Reverse Indexing for Reading Graffiti Tags

Christian Thurau and Christian Bauckhage
Fraunhofer IAIS, 53757 St. Augustin, Germany
http://www.iais.fraunhofer.de

Abstract

In this paper, we consider the problem of automatically reading graffiti tags. As a preparatory step, we create a large set of synthetic graffiti-like characters, generated from publicly available true type fonts. For each character in the database, we extract a number of scale independent local binary descriptors. Then, using binary non negative matrix factorization, a sufficient number of basis functions are learned. Basis function coefficients of novel images can then be directly used for hashing characters from the database of prototypes. Finally, graffiti tags are recognized by means of a localized, spatial voting scheme.

1. Introduction

The general public often regards graffiti tags as an unsightly damage for graffiti writers tend to vandalize property without the owner’s consent (see the example of public transport vehicles in Fig. 1). In order to enable prosecution, authorities therefore often photograph tags before cleaning and removal and thus gather collections of images characterizing the activities of individual taggers. The resulting collections of images are typically huge and unstructured and there is considerable demand for techniques that would allow for sifting through these repositories efficiently. In this paper, we present an approach to automatically reading graffiti tags. Given a preprocessed picture of a graffiti tag, our aim is to automatically deduce which letters it contains. To the best of our knowledge, this is the first time that this is being attempted.

Even though conventional (handwritten) character recognition is known to be challenging by itself [2, 7], techniques that have been developed for this purpose do not apply to our problem for they assume the writing to follow language specific rules. In contrast to conventional character recognition, graffiti tags are usually drawn in different scales and orientations, on different surfaces by different tools, and in varying thicknesses. Often, the letters do overlap, are slanted, and contain irrelevant decorative components which may impede identification. Different geometric arrangements of the letters and embellishments of a tag are yet another challenge. In this paper, we therefore interpret the task of graffiti tag reading as an image retrieval problem. Following ideas proposed in [1], we map a novel graffiti tag onto a large set of synthetically generated (from graffiti true-type fonts) prototypic characters. In detail, the contributions of this paper are: (i) a fast reverse indexing method for binary shapes based on binary matrix factorization [8], (ii) a spatial relevance patch voting that is specifically tailored towards word reading using local image patches.

In Section 2 we explain how reverse indexing of binary patches leads to a fast and robust retrieval of associated patterns. Section 2.3 introduces a spatial voting scheme for single letter recognition in tags. Finally, we present experimental validation in Section 3.
2. Retrieving letters from graffiti tags

To focus on the problem of reading tags, we for now ignore the necessary image preprocessing steps. We assume that each character or graffiti tag is represented by a (possibly noisy) binary shape. Given the peculiarities and difficulties mentioned above and since we have to deal with a very large number of prototypic shapes in our database, conventional shape matching methods are not appropriate in our setting. Similarly, established methods for recognizing handwritten text are inappropriate, too, since they rely on extensive preprocessing for baseline and slant estimation [2], which is impossible in the case of graffiti. Thus, we follow a retrieval based approach towards character recognition. Based on a large number of synthetically generated letters, we classify against each single letter, i.e. carry out a one to many matching. The overall approach is summarized in Fig. 2 and will be explained in detail in the following.

2.1. Binary Codes

We propose a feature representation that is inspired by the well known bags-of-visual-words approaches. Given a shape (here a character of a graffiti tag), we first sample locations where image patches are extracted. This can be achieved by e.g. applying k-means clustering to all black pixels in a binary image, yielding a set of cluster centers $C = \{c_1, \ldots, c_n\}, c_i \in \mathbb{R}^2$. A dense distribution of cluster centers showed to be important.

Centered around each cluster center $c_i$ we extract a quadratic image patch $p_i$. The sizes $S = \{s_1, \ldots, s_n\}, s_i \in \mathbb{R}$ of the extracted binary patches are automatically determined. We measure the $s_i$ using the distance between $c_i$ and the nearest white pixel and set $s_i = \sigma \min_k ||w_k - c_i||^2$, $w_k, c_i \in \mathbb{R}^2$, where the $w_k$ are the white pixels of the shape image and $\sigma$ denotes a constant scaling factor. All extracted binary patches are finally scaled to a uniform size, e.g. of $5 \times 5$ or $10 \times 10$ pixels, using a simple image resizing procedure. Section 3 contains more details on parameter selection.

The proposed binary image patch representation is useful for representing shapes for three different reasons: (a) it is invariant to scale since the scale is determined strictly based on local conditions and the patches are afterwards scaled to uniform size, (b) patches extracted from the inner part of a shape and from the border region of a shape contain different amounts of local context, (c) each patch can be rearranged into a $d$ dimensional binary vector. As such, it also offers a unique hash key which is, at the same time, a meaningful part of the shape itself. A more detailed description of the used feature can be found in [5].

2.2. Binary Non-Negative Matrix Factorization

Using the extracted binary patches as a hashing key offers an exact descriptor for parts of the shape. However, it may make patches very sensitive to slight distortions or changes in the appearance. A change of only one pixel results in a completely novel descriptor. Furthermore, the relatively large number of all possible image patches (hashes), e.g. $2^{25}$ for a $5 \times 5$ results in a very sparsely covered feature space. While this is not a disadvantage if we would determine distances to a set of prototypical shapes, finding direct matches for specific hash keys becomes more difficult.

Therefore, we represent each patch as a binary code, similar to [6]. However, since our input feature vectors are also binary, we aim at finding a set of suitable binary basis functions that are able of reconstructing image patches using a fixed vector length. This can be efficiently achieved by means of Binary Matrix Factorization (BMF) [8].

BMF is an extension of standard Non-negative Matrix Factorization [3, 4] to binary data. Given a binary matrix $V$, we factorize it into two binary matrices $W,H$ such that $V \approx WH$. The problem can be represented as

$$\min J(W,H) = \sum_{ij} (V_{ij} - (WH)_{ij})^2 \quad (1)$$

subject to $H_{kj}^2 - H_{kj} = 0$ and $W_{ik}^2 - W_{ik} = 0$. A
solution is found by alternately solving for \( W \) and \( H \) and introducing Lagrangian multipliers \( \lambda \) and \( \mu \)

\[
\min J(W, H) = \sum_{i,j} (\Omega)^2 + \sum_{k,j} \lambda(\Theta)^2 + \sum_{i,k} \mu(\Gamma)^2
\]

where \( \Omega = V_{ij} - (WH)_{ij} \), \( \Theta = H_{k,j}^2 - H_{kj} \), and \( \Gamma = W_{ik}^2 - W_{ik} \). See [8] for further details on the correct update rules and initialization.

By application of BMF, we can represent patches by their corresponding coefficients \( H \). Note that during training, we apply BMF to the complete set of synthetically created single characters. For testing, we use the bases \( W_T \) found during training to reconstruct novel patches. Thereby, we map each novel patch onto the training basis vectors, i.e. we reconstruct each novel patch using binary patterns learned during training. The number of basis vectors encodes the length of coefficients and specifies with how many bits binary image patches should be encoded.

### 2.3. Spatial vote propagation

Extracted binary image patches and their corresponding coefficients \( H \) map exactly onto one hash key. If we encounter a previously unknown key (a key that did not occur during training), we treat the patch as unknown and do not consider it further (this happens rarely). Since we can associate a list of letters with each patch, each patch implicitly votes for a number of different letters. We ignore hashes that vote for too many letters since they obviously do not help in recognizing a particular letter (this threshold considerably influences precision rates, see also Section 3 for detailed parameter evaluation).

The hash-voting significantly improves computation time, since it now only depends on the actual number of extracted patches and not on the number of classes (or in our cases individual letters). It only takes between 0.25 and 0.8 seconds to classify against appearances of 10764 different letters. However, as we can assume that a spatial group of patches voting for one letter should be preferred over isolated votes, we found that considering the location and spatial relations of patch votes is crucial. We now continue with a more formal definition.

Each patch \( p_i \) votes for a particular letter \( l_k \) using the following scheme \( u_{ik}^n = |p_i^l|/|p_i^n|, n = 1 \ldots k \), where \( |p_i^l| \) denotes the number of occurrences of patch \( p_i \) in letter \( l_j \), and \( |p_i^n| \) denotes the total number of occurrences of patch \( p_i \) in the training data.

For spatial distributing of patch votes, we diffuse votes according to the pairwise distance of patches, i.e.

\[
\psi_i^k = \sum_j (1 + d_{i,j})^{-n} |p_{ij}^j|/|p_i^n|
\]

where \( d_{i,j} \) denotes the spatial distance between patch \( i \) and \( j \), and \( n \) weights the propagation to nearby nodes. We determine a winning letter \( s_i \) for each node using a winner takes all selection over the associated votes \( s_i = \arg \max_k u_{ik}^n \). The votes for all letters are accumulated and thereby select a list of letters that might be present in a given tag. Here, we only compute letter occurrences and not precise locations of individual letters. Since graffiti art does not impose a specific writing direction, precise locations are not very helpful for tag reading. However, letters can be localized by computing Parzen density estimates of the votes for a particular letter, as can be seen in Fig. 3.

### 3. Experiments

For the experiments we generated 10764 letters in different rotations from 78 publicly available fonts. We preselected fonts that were labeled as graffiti-like fonts. For each letter we extracted 200 patches. Then, we learned a BMF basis matrix \( W_T \) based on all patches. The resulting coefficient Matrix \( H_T \) was used to populate a hashtable, in which a list of letters is stored for each binary pattern.

We evaluated three distinct data sets, see Fig. 4 for examples. The first data set contains 576 synthetically generated tags of 4-6 letters. The tags consist of randomly rotated and placed letters. The variations range from slight rotations and occlusions to severe distortions. The second data set contains 200 synthetically generated tags that include, besides more severe rotation and occlusion, also random character placements. The third data set contains 150 scanned Graffiti tags cre-
ated by human graffiti writers. For data set one and two we used the same fonts for generating tags for testing and training, thus, we mostly had to deal with occlusion and rotations. For data set three we additionally had to deal with unknown fonts and noisy input.

For each tag, we extract 800 patches. Extracted patches are rescaled to uniform size $5 \times 5$, or $10 \times 10$, and mapped onto the BMF basis vectors $\mathbf{W}_T$. The resulting coefficients (or hash keys) $\mathbf{H}_{\text{test}}$ vote for a certain number of letters associated with the key. The proposed vote propagation usually leads to only a few (up to 10) resulting letters that are believed to be present in the graffiti tag. The results are summarized in Fig. 5, we varied the voting threshold and the degree of propagation for patch votes. Precision denotes the number of letters successfully retrieved against the total number of retrieved letters, recall denotes the number of successfully retrieved letters vs. the total number of characters visible. As expected, the first data set gave generally the best results. Obtaining precision/recall rates of around 65% is for most applications a convenient result. The second and third data set come up to rates between 20% and 60%. The third data set showed to be the most difficult with results at about 26% for both precision and recall. We attribute the decreased performance to the considerably greater variations in font use.

4. Conclusion

This paper introduced an approach to automatically reading graffiti tags. Although this is a computer vision problem of considerable practical relevance, we are not aware of previous attempts in this direction. We proposed an approach based on reverse indexing which adopts the idea of visual words to binary shape images. The proposed descriptors accord for local structures on context dependent scales. Using binary matrix factorization, we generate hash keys that are robust to distortions and simultaneously allow for rapid retrieval of similar characters from a large database of synthetic but readily available examples. Voting over these prototypes yields an estimation of which characters are present in a newly observed tag.

While our method performs robust on tags assembled from distorted versions of characters contained in the database, the performance on photographed examples is amendable. For future research, we want to make use of the fact that the retrieval process is mostly independent of the number of characters available to the system. Thus, we will extend the font database to allow for a better coverage of different tag/font styles encountered in the wild.

References