation, Detection and Recognition

Leandro Cruz[1]
lmvcruz@isr.uc.pt
Bruno Patrão[1]
bpatrao@isr.uc.pt
Nuno Gonçalves[12]
nunogon@deec.uc.pt

[1] Institute of Systems and Robotics,
University of Coimbra
Coimbra, Portugal

[2] Portuguese Mint and
Official Printing Office
Lisboa, Portugal

Abstract

Graphic Code[1] is a new Machine Readable Coding (MRC) method. It creates coded images by organising available primitive graphic units arranged according to some predefined patterns. Some of these patterns are previously associated with symbols used to compose the messages and to define a dictionary. According to the same coding principle presented in this work, we can develop three kinds of graphic codes, each of them able to create a very different coding styles (black and white pixel-based, coloured pixel-based, and icon-based ones). It significantly improves code aesthetic. Besides that, this coding method is able to encode more information than classical approaches, which open further possibilities of applications.

Furthermore, we will present the pipeline for decoding a graphic code from a photo. It is performed by assessing some images so that coded message is recovered, and it might be supported by using of data redundancy and check digits to validate it, what provides a superior robustness to the whole process.

Figure 1: Combination of a colour pixel-based code (girl) with an icon-based code (musical notes).

1 Introduction

Graphic Code is a Machine-Readable Code (MRC) pattern that has presented significant advances in terms of code aesthetics and the large amount of information that can be encoded in it. This pattern was initially presented by Patrão et al. [2] as a smart marker for Augmented Reality applications. Then, the coding process was deeply described by Cruz et al. [1].

This pattern consists of an appropriate primitive graphic units distribution throughout a certain image. We will assume that these primitive are black or white pixels (although they may be pixels of other colours, or even drawings with some complexity, as presented in Figure 1). Such primitives, the graphic units, are distributed throughout the code into clusters called cells. Some cell patterns are previously associated with characters, forming what is called a dictionary (the primary element of encoding and decoding).

The encoding process returns an image. For practical purposes, this image is printed and must be decoded from a photograph. From this photo, we need to reconstruct the initially generated code to return this code to the decoding process. In this work, we will focus on this reconstruction process. In general, this step consists of properly identifying each primitive graphic unit and its cell patterns.

2 Coding

The basis of our coding and decoding process is a mapping between symbols and patterns that we call dictionary. It defines an alphabet which can be used to encode and decode the message. This alphabet can be binary (0 and 1), numeric (from 0 to 9), alphanumeric (characters from A to Z, numbers from 0 to 9), etc. Our work is a dithering-based approach [4]. This technique converts each pixel of a grayscale image into $k \times k$ black or white pixels. This process reduces the colour space of the image (from 256 levels of grey to black and white) in order to increase image resolution (multiplies each dimension by k) and to preserve perception of grayscale (through colour integration of a cell made by human eyes). According to the dithering technique, each grayscale interval is associated with a specific pattern. This approach can be easily extended to coloured images just by shifting colours to a specific colour-base distribution (when pixels are seen together, they produce original colour perception). We define specific patterns and associate them with symbols.

Coding process receives as input (i) a dictionary, (ii) a base image (which determines the size and overall appearance of the code) and (iii) a message that will be encoded. It creates an image in which each pixel refers to a graphic unit. We will assume that the graphic units (black or white pixels) are grouped into 3×3 cells, this pipeline is illustrated in Figure 3. In addition, we also assume that we will add a checker digit to the end of the message consisting of 3 characters that will be used to validate the decoding process.

After generating the encoded image, we add a specific border, as shown in Figure 2. This border is essential to the reconstruction process, which will be discussed below.

Decoding process, in turn, consists of scanning the code searching in order to reach cell patterns that form the employed dictionary. When such a pattern is found, its corresponding character is added to the message being retrieved.

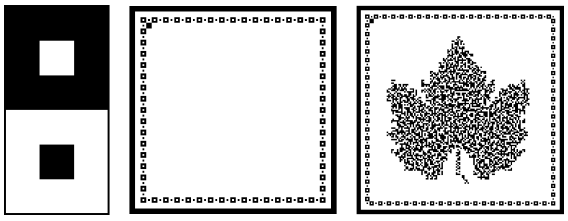

Figure 2: Frame used to support graphic code reconstruction and decoding from a photo.

3 Graphic Code Reconstruction

Reconstruction process is described in Figure 4. It begins with the printed code photo acquisition. Then, we detect the code in this image, and in that region, we identify some points characteristics of the frame. Since our frame is rectangular, we detect in this region the inner and outer quadrilaterals of the external black rectangle of the frame. These quadrilaterals are illustrated in Figure 4 by a red and green polygon. It is noteworthy that, within the region detected as code there are several quadrilaterals,

[1] Patent Application: INPI 20171000063377

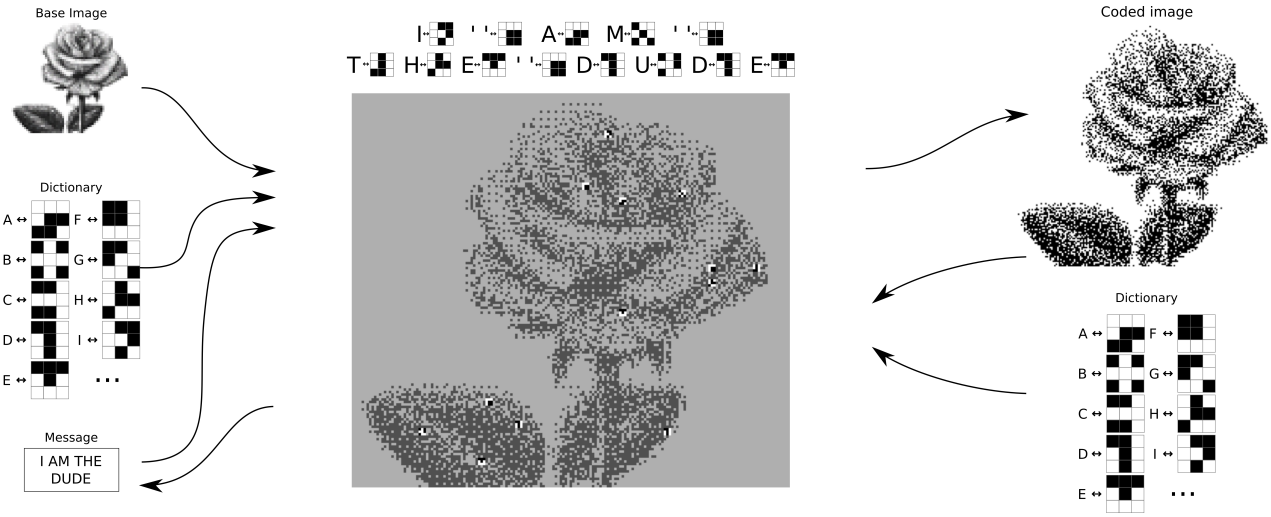

Figure 3: The encoding pipeline starts with a base image, a dictionary and a message. Next, it defines the candidate pixels according to the quanta used in the dictionary, then it places the respective pattern of each message symbol and the encoding finishes by filling the remaining cells while decoding is the opposite process.

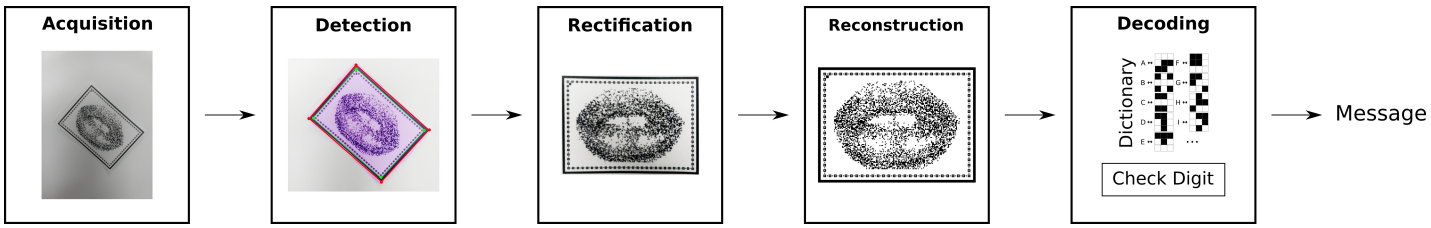

Figure 4: Decoding pipeline process from photos. Starting with photo acquisition, code detection, image rectification, code reconstruction and message decoding.

therefore, this element choice is made by taking the two concentric contours that have the largest area and their respective points.

After retrieving these points, we can rectify the image. The rectification purpose is to obtain an image in which the code appears without distortion and with a known scale. It is achieved by defining an homography matrix that establishes the relationship that transforms the two quadrilaterals obtained in the detection process in rectangles with the expected dimensions. We also use the 3×3 black square at top-left corner of the frame to define the code orientation.

With the rectified image, we proceed with the reconstruction process, that is, identification of each graphic unit. Recognition process of the graphic unit is done as follows: (i) identification of the position in which the samples will be collected and (ii) graphic unit colour assessment.

The rectified code image is 9 times larger in each dimension than the original one. In this way, we have an area of 9×9 pixels referring to the same graphic unit. In other words, the reconstruction process consists of choosing some of these pixels and defining graphic unit colour.

It is important to highlight that although the graphic units are regularly arranged in the initial code (pixels of an image), the printing process, paper curl, photograph artefacts, or inaccuracies in the rectification process create deformations in this rectified image. For this reason, they cause irregular sampling spacing between graphic units and produce such an undesirable target result.

As a result, we perform a correction in the position of these samples from an analysis of the image deformation in the frame edge pattern region. Since the edge pattern is known, we can identify each pixel deformation. Then, it is interpolated to the interior of the model, hence we achieve a better sampling of the graphic units. Such approach has provided impressive results when the printed code lightly curled.

When the position of the samples is decided, we collect 9 of them and apply a voting process to select graphic unit colour. Since the graphic units are black or white, and these samples from the photo are usually grayscale, we need to apply a threshold to decide their colour. This threshold is determined with an analysis of the edge patterns of alternating squares (where there is a properly balanced black and white pixel distribution) that belong to the frame.

4 Concluding remarks

Graphic Code has played such an important role regarding tag patterns and smart markers used in Augmented Reality applications. In this work, we aimed to present how we can decode it from a printed version photograph.

Coding process has a linear complexity ratio. It is particularly accomplished between 20ms and 30ms for models with 60×40 cells of 3×3 graphic units. Similarly, reconstruction process complexity is linear, but it may not have a good reconstruction efficiency at first due to previously identified limitations. However, it has been empirically noticed that when the printed code is lightly curled and the photograph is properly in focus, it does not take more than 10 trials, hence the process is concluded in less than 3 seconds.

Afterwards, the resulting code is translated by the decoding process. Then we verify whether the check digit of the first characters matches the last three ones. When they do not match, it means the reconstruction process failed at some point, and it starts processing the next frame.

Future work points out towards insert redundancy into the graphical units [3] and use it to become reconstruction and decoding processes even more robust. Another future work is to present the reconstruction to other kinds of graphic units (like more general drawings).

References

[1] Leandro Cruz, Bruno Patrão, and Nuno Gonçalves. Halftone Pattern: A New Steganographic Approach. *Eurographics - Short Papers*, 2018.

[2] Bruno Patrão, Leandro Cruz, and Nuno Gonçalves. An application of a halftone pattern coding in augmented reality. *SIGGRAPH Asia Posters*, 2017.

[3] Inving Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 1960.

[4] Robert Ulichney. The void-and-cluster method for dither array generation. *Human Vision, Visual Processing, and Digital Display*, 1993.