Ghost fingerprint detection using Local Binary Patterns

Réka Szabó - *s2007320*
r.szabo@student.utwente.nl

Abstract

During the process of taking a set of fingerprints for registration at a police office, a type of contamination called ghost fingerprints may occur. Ghosts render the print unsuitable for recognition, therefore the contaminated images must be excluded from the dataset. For this purpose, a ghost fingerprint detection software was developed using MatLab. The method is based on Local Binary Patterns to process the texture, and produces results with an average accuracy of 89% tested on 1000 images.

Keywords: ghost fingerprint, LBP.

Introduction

This work is part of a project for the Dutch police. During the process of taking a set of fingerprints for registration at a police office, a type of contamination called ghost fingerprints may occur. This happens with certain sensors during a calibration stage. Effectively, the remains of a previous fingerprint are automatically subtracted from a current one. Ghosts render the print unsuitable for recognition. The effect is illustrated in the figure below.

Example of a ghost fingerprint

The goal of this assignment was to develop a software based on Local Binary Patterns that can detect ghosts in a fingerprint by classifying blocks in an image as either: background (no fingerprint), uncontaminated, contaminated (i.e. ghost). Contaminated blocks/images are later referred to as *ghosty* blocks/images, while uncontaminated ones are referred to as *non-ghosty*.

The developed software follows the proposed procedure and produces 12.4% of false negatives and 9.6% of false positives on the tested 1000 images. The background extraction process uses the convex hull of the fingerprint. The LBP image of the original image is divided into blocks and each block is classified as either contaminated or uncontaminated by utilizing the white - dark pixel ratio of the blocks. Based on the results, the whole image is categorized as ghosty or non-ghosty.

The rest of the paper will be structured as follows. First, an overview is given on the related literature. Next, the methodology is presented with a detailed description of the algorithm. Afterwards, the experiments with the first test images are demonstrated and lastly, the final results are presented.

1 Related literature

The Local Binary Pattern texture operator was first introduced by Ojata et al. in 1999 [3]. It is based on the assumption that two complementary aspects belong to a texture locally: a pattern and its strength [4]. The original version is using 3×3 pixel blocks of an image. The center pixel serves as a threshold for the neighborhood pixels and the operation results in an 8-bit number. This binary number is then converted to a decimal number, being the LBP value of the central pixel. As there are 8 neighborhood pixels, 2^8 different labels can be obtained depending on the pixels in the block.

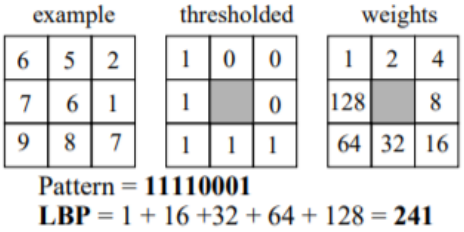

Figure 1.1: Calculation of the LBP [5].

Later, a more generic LBP operator was derived from the initial version by Ojala et al., allowing to change the size of a block by defining a radius and determining the neighborhood pixels accordingly [2].

Local Binary Patterns have been used in several applications since its first occurrence. The first paper presented an unsupervised texture segmentation method which uses distributions of local binary patterns and pattern contrasts or measuring the similarity of adjacent image regions during the segmentation process [3]. The later work of Ojala et al. uses LBP for robust texture classification by extending the definition of the LBP operator.

Ahonen et al. [1] presented a novel and efficient facial image representation based on LBP texture features. In their implementation, the face image is divided into several regions from which the LBP feature distributions are extracted and concatenated into an enhanced feature vector to be used as a face descriptor.

2 Methodology

With the help of the previously described Local Binary Patterns, the blocks in an image are classified as either contaminated or uncontaminated and accordingly the image is categorized as ghosty or not.

The method can be divided into four major parts:

1. Separation of background
2. Creating the LBP image and dividing the picture into blocks
3. Calculating the white - dark pixel ratio per block and the average on all blocks
4. Comparison to average and classification of the image

The first step is to separate the white background from the fingerprint. This is achieved by generating the convex hull of the image, using MatLab's built-in function *bwconvhull*.

The function can work with binary images only, and to get better results, it is used on the complement of the image. To separate the background completely, the pixels that are included in the convex hull are converted to Not a Number (NaN) value, which will be handy subsequently.

In the following step, the LBP image is created from the original picture. Local binary patterns are used to stress the differences of the texture. The LBP image highlights the irregularity of the ridges, whether the distance between them is relatively constant or diverse, perhaps they are damaged, as seen in Figure 2.1. Next, the picture is divided into square shaped blocks of *fixed size* for the operations that are executed afterwards.

Figure 2.1: From left to right: the original image,
image after background separation, the LBP image.

The decision of whether a fingerprint is ghosty or not, is based on the white - dark pixel ratio of the blocks of the LBP image. This derives from the observation that the ghosty images contain areas besides the normal fingerprint areas that are brighter due to disordered ridges. These irregular areas are highlighted on the LBP image by resulting in a higher number of dark pixels compared to the other parts of the image. By dark pictures, we mean the pixels which value is higher than 220 in a gray scale image. The number of white and darks pixels can change significantly depending on the location of the block, for instance on the edges, where the block contains little part of the fingerprint and the remainder is background. Therefore, the ratio of white and dark pixels is calculated within each block. If this number is *low enough*, the current block is considered to be ghosty. After classifying each block, we compare the number of ghosty blocks to a *proportion* of the number of blocks and decide whether the whole picture is in fact a ghosty fingerprint.

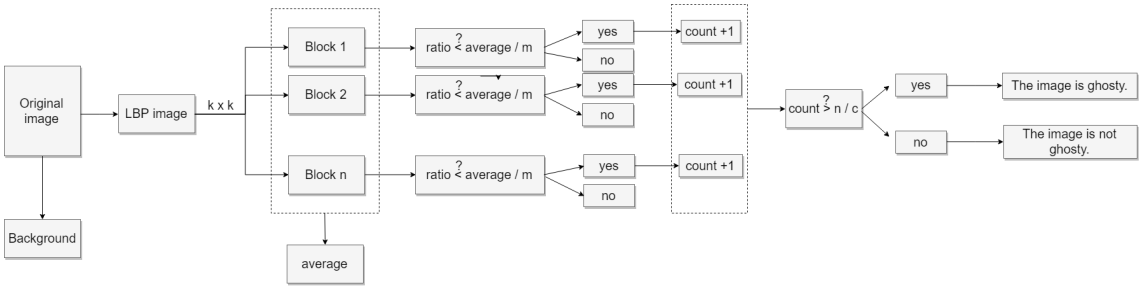

Figure 2.2: Decision-making algorithm.

Figure 2.2 illustrates the subsequent steps of the algorithm. The variables are discussed in more detail in the next section.

3 Experiment

In the previous section, three measures were mentioned that were not precisely defined, namely the size of the blocks, the threshold for the ratio and the proportion of the number of blocks for comparison. To create a training dataset, 200 images were selected with ghosts and 200 clear images were selected without any ghosts, both done manually.

In the algorithm, the above mentioned three values need to be predefined. The first one is the size of the square which equals a block. As it was mentioned in Section 2, a block is considered ghosty if the ratio of the white - dark pixels is *low enough*. The ratio of each block is compared to the average taken from each block. If the ratio is lower than the *average* divided by a certain m, then we consider the block to be ghosty. Similarly, if the number of ghosty blocks is higher than the number of blocks divided by a certain c, we consider the whole image to be ghosty.

To find the most suitable combination, the (k, m, c) triple was defined, where k denotes the size of the block, while m and c are the above mentioned values, namely the divisor of the average ratio and the divisor of the number of blocks for comparison. The triple was tested on two times 200 images with the following values:

$$\begin{aligned} k &\in \{p * 5 \mid p \in [1, 7],\ p \in \mathbf{Z}\}, \\ m &\in \{p * 0.5 \mid p \in [4, 10],\ p \in \mathbf{Z}\}, \\ c &\in \{p \mid p \in [4, 7],\ p \in \mathbf{Z}\}. \end{aligned}$$

Out of 392 tests, the results of the chosen combinations are shown in the following table. The false negative rate determines the ghosty images that were classified as non-ghosty images, while the false positive rate means the non-ghosty images that were classified as ghosty images.

No.	k	m	c	False negatives	False positives
1	15	2.5	5	12%	13%
2	15	3	5	25%	4%
3	10	2.5	5	4%	21.5%

The average accuracy of the first combination is 87.5%, while the second and third combinations were chosen due to the significantly lower false negative and false positive rates, which is only 4% in both cases. The importance of these differences is further discussed in the next section.

4 Results

Based on the test results, the above triples were tested on two times 500 images which were selected manually into either ghost or non-ghost category. The results of the extended collection are the following:

No.	k	m	c	False negatives	False positives
1	15	2.5	5	12.4%	9.6%
2	15	3	5	73.2%	3.4%
3	10	2.5	5	4.8%	19.6%

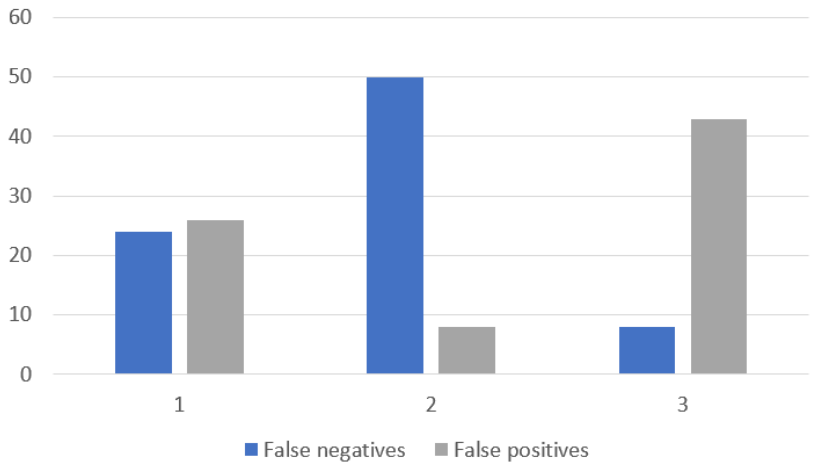

Figure 3.1: Number of false negatives and false positives out of 2*200 images.

It is noticeable that using the first triple, both the number of false negatives and false positives are fairly low, therefore they are preferred to use in a general case. However, if the requirement is to achieve the lowest false negative or false positive rate, the other two combinations are recommended. For instance, if we would like to have the least number of wrongly selected ghosty images, it is preferred to use the second combination.

The tests were run on a notebook with an Intel Core i5-7200U processor and 8GB of RAM using MatLab R2017a. With this setup, the elapsed time during examining 500 images was between 570 and 696 seconds, approximately 9-12 minutes. The difference may occur due to the variety in the size of the images.

Conclusion

During the process of taking a set of fingerprints for registration at a police office, a type of contamination called ghost fingerprints may occur. Ghosts render the print unsuitable for recognition, thus the contaminated images must be eliminated from the database. For this purpose, a ghost fingerprint detection software was developed using MatLab.

Local Binary Patterns have been applied in different scenarios but not yet for detecting ghost fingerprints. The background is first extracted using the convex hull of the fingerprint. The LBP image of the original image is divided into blocks and each block is classified as either contaminated or uncontaminated by utilizing the white - dark pixel ratio of the blocks. Based on the results, the whole image is categorized as ghosty or non-ghosty. This method produced an average accuracy of 89% on 1000 images, more precisely, 12.4% of false negatives and 9.6% of false positives. Two alternatives are proposed as well with either fairly low false positive rate or false negative rate, respectively 3.4% and 4.8%.

The results are not yet perfect, there is still room for improvement. A possible direction is using a different indicator instead of the white - dark pixel ratio, or to modify the decision-making algorithm. Another possibility would be to apply machine learning for detecting ghost fingerprints.

References

[1] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, Dec 2006.

[2] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, July 2002.

[3] Timo Ojala and Matti Pietikäinen. Unsupervised texture segmentation using feature distributions. 32:477–486, 03 1999.

[4] Matti Pietikinen, Abdenour Hadid, Guoying Zhao, and Timo Ahonen. *Computer Vision Using Local Binary Patterns*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[5] M. Topi, O. Timo, P. Matti, and S. Maricor. Robust texture classification by subsets of local binary patterns. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 3, pages 935–938 vol.3, Sept 2000.