Recognizing the Presence of Hidden Visual Markers in Digital Images

Liming Xu
University of Nottingham Ningbo China
International Doctoral Innovation Centre
Ningbo 315000, Zhejiang, China
liming.xu@nottingham.edu.cn

Andrew P. French
University of Nottingham, Nottingham, UK
School of Computer Science, Wollaton Road
Nottingham, UK NG7 2RD
andrew.p.french@nottingham.ac.uk

Dave Towey
University of Nottingham Ningbo China
AIOP Group, School of Computer Science
Ningbo 315000, Zhejiang, China
dave.towey@nottingham.edu.cn

Steve Benford
University of Nottingham, Nottingham, UK
School of Computer Science, Wollaton Road
Nottingham, UK NG7 2RD
steve.benford@nottingham.ac.uk

ABSTRACT

As the promise of Virtual and Augmented Reality (VR and AR) becomes more realistic, an interesting aspect of our enhanced living environment includes the availability — indeed the potential ubiquity — of scannable markers. Such markers could represent an initial step into the AR and VR worlds. In this paper, we address the important question of how to recognise the presence of visual markers in freeform digital photos. We use a particularly challenging marker format that is only minimally constrained in structure, called Artcodes. Artcodes are a type of topological marker system enabling people, by following very simple drawing rules, to design markers that are both aesthetically beautiful and machine readable. Artcodes can be used to decorate the surface of any objects, and yet can also contain a hidden digital meaning. Like some other more commonly used markers (such as Barcodes, QR codes), it is possible to use codes to link physical objects to digital data, augmenting everyday objects. Obviously, in order to trigger the behaviour of scanning and further decoding of such codes, it is first necessary for devices to be aware of the presence of Artcodes in the image.

Although considerable literature exists related to the detection of rigidly formatted structures and geometrical feature descriptors such as Harris, SIFT, and SURF, these approaches are not sufficient for describing freeform topological structures, such as Artcode images. In this paper, we propose a new topological feature descriptor that can be used in the detection of freeform topological markers, including Artcodes. This feature descriptor is called a Shape of Orientation Histogram (SOH). We construct this SOH feature vector by quantifying the level of symmetry and smoothness of the orientation histogram, and then use a Random Forest machine learning approach to classify images that contain Artcodes using the new feature vector. This system represents a potential first step for an eventual mobile device application that would detect where in an image such an unconstrained code appears. We also explain how the system handles imbalanced datasets — important for rare, handcrafted codes such as Artcodes — and how it is evaluated. Our experimental evaluation shows good performance of the proposed classification model in the detection of Artcodes: obtaining an overall accuracy of approx. 0.83, F_2 measure 0.83, MCC 0.68, AUC-ROC 0.93, and AUC-PR 0.91.

CCS CONCEPTS

• **Human-centered computing → Mixed / augmented reality**; **Human computer interaction (HCI)**; *Mobile computing*; • **Computing methodologies → Supervised learning by classification**; *Bagging*;

KEYWORDS

Visual markers; Artcodes; Topological feature descriptor; Classifier

1 INTRODUCTION

With the proliferation of augmented reality techniques, we can expect to encounter various digital markers in everyday life: scanning QR codes for mobile pay, watching objects come alive via services such as Blippar, or triggering digital footprints by scanning Artcodes [25]. We are living with visual markers; some are visible to people and some others are "hidden", appearing as commonly encountered images, drawings or shapes. In this paper, we investigate the challenge of recognizing the presence of these hidden visual markers in images. In particular, we propose a system that can be used to alert mobile devices to be aware of the presence of Artcodes in images.

Artcodes (Figure 1) are visual markers originated from D-touch [11] in which computer-readable visual codes are embedded into images, allowing a designer to create machine-readable and human-meaningful codes that adopt the space between the visibility of the QR codes and the secrecy of these "invisible" markers. As an augmented reality artefact, Artcodes are human-designed topological visual markers that are both attractive and meaningful to humans and machine readable [25], and have been applied in various scenarios, including enhancing a dining experience [25], augmenting

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ThematicWorkshops'17, October 23–27, 2017, Mountain View, CA, USA
© 2017 ACM. ISBN 978-1-4503-5416-5/17/10…$15.00
DOI: https://doi.org/10.1145/3126686.3126761

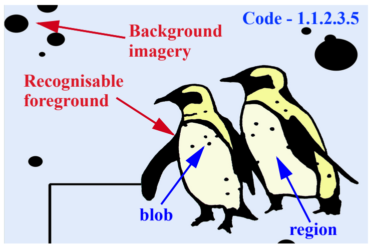


Figure 1: Artcode and its basics

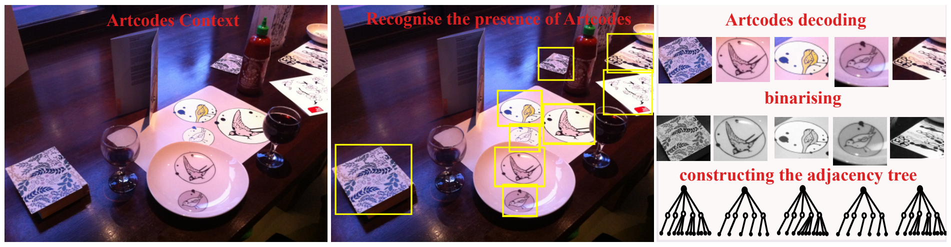

Artcodes Context
Recognise the presence of Artcodes
Artcodes decoding
binarising
constructing the adjacency tree

Figure 2: Localizing the presence of Artcodes and the process of Artcodes decoding

an accoustic guitar with digital footprints [4, 5], attaching complex narratives to public illustrations [35], and designing a mobile garden guide by Artcode-based signages [28].

As codes such as Artcodes enter into everyday life and appear on the surface of objects, automatically recognizing their presence becomes a key first step before decoding and triggering its digital effects: before we can act on a code, we must first know it exists. Unlike the restrictive geometry of other commonly used markers such as QR codes [34] or ARTags [15], Artcodes are built topologically and have no fixed geometric shape. Therefore, it is difficult to detect them using traditional differentiation geometry-based feature descriptors. The characteristic that makes Artcodes different from other images is this topological structure, a number of connected regions containing several *blobs* (as presented in Figure 1) [11, 25].

This paper addresses the question of recognizing the presence of Artcodes in images – namely, the problem of Artcode *detection* rather than decoding. In other words, we wish to classify an image as containing an "*Artcode*" or not, a binary classification problem. There are three steps in general to address such a classification problem: *first*, build the training data; *second*, find a feature descriptor with high invariance under a variety of photometric and geometric changes; and *third*, train a classifier (e. g. Support vector machine [10], or Random forests [6]) using these features on training data, and make predictions on forthcoming unseen input data. Considering the necessarily real-time response of detection and recognition of Artcodes on a mobile platform, random forests are used as the classifier.

Here, we propose a new feature descriptor more suited to describe Artcode images — the Shape of Orientation Histograms (SOH). Unlike traditional geometry-based feature descriptors (e. g. SIFT, or SURF), SOH is a topological descriptor designed for discriminating different topologies. Topological distinctiveness can be calculated from the smoothness and symmetry of orientation histograms [16]. Two factors — *symmetry* and *smoothness* — are proposed to describe the shape of histogram, representing the symmetrical similarity between the left and right sections of the histogram, and the smoothness level of this histogram, respectively. Distance metrics are then employed to quantify the symmetry and smoothness of the orientation histogram. As we shall see, this allows us to pick out unique features of the closed-contour Artcodes.

The rest of this paper is organized as follows: Section 2 describes some related work. Section 3 introduces Artcodes and Artcodes detection. Section 4 describes the SOH feature descriptor and the process of constructing the SOH vector. Experimental evaluation is given in Section 5, and Section 6 concludes this paper.

2 RELATED WORK

Over the past decades, there has been extensive research into feature localization and/or description, mainly including two classes: geometrical features (Harris [19], SIFT [22], and SURF [1]) and binary features (FAST [31], BRIEF [7], and ORB [32]). Geometric features commonly first detect the keypoints at the most distinctive locations (local extrema after using Hessian or Harris operators, for instance) in the Laplacian of Gaussian (LoG) or approximate LoG scale space of the images, such as *corners*, *blobs* and *T-junctions*; and then construct feature descriptors based on local patches around the keypoints, such as histogram of oriented gradients (SIFT), Haar wavelet responses (SURF), or convolved orientation maps (DAISY[36]).

Binary feature detectors describe keypoints by examining the intensities of local patches. Rosten et al. [31] proposed FAST detector, a decision tree based detector which is able to effectively classify the image patches based on a relatively small number of pair-wise intensity comparisons. Colonder et al. [7] proposed BRIEF, directly comparing the intensities of pairs of pixels in an image patch after applying Gaussian smoothing to build the descriptor vector without requiring a training phase. Rublee et al. [32] proposed an Oriented FAST and Rotated BRIEF (ORB) descriptor, which is invariant to rotation and robust to noise, and much faster than even FAST and BRIEF features.

However, both geometric and binary features are based on local image patches, they detect and/or describe keypoonts depending on these keypoints' neighborhoods. They are not designed to model topology. We propose in this paper a new topological feature descriptor called Shape of Orientation Histogram (SOH) to model the image's topology. SOH is built from the existing orientation histogram [16]. However, rather than directly using the orientation histogram, SOH is a quantitative feature vector constructed from the quantification of the symmetry and smoothness of the orientation histogram.

Orientation histograms were first used by Freeman et al. [16] in 1995 for hand gesture recognition, where an orientation histograms were computed using steerable filters and the orientations with magnitudes below a threshold were suppressed. This orientation histograms have a certain level of robustness to lighting changes and transformational invariance. Orientation histograms are fast to calculate and therefore are suited to real-time applications.

Using histograms to build a feature vector has been well studied. Lowe used histograms of oriented gradients (HoGs) in the neighborhood of keypoints to build the feature description vector [22]. Makolajczyk et al. [26] used orientation-location histograms combined with thresholded gradient magnitude as feature descriptors

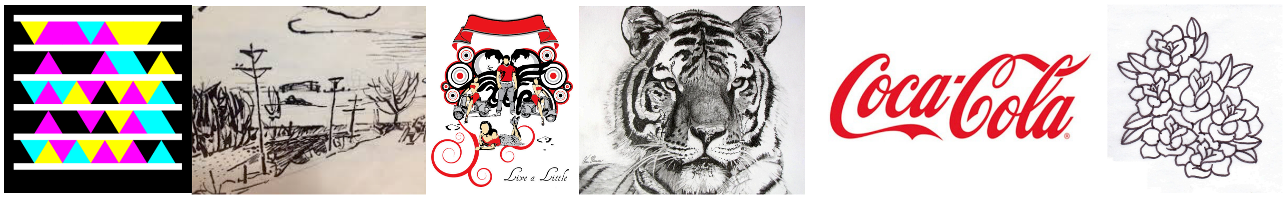


Figure 3: Non-artcode examples selected from the Artcodes Dataset

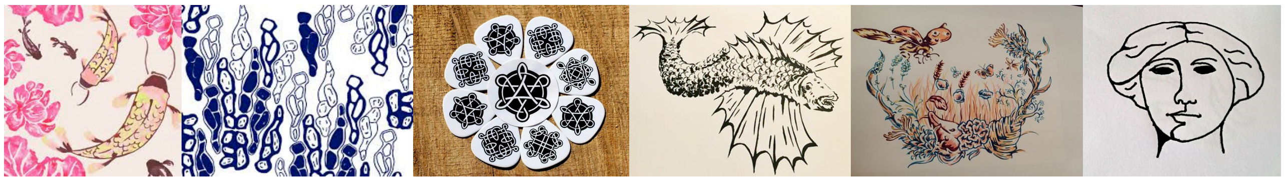


Figure 4: Artcode examples selected from the Artcodes Dataset. Artcodes are visually hidden markers which can be decoded, like Barcodes.

to represent different parts of an object. Dalal et al. [12] used the HoG as feature sets and shown that this descriptor significantly outperformed existing features for human detection. These HoG descriptors are reminiscent of orientation histograms and HoG in SIFT; however, to improve performance, they were computed on a dense grid of uniformly spaced cells and used overlapping local contrast normalizations. These features are effective to represent local image patches, such as geometry and appearance, but they are inappropriate for describing such global information as topology.

3 ARTCODES DETECTION

As shown in Figure 2, the surface of some objects (menu, plate, and mat) in the first image are decorated with Artcodes. To alert people to the presence of Artcodes and trigger their decoding, the first step is to detect their presence and then locate them (2nd image), and then decode the Artcodes by constructing the *adjacency* tree of each Artcode (3rd image) — the code of this Artcode is a string of numbers of *leaves* in each node. In this paper, we address the first step: recognizing the presence of Artcodes in digital images.

As human-designable aesthetic visual markers, Artcodes include a recognisable *foreground* part and *background* imagery (Figure 1) [25]. As shown in Figure 1, the recognisable part of an Artcode contains a closed boundary that is split into several regions (usually five), with each region containing one or more *blobs* – solid objects disconnected from the region edge – the code embedded in an Artcode is a string of numbers of *blobs* (leaves) in each connected region: for example, the code in Figure 1 is "1.1.2.3.5". Additionally, background imagery can be added to the recognisable part of an Artcode to enhance the aesthetics, but only if it does not break the Artcode's topological structure. More information about Artcodes can be found, for example, in the work of Meese et al. [25], and Benford et al. [4].

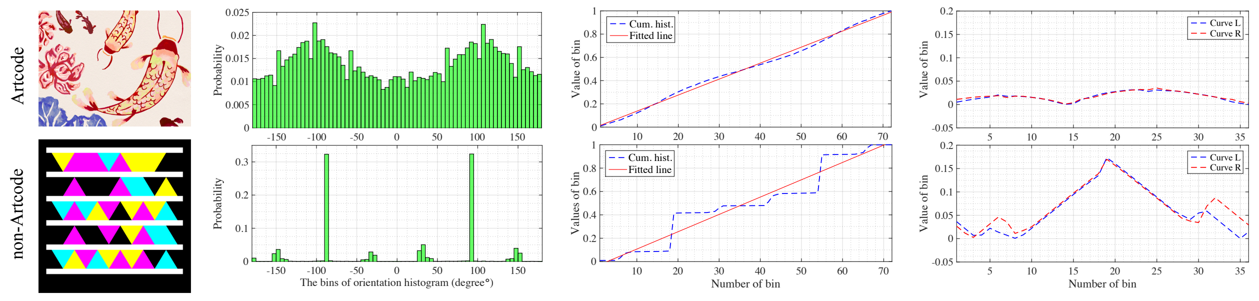
Artcode detection is a binary classification problem, classifying an input image or video sequence as either containing an Artcode, or not — labelled "*Artcode*" or "*non-Artcode*" classes, in this study. The Artcode class follows the topological definition of Artcodes, whereas the non-Artcode class is simply images that do not conform to these topological rules. As can be seen from the examples in Figures 3 and 4, there is no obvious difference in geometrical shape or appearance between Artcodes and non-Artcodes; here, we wish to uncover the hidden topological properties. The geometrical variation associated with Artcodes is very different to that of other well-known well-structured markers (such as Barcodes [24], QR codes, ARTags, ARToolKit [21], and reacTIVision [3]).

4 SOH FEATURE DESCRIPTOR

The shape of orientation histogram (SOH) is used for classification between the two categories: Artcode and non-Artcode. This section describes the motivation for the SOH and the process of constructing a SOH feature vector. Orientation histograms or oriented gradients have been well studied for the description of local information in various applications. Previous work [2, 12, 16, 22, 26, 27, 29] based on gradient orientation only concerns the geometric representation property of the orientation histograms, but pays little attention to the shape of the histogram itself, which is a potential property for describing the topological property of image patches. Being inherently topological visual markers with predefined topological structure (Section 3), any geometrical shapes are possible for a valid Artcode; therefore traditional geometrical and binary feature descriptors are inappropriate for representing this class of objects.

4.1 Shape of orientation histogram

Unlike well-structured visual markers, Artcodes are defined by following predefined topological rules. Artcode detection in images can then depend on their specific topological structure, a closed boundary containing several hollow regions dominating the decodable topology of Artcodes. Considering the ideal situation, that all the connected hollow regions in Artcode are circles, then the directions of the gradient at the points (imagining a line passing through the circle and its center) of the circles are collinear with the *normal vector* at those points. The gradients at the four points intersected by the line are two pairs of opposite directional vectors,



(a) Input images (b) Orientation histograms (c) Cumulative histograms (d) Distance curves

Figure 5: Orientation histograms, cumulative histograms, and distance curves calculated from the Artcode (top in (a)) and non-Artcode (bottom in (a)) images. (b) shows the two orientation histograms of Artcode and non-Artcode, where the top histogram is more symmetric and smoother than the bottom histogram; (c) illustrates the cumulative histogram curves of the two classes of images; and (d) shows the left and right sections of the distance curves — the top figure also clearly shows higher levels of both symmetry and smoothness between the left and right parts of the cumulative curve than that of non-Artcode. Our SOH feature vector is mainly built by comparing the similarity between the two curves.

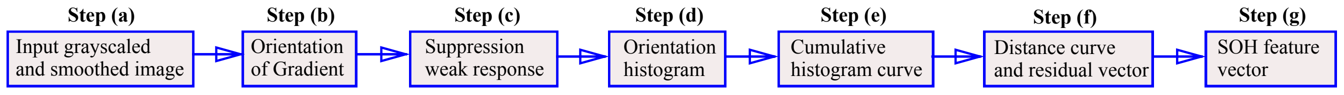

Step (a)
Input grayscaled and smoothed image
Step (b)
Orientation of Gradient
Step (c)
Suppression weak response
Step (d)
Orientation histogram
Step (e)
Cumulative histogram curve
Step (f)
Distance curve and residual vector
Step (g)
SOH feature vector

Figure 6: The process of constructing SOH feature vector

assuming the intensity of boundary is uniform, then the orientation of all the gradients around the circles are uniformly distributed from $-180°$ to $180°$. Therefore, the orientation histogram will be uniformly distributed, with its left ($-180°$ to $0°$) and right half ($0°$ to $180°$) being symmetric (the line perpendicular to the x-axis at $0°$ is the symmetry axis). The two properties in this paper that we will use to interrogate these histograms are smoothness (waviness) and symmetry of the orientation histogram.

As shown in Figure 5(b), the top and bottom figures are the orientation histograms from Artcode and non-Artcode, respectively. The orientation histogram in bottom figure of Figure 5(b) displays a certain level of symmetry but is highly non-uniformly distributed, whereas the top figure shows both symmetry and smoothness. This observation inspires us to employ both symmetry and smoothness of the orientation histogram for the representation of topological structure of an image, and use them as Artcode classification features, which we call a Shape of Orientation Histogram (SOH).

4.2 Feature vector construction

We present, in this section, details of how to describe the Shape of Orientation Histogram, particularly, to quantify the level of symmetrical similarity and smoothness of orientation histogram, and construct the appropriate feature vector.

4.2.1 Orientation histogram. As shown in Figure 6, the input image is first converted into a grayscale image and smoothed using a small Gaussian kernel to remove the color information and noise (step (a)). Then we calculate the magnitude and orientation of the gradient at each pixel of the grayscale image (step (b)). We remove the orientation of pixels whose corresponding magnitude is below a given threshold (step (c)). We then calculate the orientation histogram from the suppressed orientation map and *normalize* the values of bins using $v_i = \frac{C_i}{N}, i = 1, \cdots, nBins$, where C_i and N are the number of elements in the ith bin and in the input data, respectively, $nBins$ is the number of bins in this orientation histogram (we give $nBins = 72$ in this study, i.e. the range of bin is $5°$) (step (d)).

4.2.2 Cumulative orientation histogram. The symmetry of an orientation histogram can be calculated directly by comparing the left and right sections of the orientation histogram using similarity measures. The smoothness of the orientation histogram is difficult to represent directly from orientation histogram. The orientation histogram is sensitive to image rotation, but its level of smoothness and symmetry is invariant to image orientation (see Figure 7). Thus, we calculate *cumulative* orientation histogram (step (e)), which is a special orientation histogram whose kth bin satisfies: $cumbin(k) = \sum_{i=1}^{k} bin(i), k = 1, \ldots, nBins$, where $bin(i)$ is the value of the ith bin of the orientation histogram. Using a cumulative orientation histogram, the smoothness is visually represented: the convex and concave parts of the cumulative orientation histogram reflect the increase and decrease in corresponding parts of the orientation histogram. We next enter into step (f): fit a straight line (red line in Figure 5(c)) for the cumulative histogram curve, and then calculate the distances from this straight line, and generate the distance curves (see Figure 5(d)).

4.2.3 Construction of SOH feature vector. As presented in previous sections, we need to quantify the symmetry and smoothness of the orientation histograms. We calculate these two aspects from the distance curve, separating the distance curve into two uniform parts: left and right curves (denoted as C_l and C_r, as illustrated in Figure 5(d)), and then compute the similarity between the two

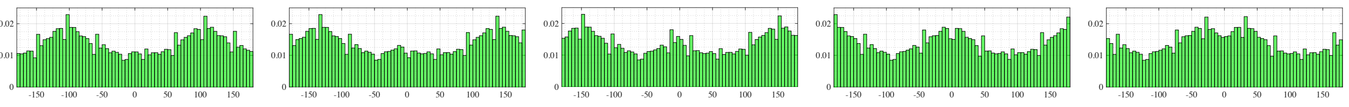


Figure 7: Orientation histogram of the images under rotations: 15°, 45°, 60°, 90°, **and** 120°. **The level of symmetry and smoothness of the orientation histogram of rotated images are preserved, showing SOH is invariant to image rotations.**

curves using similarity metrics. The similarity metrics used in this study are *Procrustes* and *Chi-square* distance. Formally, the two curves with n points are denoted as $C_l = \left\{\left(x_i, y_i\right) \mid i = 1, \ldots, n\right\}$, and $C_r = \left\{\left(u_i, v_i\right) \mid i = 1, \ldots, n\right\}$, where $n = \frac{nBins}{2}$.

Procrustes analysis is a form of statistical shape analysis of a set of shapes. Before calculation of Procrustes distance, it is necessary to perform Procrustes superimposition: *translation*, *rotation*, *scaling*, and *reflection*. After transformation, the Procrustes distance between the two curves C_l and C_r, is computed by:

$$proDist\left(C_l, C_r\right) = \sqrt{\sum_{i=1}^{n}\left((x_i - u_i)^2 + (y_i - v_i)^2\right)} \tag{1}$$

In this case, the left and right parts of the distance curve are already aligned, thus only reflection is required. Since the x-coordinates of the points on the distance curve are always zero, Equation 1 is simplified as:

$$proDist\left(C_l, C_r\right) = \sqrt{\sum_{i=1}^{n}\left((y_i - v_i)^2\right)} \tag{2}$$

Additionally, *Chi-square* distance is used to measure the similarity between two histograms or shapes, in this case, the *Chi-square* distance between C_l and C_r is denoted as:

$$\chi^2\left(C_l, C_r\right) = \sum_{i=1}^{n}\frac{(y_i - v_i)^2}{(y_i + v_i)} \tag{3}$$

We also measure the sensitivity of change of the distance curve by computing the similarity between the first derivatives of C_l and C_r usling Procrustes distance, denoted as $proDist(dC_l, dC_r)$.

The smoothness of orientation histogram is measured by the *mean* and *std*. (standard deviation) of the *residual* vector (Figure 6 step (f)), where the residual is the *difference* between the fitted line p and cumulative histogram curve *cumhist* (i.e. the vertical distance between the red and blue lines in Figure 5(d)). Therefore, the residual vector has the same dimensions as the cumulative histogram. Formally, the residual vector **r** is denoted as:

$$\mathbf{r} = \left\{r_i \mid r_i = |p_i - cumhist_i|, i = 1, \ldots, nBins\right\} \tag{4}$$

where p_i and $cumhist_i$ are the values of the fitted line and the i^{th} bin of the cumulative histogram, respectively. Then we have two variables, $mean(\mathbf{r})$ and $std(\mathbf{r})$, to describe the smoothness of the orientation histogram. The quantities describing the symmetry and smoothness combined with other global image statistics compose the 7-dimensional SOH feature vector:

$$\mathrm{S} = \left\{S_1, S_2, S_3, S_4, S_5, S_6, S_7\right\} \tag{5}$$

where $S_1 = proDist(C_l, C_r)$; $S_2 = \chi^2(C_l, C_r)$; $S_3 = proDist(dC_l, dC_r)$; $S_4 = mean(\mathbf{r})$; $S_5 = std(\mathbf{r})$; $S_6 = mean(im)$; and $S_7 = std(im)$ — im is the grayscaled and smoothed input image.

5 EXPERIMENTAL EVALUATION

5.1 Dataset

To study the Artcodes detection problem, we created a dataset of Artcode and non-Artcode images. Artcodes are hand crafted by designers by following the predefined drawing rules, while non-Artcodes are the images not containing any Artcodes. Figure 4 presents some of Artcode examples selected from the Artcode dataset: *fish*, *Scotland knots*, *flowers*, and *human face*, which are very different in geometry. Rather than the completely randomly selected images, the non-Artcodes were intentionally picked to be confusing for such a system. Figure 3 shows six non-Artcode images, including a *Microsoft Tag*, *trees*, an *advertisement*, *tiger*, *Coco-cola logos*, and *flowers*. Although they are human created images that are visually very similar to Artcodes, they do not have any imposed topological structure. Thus it is difficult, or even impossible, to recognize the presence of Artcodes through inspection of the appearance and geometry alone. In this dataset, we collected non-Artcodes from various human-created images, like *logos*, *drawings*, *advertisements*, *paintings*, and *graphics*. Other types of images such as *natural scenes*, *human images*, *daily life images*, were excluded, as these categories are obviously different from Artcodes in visual aspects, and thus would be easy to make correct predictions.

Because Artcodes are manually created by designers, the number of available Artcodes is currently small (especially considering the large number of available non-Artcodes), but work is ongoing to extend this dataset. A consequence of this limited number of available Artcodes is an imbalanced dataset, with a larger number of non-Artcode images, containing 47 Artcode and 148 non-Artcode images. However, this does reflect the real-life expected usage, as many more non-Artcode images will be presented to the camera than Artcodes.

5.2 Classifier

Since Artcode detection and the decoding system are deployed via a mobile platform, speed and memory are key consideration factors for choosing an appropriate classifier. Random forests [6], as an ensemble learning method, can provide accurate prediction for classification. More importantly, the random forests technique is efficient when running, and is sufficient for a real time application. Therefore, Random forests were taken as the classifier in this study.

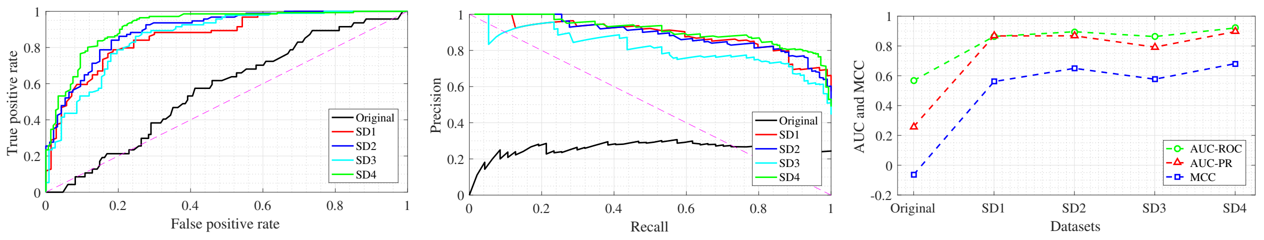
Random forests were proposed by Breiman in 2001, as an ensemble of decision trees such that each tree depends on the values of a random vector sampled independently, with the same distribution for all trees in the forest. Random forests have desirable characteristics for the learning task in this study: accurate as Adaboost[17]; robust to outliers and noise; faster than bagging and boosting; they have intuitive, easy to tune parameters — only the number of decision trees required to consider.

Table 1: SOH feature vector examples selected from SD4 dataset

S_1	S_2	S_3	S_4	S_5	S_6	S_7	Class
0.087523	0.005356	0.086905	204.19	69.02	0.005816	0.004186	1
0.096828	0.0070071	0.13416	86.394	47.851	0.005721	0.0032236	0
0.15182	0.0036975	0.044452	147.78	32.841	0.015098	0.0080612	1

Table 2: Original and SMOTED datasets

	Original	SD1	SD2	SD3	SD4
#Artcodes.	47	94	94	94	141
#non-Artcodes.	148	70	94	117	147



(a) ROC curves **(b) PR curves** **(c) AUC and MCC**

Figure 8: Performance comparison among the classifiers (nTrees = 80) **trained on datasets: Original, SD1, SD2, SD3, and SD4.**

It is well known that learning from an imbalanced dataset usually produces biased classifiers that have higher predictive accuracy over the *majority* class, but poorer predictive accuracy over the *minority* (rare) class [33]; we therefore first deal with this imbalance in the Artcode dataset in the next section.

5.3 Dealing with imbalance

A number of techniques [8, 9, 13, 14, 18, 33] have been proposed to handle imbalanced data learning, mainly including: methods at data or at algorithmic levels. Methods at data level ([8, 9, 18, 33]) attempt to balance distributions by examining the representative proportions of class examples in the dataset, while methods at algorithmic level ([13, 14]) consider the costs associated with misclassifying examples, also known as *cost-sensitive learning*. In this section, we only describe the methods working on data level, a comprehensive survey on investigating learning from imbalanced data can be found in [20].

One simple and effective imbalance handling technique is random resampling, either random oversampling or undersampling. The former randomly adds samples from the *minority* class and augments the original dataset, whereas the latter achieves a balanced distribution by randomly removing samples from the *majority* class. Random oversampling simply appends replicated data to the original dataset, easily leading to overfitting. With regard to avoid overfitting, Chawla et al. [8] proposed a synthetic sampling method – the Synthetic Minority Oversampling TEchnique (SMOTE), where the minority class is oversampled by creating "synthetic" examples rather than simply oversampling with replacement. SMOTE generates synthetic examples in a less application-specific manner, by working in "feature space" rather than "data space". The minority class is oversampled by taking each minority class sample and introducing synthetic examples along the line segments connecting any/all of k nearest neighbors of the minority class. The synthetic examples are generated by the following formula:

$$\left\{\mathbf{f}_{new} \mid \mathbf{f}^{n}_{new} = \mathbf{f} + r \cdot \left(\mathbf{f} - \mathbf{f}_i\right), i = 1 \ldots k; n = 1 \ldots \lfloor N/100 \rfloor\right\} \quad (6)$$

where $\mathbf{f}_{new}$ and $\mathbf{f}_i$ are the new set of synthetic feature vectors (examples) and the i^{th} neighbor of the feature vector $\mathbf{f}$ under consideration, respectively; r is a random number between 0 and 1; and $N\%$ denotes the amount of SMOTE. How many neighbors are randomly chosen from the k nearest neighbors is based on the required amount of synthetic examples ($(N/100) \cdot T$, where T is the number of minority class samples).

Obviously, SMOTE is unable to be directly used in an Artcode dataset that only contains images. Due to the proposed SOH feature vector, each image in the Artcode dataset will be converted into a quantitative feature vector (Equation 5), producing a *quantitative* Artcode dataset with same amount of samples as the original Artcode dataset. In this dataset, Artcode and non-Artcode classess are assigned integers 1 and 0, respectively, as shown in Table 1.

To obtain an "optimal" dataset contributing to best classification performance, four SMOTED datasets (SDs) (SD1, SD2, SD3, and SD4) (Table 2) with different numbers of Artcodes and non-Artcodes were generated using the SMOTE algorithm. We compare the performance of the classifiers evaluated on Original and SDs using ROC and PR curves and their respective area under curve (AUC). Because MCC (Matthews Correlation Coefficient[23]) is an informative measure for both balanced and imbalanced datasets, we also use MCC. As illustrated in Figure 8, the classifiers on the SDs greatly outperform Original dataset in terms of ROC, PR curves, AUC-ROC, AUC-PR and MCC, showing that the SDs significantly enhance the classifier's performance. Among the four SDs, the classifier evaluated on SD4 achieves the best performance in terms of all four measures — the improved performance is not very large though. Therefore, we used SD4 as the dataset for further experimental evaluation of the proposed classifier.

5.4 Experimental setup

We implemented the classification model using Matlab, and evaluated its performance using k-fold cross-validation on the SD4 dataset. Because random forests are used in the classifier, the performance exhibits a certain level of variation on the execution due

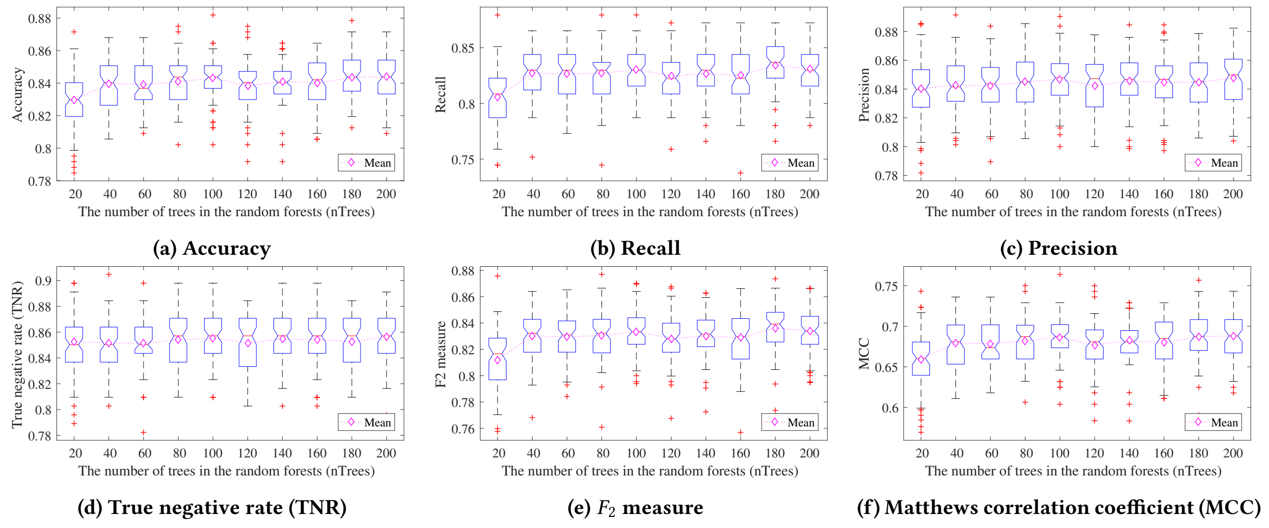

(a) Accuracy (b) Recall (c) Precision
(d) True negative rate (TNR) (e) F_2 measure (f) Matthews correlation coefficient (MCC)

Figure 9: Performance measures of the proposed classifier evaluated on the SD4 dataset.

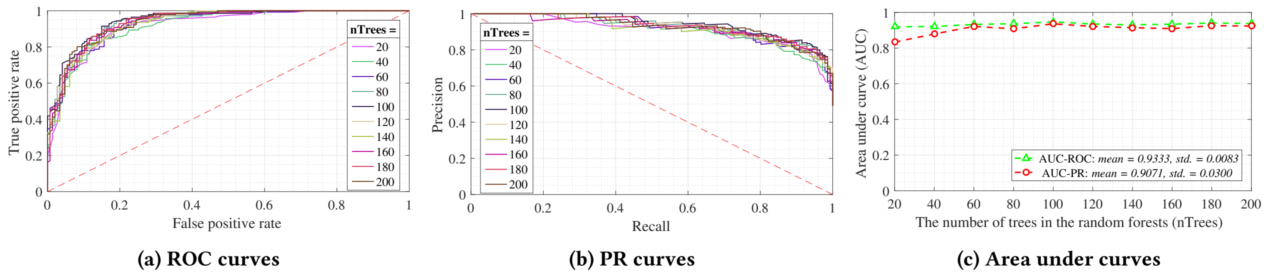

(a) ROC curves (b) PR curves (c) Area under curves

Figure 10: Performance curves of the proposed classifier with different values of nTrees**, evaluated on the SD4 dataset.**

to the random variable selection from the feature vector. We run this procedure 50 times and produce *Notched* boxplots of quantitative performance measures for performance evaluation. In this experiment, a random forests were used, and we conducted experiment to study the impact of the tuning parameter — the number of trees (nTrees) in the forests — on the classifier's performance.

5.4.1 Performance metrics. Considering the sensitivity of single performance metrics, we selected a group of measurements to provide an informative view of the proposed classifier's performance: Recall, Precision, TNR (True negative rate), Accuracy, F_2 measure, and MCC (Matthews Correlation Coefficient) [23] were all used to examine the classifier's performance.

Recall (sensitivity) and precision are two measures focusing on the positive examples (Artcodes) and predictions. Their importance varies over learning tasks: precision, for example, being more desirable than recall in information retrieval. However, in the case of Artcode detection, we may desire higher Recall than Precision, because recognizing the presence of Artcodes is the basis for their later decoding in augmented reality. Similarly, TNR (specificity) measures the proportion of negatives (non-Artcodes) that are correctly identified as such. Accuracy, the F_2 measure, and MCC measure the overall performance of the classifier, considering both positive and negative classes. Accuracy is the overall proportion of correct predictions, for both Artcode and non-Artcode classes, and is a simple way of describing a classifier's performance on the given dataset. However, Accuracy is sensitive to size differences among classes. F_2 measure is a special instance of F_β measure, which F_β measure uses the weighted harmonic average of Precision and Recall to evaluate the classifier's preciseness (β is the factor used to control the importance of recall over precision). We in this experiment, considering the desirability of recall over precision, used the F_2 measure ($\beta = 2$), making recall twice importance as precision. Compared with Accuracy, F_2 measure provides more insight into the performance of a classifier, but can be as sensitive to data distributions as Accuracy, although the sensitivity is less. MCC is in essence a correlation coefficient between the observed and predicted classifications, incorporating true and false positives and negatives. It is generally regarded as one of the best measures for classifier performance evaluation [30], and remains effective even if the dataset is imbalanced.

Additionally, ROC (Receiver operating characteristic) and PR (Precision-recall) curves and their area under curve (AUC-ROC and AUC-PR) were used to illustrate the classifier's performance. ROC and PR curves are commonly used to evaluate the performance of the classifier in binary classification problems in machine learning. ROC curves plot the false positive rate on the x-axis versus the true positive rate on the y-axis, while PR curves plot Recall on the x-axis and Precision on the y-axis. The goal of ROC curves is to be in the upper-left-hand corner, whereas the goal of PR curves is to be in upper-right-hand corner. The closer the curves approach to these two corners ([0, 1] and [1, 0], respectively), the better is the performance of the classifier. ROC and PR curves provide a visual inspection of the classifier's performance, AUC-ROC and AUC-PR provide a single numeric metrics for evaluating the classifier's performance. Models with higher AUC values are preferred over those with lower AUCs. However, when dealing with highly skewed datasets, PR curves present a more informative picture of a classifier's performance.

5.5 Results

Figures 9 and 10 present the classifier's performance with different values of nTrees (the number of decision trees in random forests). Figure 9 presents the six performance metrics generated from the 50 rounds of running cross-validation of the classifiers. This figure also shows the average values (pink diamonds in boxplots) of these performance metrics also calculated from 50 rounds of running.

The impact of nTrees on the classifier's performance illustrated in Figures 9 and 10, where the classifier stabilizes at about nTrees = 40, in terms of the evaluation metrics, with a small variations on the performance when nTrees = $40, \ldots, 200$. This is reflected by whether the notches between the classifiers with different values of nTrees overlap or not. As shown in Figures 9(a), (e), and (f), the notch of the first box (nTrees = 20) does not overlapped with the notches of other boxes in these three overall performance metrics (Accuracy, F_2 measure, and MCC). According to the definition of notched boxplot, this means that the medians of the three metrics differ at a 95% confidence level. On the other hand, the notches of these boxes (nTrees = $40, \ldots, 200$) are all overlapped, indicating their medians are not significantly different. This is also shown in Figure 10(c) by the sharp increase of PR-AUC when the value of nTrees is increased from 20 to 40. In the following analysis, we focus on the performance evaluation of the classifiers with nTrees ≥ 40.

Regarding the classifier's performance predicting the positive class (Artcodes), as measured by Recall and Precision, the classifier obtains scores of about 0.82 and 0.84 for Recall and Precision, respectively. This means that the classifier correctly predicts 82% of Artcodes from the given dataset and predicts an Artcode class with 84% probability to be an true Artcode. Likewise, the classifier performs well on predicting the negative class (non-Artcodes) as predicting positive class, as measured by TNR (Figure 9(d)), with 85% of non-Artcodes correctly predicted. Formally, given 1 and 0 to denote Artcode and non-Artcode classes, respectively, and $\hat{Y}$ as the predicted class of the true class Y, the proposed classifier has following conditional probabilities:

$$\text{Recall} = \Pr\left(\hat{Y} = 1 \mid Y = 1\right) = 0.82 \tag{7}$$

$$\text{Precision} = \Pr\left(Y = 1 \mid \hat{Y} = 1\right) = 0.84 \tag{8}$$

$$\text{TNR} = \Pr\left(\hat{Y} = 0 \mid Y = 0\right) = 0.85 \tag{9}$$

Accuracy, F_2 measure, and MCC assess the classifier's overall performance, considering both positive and negative classes. As shown in Figures 9(a) and (e), the Accuracy and F_2 measure of the classifiers are about 0.84 and 0.83, respectively, indicating that the classifier correctly predicts 84% proportion of samples in SD4 dataset and achieves a good trade-off (0.83) between preciseness and robustness. Because the given dataset SD4 is almost balanced, Accuracy is as an effective performance measure as is F_2 measure: both of them show good overall performance of the proposed classifier. The MCC of the classifier on the given dataset is approximately 0.68, as a Pearson correlation coefficient for two binary variables, MCC has similar interpretation as Pearson correlation coefficient: MCC = 0.68 exhibits a *strong* positive agreement between predictions and observations, showing the good predictive ability of the classifier.

The good predictive ability of the classifier is also visually reflected by the ROC and PR curves. In Figures 10(a) and (b), the ROC and PR curves of the classifiers with different numbers of nTrees are all approaching to the *upper-left-hand* and *upper-right-hand* corners, respectively, showing good overall performance of the classifiers. Moreover, the area under the curves of the classifiers are high, compared to random change, as shown in Figure 10(c), with mean of 0.93 (std.=0.0083) and 0.91 (std.=0.0301) across all different values of nTrees, respectively. This shows the *probability* (Equation10) that the classifier will rank a randomly chosen positive example (Artcode) higher than a randomly chosen negative example (non-Artcode), denoted as follows, indicating the good performance of the classification model.

$$\Pr\left(score(Y = 1) > score(Y = 0)\right) = 0.93 \tag{10}$$

6 CONCLUSIONS

In this paper, we have presented a new topology-based system for the detection of the presence of "invisible" yet structured markers in images. We proposed a new system for Artcode detection, which shows the feasibility of using such geometrically-variable markers in free-form images, thanks to a machine learning-driven approach. An Artcodes datasets was collected for the study. Artcodes are handcrafted by designers, which by their nature means that the number of Artcode samples is limited, therefore introducing an imbalance to the dataset. SMOTE was used to tackle this imbalance, creating an augmented dataset with relatively-balanced class distribution, showing great performance enhancement.

To detect such free-form codes, traditional geometry-based features were not suitable. Therefore, the SOH feature descriptor based on the symmetry and smoothness of the orientation histogram was proposed in this paper, and showed its effectiveness on the classification of Artcodes versus non-Artcodes in a random forest framework. Here we use Artcodes as one example case for recognizing the presence of similar hidden visual markers in a real-life context. In this paper, we demonstrate how we can use topological information about the markers to detect their existence in images. Our future work will include deploying this system via the mobile platform to study its impact on the user experience in augmented reality.

ACKNOWLEDGMENTS

The author acknowledges the financial support from the International Doctoral Innovation Centre, Ningbo Education Bureau, Ningbo Science and Technology Bureau, and the University of Nottingham. This work was also supported by the UK Engineering and Physical Science Research Council [grant No. EP/L015463/1].

REFERENCES

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. 2008. Speeded-up robust features (SURF). *Computer vision and image understanding* 110, 3 (2008), 346–359.

[2] Serge Belongie, Jitendra Malik, and Jan Puzicha. 2001. Matching shapes. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, Vol. 1. IEEE, 454–461.

[3] Ross Bencina, Martin Kaltenbrunner, and Sergi Jorda. 2005. Improved topological fiducial tracking in the reactivision system. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on.* IEEE, 99–99.

[4] Steve Benford, Adrian Hazzard, Alan Chamberlain, Kevin Glover, Chris Greenhalgh, Liming Xu, Michaela Hoare, and Dimitrios Darzentas. 2016. Accountable artefacts: the case of the Carolan guitar. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'16)*. ACM, San Jose, CA, USA, 1163–1175.

[5] Steve Benford, Adrain Hazzard, Alan Chamberlain, and Liming Xu. 2015. Augmenting a Guitar with Its Digital Footprint. In *Proceedings of International Conference on New Interfaces for Musical Expression (NIME'15)*. Louisiana, USA, 303–306.

[6] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[7] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. 2010. Brief: Binary robust independent elementary features. *Proceedings of the 2010 European Conference on Computer Vision (ECCV'10)* 6314 (2010), 778–792.

[8] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[9] Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. 2003. SMOTEBoost: Improving prediction of the minority class in boosting. In *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 107–119.

[10] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20, 3 (1995), 273–297.

[11] Enrico Costanza and Jeffrey Huang. 2009. Designable visual markers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1879–1888.

[12] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 1. IEEE, 886–893.

[13] Charles Elkan. 2001. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, Vol. 17. Lawrence Erlbaum Associates Ltd, 973–978.

[14] Wei Fan, Salvatore J Stolfo, Junxin Zhang, and Philip K Chan. 1999. AdaCost: misclassification cost-sensitive boosting. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML'99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 97–105.

[15] Mark Fiala. 2005. ARTag, a fiducial marker system using digital techniques. In *Proceedings of 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 2. IEEE, San Diego, CA, USA, 590–596.

[16] William T Freeman and Michal Roth. 1995. Orientation histograms for hand gesture recognition. In *International workshop on automatic face and gesture recognition*, Vol. 12. 296–301.

[17] Yoav Freund and Robert E Schapire. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*. Springer, 23–37.

[18] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. 2005. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing*. Springer, 878–887.

[19] Chris Harris and Mike Stephens. 1988. A combined corner and edge detector. In *Alvey vision conference*, Vol. 15. Manchester, UK, 50.

[20] Haibo He and Edwardo A Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering* 21, 9 (2009), 1263–1284.

[21] Hirokazu Kato and Mark Billinghurst. 1999. 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.* IEEE, 85–94.

[22] David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60, 2 (2004), 91–110.

[23] Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure* 405, 2 (1975), 442–451.

[24] Michael W Maynard. 1993. Classifying apparatus and method. US Patent 5,232,099. (3 Aug. 1993).

[25] Rupert Meese, Shakir Ali, Emily-Clare Thorne, Steve D Benford, Anthony Quinn, Richard Mortier, Boriana N Koleva, Tony Pridmore, and Sharon L Baurley. 2013. From codes to patterns: designing interactive decoration for tableware. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*. ACM, Paris, France, 931–940.

[26] Krystian Mikolajczyk, Cordelia Schmid, and Andrew Zisserman. 2004. Human detection based on a probabilistic assembly of robust part detectors. In *Proceedings of 2004 European Conference on Computer Vision (ECCV'04)*. Springer, 69–82.

[27] Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio. 2001. Example-based object detection in images by components. *IEEE transactions on pattern analysis and machine intelligence* 23, 4 (2001), 349–361.

[28] Kher Hui Ng and Shazia Paras Shaikh. 2016. Design of a mobile garden guide based on Artcodes. In *Proceedings of 2016 International Conference on User Science and Engineering (i-USEr'16),*. IEEE, 23–28.

[29] Constantine Papageorgiou and Tomaso Poggio. 2000. A trainable system for object detection. *International Journal of Computer Vision* 38, 1 (2000), 15–33.

[30] David M. W. Powers. 2011. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies* 2, 1 (2011), 37–63.

[31] Edward Rosten and Tom Drummond. 2006. Machine learning for high-speed corner detection. In *Proceedings of 2006 European Conference on Computer Vision (ECCV'06)*. Springer, 430–443.

[32] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. ORB: an efficient alternative to SIFT or SURF. In *Proceedings of 2011 IEEE International Conference on Computer Vision (ICCV'11)*. IEEE, 2564–2571.

[33] Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. 2010. RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 40, 1 (2010), 185–197.

[34] ISO/IEC International Standard. 2000. QR Code. (2000).

[35] Emily-Clare Thorn, Stefan Rennick-Egglestone, Boriana Koleva, William Preston, Steve Benford, Anthony Quinn, and Richard Mortier. 2016. Exploring large-scale interactive public illustrations. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*. ACM, 17–27.

[36] Engin Tola, Vincent Lepetit, and Pascal Fua. 2008. A fast local descriptor for dense matching. In *Proceedings of 2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*. IEEE, 1–8.