SHIV NADAR
INSTITUTION OF EMINENCE DEEMED TO BE UNIVERSITY
DELHI NCR

SCHOOL OF ENGINEERING

EED 305: Digital Signal Processing

Name: Dushyant Singh Satyapal (ds668)
Abhijeet Chahal (ac386)
Prerit Rajpal (pr987)

Group No: 25.

IEEE Paper: Fingerprint Liveness Detection Using Local Coherence Patterns
Volume-24; Year-2017
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7775054&tag=1

Introduction-

With the rapidly increasing demand for payment services working on mobile devices such as smartphones and tablets, biometric-based verification techniques have recently been actively explored. Most notably, the fingerprint has started to be widely applied for this task due to the high-level security without users' inconvenience; however, it still suffers from spoofing attacks based on various materials, e.g., silicone, wood glue, etc. Moreover, an attacker can capture the remaining fingerprints on the stolen device and readily utilize them to penetrate the security system. To detect such spoofing attacks, in the early days, the characteristics of ridges, e.g., ridge clarity, ridge frequency, ridge continuity, etc., have been widely used for describing the difference between live and fake fingerprints. The image quality also has consistently employed helpful valuable features based on the observation that fake fingerprints are apt to yield a low-quality image due to acquisition artifacts such as spots and patches.

However, those are still vulnerable to revealing local differences between live and fake fingerprints. Several researchers have devoted considerable effort to developing diverse local image descriptors for fingerprint liveness detection to address this problem. Most predominantly, the local binary pattern (LBP) has been adopted as one of the state-of-the-art descriptors. Since LBP efficiently encodes the textural information based on the simple relationship between neighbor pixels in a local region, it can accurately represent the difference in the underlying structures between live and fake fingerprints.

This report proposes a novel image descriptor for detecting fingerprint liveness. The key idea of the proposed method is that the difference between live and fake fingerprints is well revealed by the directional coherence in the gradient field.

Objective-

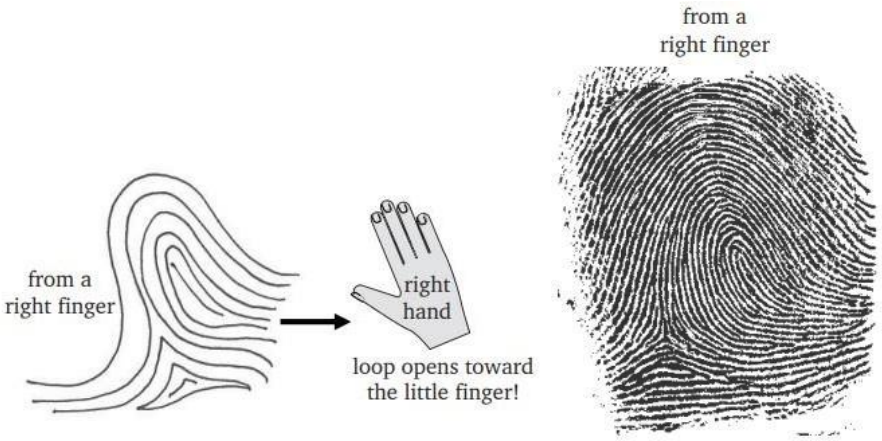
To develop an image descriptor for fingerprint liveness detection using the local coherence pattern of a given image. It would enhance the security of biometric recognition frameworks.

It was found that materials used for making fake fingerprints bring nonuniformity in the captured image due to the replica fabrication process. Thus, we take local coherence patterns to be fed into a linear Support Vector Machine (SVM) classifier to determine whether a given fingerprint is fake.

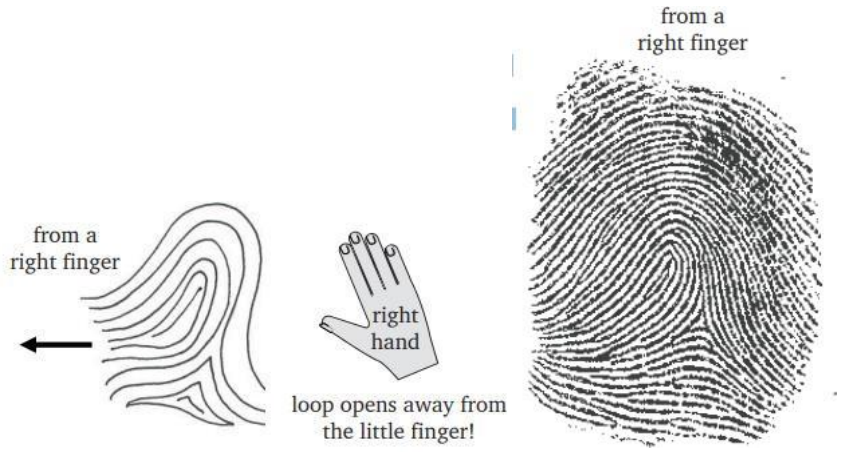
Classification of fingerprint-

- **Loop patterns**- Several ridges enter from one side of the fingerprint, then fold or curve back and exit from the same side, forming a loop. There are two kinds of loops-

    1. **Ulnar Loop**- if the loop opens toward the little finger, it is an ulnar loop. An ulnar loop fingerprint taken from a right finger will open to the right side of the print, and an ulnar loop fingerprint taken from a left finger will open to the left side.



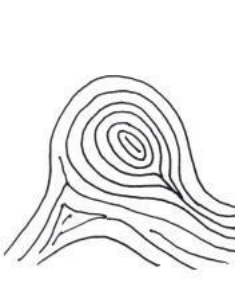
2. **Radial Loop-** if the loop opens away from the little finger (toward the thumb side of the hand), it is a radial loop. An ulnar loop fingerprint taken from a right finger will open the left side of the print, and an ulnar loop fingerprint taken from a left finger will open the right side of the image.



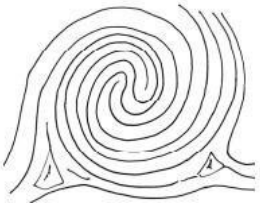
- **Whorl Patterns-** These fingerprint patterns have ridges that form circles or are a combination of two patterns. There are four kinds of whorls.

    1) **Plain Whorl** - ridges form a series of complete circular rings or ovals in the center of the print.

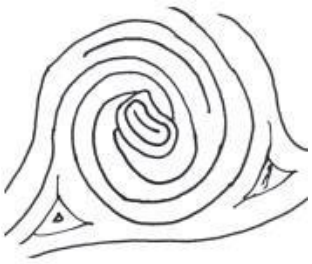
2) **Central Pocket Whorl** - looks like an ulnar or radial loop, but some of the ridges in the center form complete circles.




3) **Double Loop-** ridges form two loops turning around each other.




4) **Accidental Whorl-** ridges form a combination of several patterns. A double loop with a complete circle or two in the center is the most common.

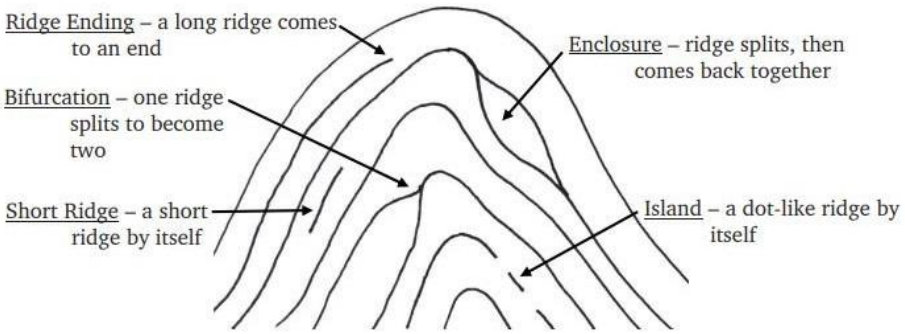



- **Arch Patterns-** All ridges move across the fingerprint from one side to the other side and rise in the center like a wave. There are two kinds of arches.

I. **Plain Arch-** a smooth, slightly raised arch.

II. **Tented Arch-** a spike-like ridge pushes the arch up abruptly (like a tent pole holding up a tent).

- **<u>Special Ridge Characteristics-</u>**



Understanding of the work done in the paper-

The rationale behind our approach is that the nonuniformity of materials occurring in the process of fingerprint fabrication reduces the contrast between ridges and valleys, leading to the gradient distribution dispersion. Since the fingerprint is a directional structure, such dispersion makes the coherence along the dominant orientation.

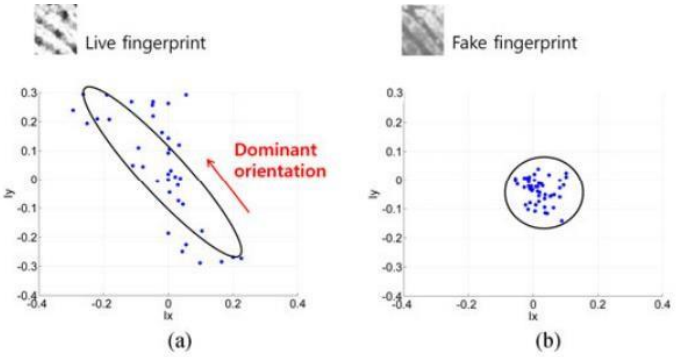


Fig. 1. Gradient distributions obtained from the small local region of live and fake fingerprints (I_x and I_y denote gradients in horizontal and vertical directions, respectively). (a) Result on the live fingerprint. (b) Result on the fake fingerprint. Note that images are from the ATVS dataset [7].

The figure shows that image gradients from the live fingerprint are distributed along the dominant orientation. In contrast, gradients from the fake fingerprint are scattered even though those are extracted from a similar structure. Therefore, local patterns of coherence in the gradient field discriminate live fingerprints from fake ones. We will model this characteristic by allowing for the orthogonal analysis [e.g., singular value decomposition (SVD)] of the intensity-gradient distribution.

Coherence Computation-

As we know, coherence is a fixed relationship between the phase of waves in a beam of radiation of a single frequency. To compute the directional coherence in the fingerprint image, we first calculate the intensity-gradient distribution as follows-

$$C= [c_1, c_2, c_3, \ldots, c_m], c_k=[I(k), I_x(k), I_y(k)]^T$$

Where I(k), I_x(k), and I_y(k) denote the intensity and image gradients of horizontal and vertical directions at each pixel position (x_k,y_k), M represents the number of pixels belonging to the local region. It is worth noting that our combined intensity value plays a vital role in involving the textural information of different materials (e.g., skin, silicone, wood glue, etc.) more efficiently and further projects the gradient distribution into the higher feature space (i.e., three-dimensional space), which brings a significant increase of the discriminative power to determine whether a given fingerprint is fake or not.

To find the dominant orientation and its coherence based on the intensity-gradient distribution, we adopt the Singular Value Decomposition (SVD) because it decomposes the given decomposition into independent axes with the corresponding energy. Therefore, the dominant orientation and its energy in the intensity-gradient distribution can be efficiently estimated by computing the SVD of C as follows:

$$\mathbf{C} = \mathbf{UWV}^T = \mathbf{U} \begin{bmatrix} w_1 & 0 & 0 \\ 0 & w_2 & 0 \\ 0 & 0 & w_3 \\ \vdots & \vdots & \vdots \end{bmatrix} [\mathbf{v}_1 \ \mathbf{v}_2 \ \mathbf{v}_3]^T$$

Where U is an M*M unitary matrix containing energies in directions of v1,v2, and v3.

A unitary matrix is a matrix in which its conjugate transpose is also its inverse.

V is a 3*3 matrix in which the column vector v_1 represents the dominant orientation of the given distribution.

Coherence, $c=w_1-(w_2+w_3)$, where $w_1>=w_2>=w_3$

Here the more significant the value c is, the higher the directional coherence of ridges and valleys is.

Singular decomposition analysis(SVD)

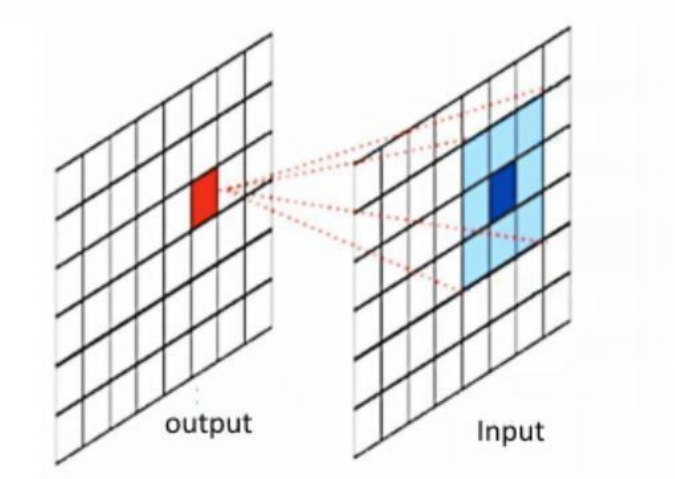
$$C_{m\times n} = U_{m\times r} \times \sum_{r\times r} \times V^1_{r\times n}$$

Local Binary Pattern (LBP)-

The main idea behind LBP is to describe the neighborhood of the image element using binary codes. This method is usually used to study their local properties and identify the characteristics of individual parts of the image.

- **How it works**
  As the name suggests, a Local Binary pattern is a feature of the local representation of an image. It is composed of relative values by comparing each pixel with its neighboring pixels.
  LBP detects microstructures such as edges, lines, spots, and flat areas, which the histogram can estimate.

- **LBP method steps**



1. Convert the image into grayscale space.
2. For each pixel(gp) in the image, select the P neighborhoods surrounding the central pixel. The coordinates of gp are given by-

   *(gc_x-Rsin (2*pi*p/P), gc_y+Rcos(2*pi*p/P))*

3. Take the central pixel (gc) and set it as a threshold for its P neighbors.
4. Set to 1 if the value of the adjacent pixel is greater than or equal to the value of the center pixel,0 otherwise.
5. Now compute the LBP value: Sequentially counterclockwise, write a binary number consisting of digits adjacent to the center pixel. This binary number (or its decimal equivalent) is called LBP-central pixel code and is used as a characteristic selected local texture.

$$LBP(gp_x, gp_y) \sum_{p=0}^{P-1} S(gp - gc) \times 2^p$$

Uniform LBP formula

gc- the intensity value of the central pixel.

gp- the intensity of the neighboring pixel with index p.

the function S can be expressed as:

$$s(x) = \begin{cases} 1 & if \quad x \geq 0 \\ 0 & if \quad x < 0. \end{cases}$$

threshold (step) function

P- number of sampling points on a circle of radius R (circular neighborhood).

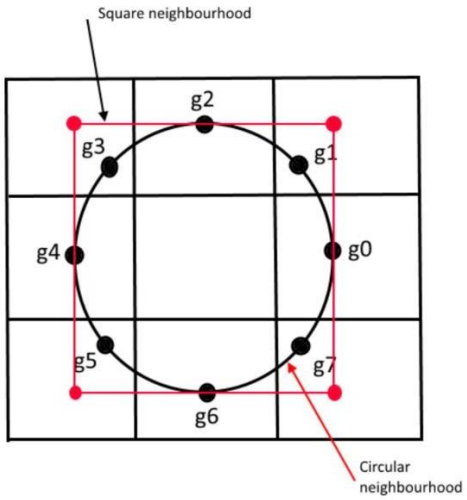
P- controls the quantization of the method.

R- determines the spatial resolution of the method or operator.

Pseudo-code-

To implement this method, we require 3 for loops:

```
For i in height range:
     For j in width range:
          Select a chunck of the image to compute its LBP
value
          For each block neighbors:
              check if interpolation is needed
              Compute_LBP(bock)
              Add the result
          Update the matrix with the value of LBP
```



Local Coherence Pattern-

We define LCP by encoding the relationship between coherences of neighbor pixels as follows.

$$f_{\mathrm{LCP}}(x,y) = \sum_{1 \le i \le n} 2^{i-1} \mathrm{LCP}^i(x,y)$$

$$\mathrm{LCP}^i(x,y) = \begin{cases} 1, & \text{if } c(x,y) > c(x_i,y_i) \\ 0, & \text{otherwise} \end{cases}$$

Where n is the number of sampling pixels in the neighbor region of 3*3 pixels, and i denotes the index of the neighbor pixel centered at the current pixel position (x,y). We can represent f_{LCP} as a gray-scale image based on the above equation. It is worth noting that the local pattern of the coherence obtained from live and fake fingerprints is quite different from each other.

To make the histogram features based on f_{LCP} values, we first filter out the background region by simply checking the standard deviation of intensities in image block B, whose size is set to 16 × 16 pixels in our implementation. To make the histogram features based on f_{LCP} values, we first filter out the background region by simply checking the standard deviation of intensities in image block B, whose size is set to 16 × 16 pixels in our implementation. Note that each block is overlapped by its half size (i.e., 8 pixels) in horizontal and vertical directions, respectively.

We numerically define the LCP histogram h = (h_1 , h_2 ,...,h_{59}) with the L_2 normalization as follows:

$$h_k = \frac{N_k}{\sqrt{\sum_{q=1}^{59} N_q^2 + \delta}}, \quad N_k = \sum_{\tilde{f}_{\mathrm{LCP}}(x,y)\in k} 1$$

Where ˜ F_{LCP} denotes the bin of the uniform patterns generated from F_{LCP} defined in the above equation, δ is the small positive number. Finally, as our baseline, we compute the average of all the LCP histograms constructed in the fingerprint region.

$$\mathbf{F}_{\mathrm{LCP}} = \frac{1}{M}\sum_{k=1}^{M} \mathbf{h}_k$$

where M denotes the number of blocks belonging to the fingerprint region, and FLCP is the feature vector for the given fingerprint image, fed into the linear classifier for training and testing.

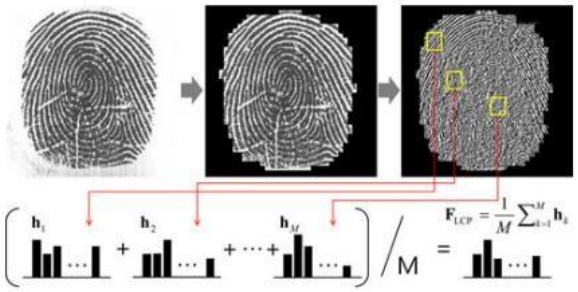


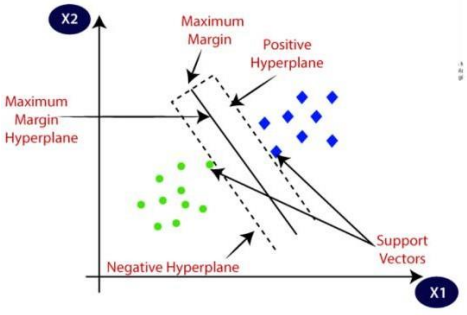
Fig. 3. Overall procedure for our LCP histogram-based feature. Left: input image. Middle: Filtered image based on the standard deviation. Right: LCP image. Bottom: Feature extraction based on our LCP histograms.

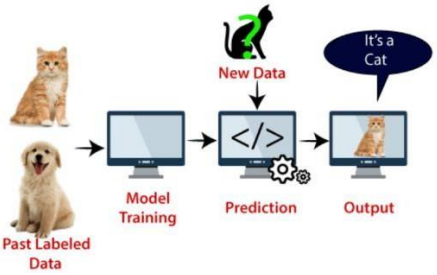
Support Vector Machine (SVM)-

The objective of the support vector machine algorithm is to find a hyperplane in N-dimensional space (N- the number of features) that distinctly classifies the data points.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can quickly put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These severe cases are called support vectors, and hence algorithm is termed a Support Vector Machine. Consider the below diagram in which two different categories are classified using a decision boundary or hyperplane:





Algorithm-

Input: fingerprint image

Step 1: Coherence computation

*Construct C using I, I_x, I_y at every pixel position

*Conduct SVD of C and compute the coherence

Step 2: LCP construction (for fingerprint blocks)

*****Compute the f_{LCP} using the above equation.

* Construct an LCP histogram **h** with L_2-norm

*Compute the average of all the **h** for fingerprint blocks.

$\mathbf{F}_{\mathrm{LCP}} = \frac{1}{M}\sum_{k=1}^{M} \mathbf{h}_k$ (M: # of fingerprint blocks).

Extended Work-

We further extended our work beyond the paper and implemented the Local Ternary Pattern to check the efficiency of the model based on LTP.

LTP-

Local ternary patterns (LTP) are an extension of the local binary patterns (LBP). Unlike LBP, it does not threshold the pixels into 0 and 1; rather, it uses a threshold constant to threshold pixels into three values. Considering k as the threshold constant, c as the value of the center pixel, and a neighboring pixel p, the result of the threshold is:

$$\begin{cases} 1, & \text{if } p > c + k \\ 0, & \text{if } p > c - k \text{ and } p < c + k \\ -1 & \text{if } p < c - k \end{cases}$$

In this way, each thresholded pixel has one of the three values. Neighboring pixels are combined after thresholding into a ternary pattern. Computing a histogram of these ternary values will result in a large range, so the ternary pattern is split into two binary patterns. Histograms are concatenated to generate a descriptor double the size of LBP.

LBPs have been proven to be highly discriminative features for texture classification. They are resistant to lighting effects in the sense that they are invariant to monotonic gray-level transformations. However, because they threshold at exactly the value of the central pixel, they tend to be sensitive to noise, particularly in near-uniform image regions, and to smooth weak illumination gradients. Many facial regions are relatively uniform, and it is legitimate to investigate whether the robustness of the features can be improved in these regions. This section extends LBP to 3-valued codes, LTP, in which gray levels in a zone of width around are quantized to zero, ones above this are quantized to -1, and ones below it to 1, i.e., the indicator is replaced with a 3-valued function and the binary LBP code is replaced by a ternary LTP code. Here k/t is a user-specified threshold—so LTP codes are more resistant to noise but no longer strictly invariant to gray-level transformations.

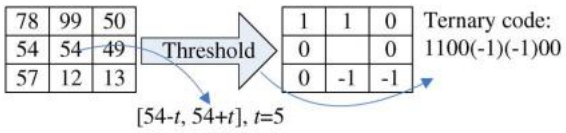


Fig. 7. Illustration of the basic LTP operator.

The LTP encoding procedure is illustrated in Fig. 7. Here, the threshold was set to 5, so the tolerance interval is [49, 59].
When using LTP for visual matching, we could use valued codes, but the uniform pattern argument also applies in the ternary case. For simplicity, the experiments below use a coding scheme that splits each ternary pattern into its positive and negative halves as illustrated in Fig. 8, subsequently treating these as two separate channels of LBP descriptors for which separate histograms and similarity metrics are computed, combining the results only at the end of the computation.

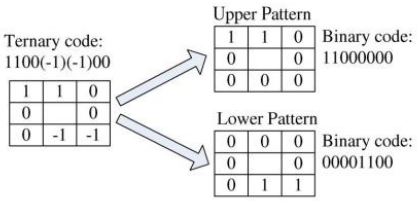


Fig. 8. Splitting an LTP code into positive and negative LBP codes.

Conclusion-

A simple and novel local descriptor for fingerprint liveness detection has been used by us. Our key observation is that the fabrication process mostly makes the dispersion of the directional structures in the fingerprint image (i.e., ridge and valley). Therefore, we propose exploiting the local coherence patterns along the dominant orientation in the intensity-gradient distribution of the given fingerprint image.

Future Work

Currently, we do not have a proper dataset; we are using some of the fingerprint images from the web. We have requested the author and some other fingerprint dataset providers (ATVs, LivDet) to provide the dataset so that we can train our model correctly and make more changes in the code to make the model more efficient. So far, we are done with the base model and have extended the work by implementing LTP; we are further implementing SVM classification, which basically tell us whether the fingerprint image is real or fake. Future work concerns the deeper analysis of mechanisms, new proposals to try different methods, or simply curiosity.

References

https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

https://www.geeksforgeeks.org/singular-value-decomposition-svd/

https://en.wikipedia.org/wiki/Support_vector_machine

https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

https://en.wikipedia.org/wiki/Local_ternary_patterns

https://en.wikipedia.org/wiki/Local_binary_patterns

https://medium.com/swlh/local-binary-pattern-algorithm-the-math-behind-it-%EF%B8%8F-edf7b0e1c8b3

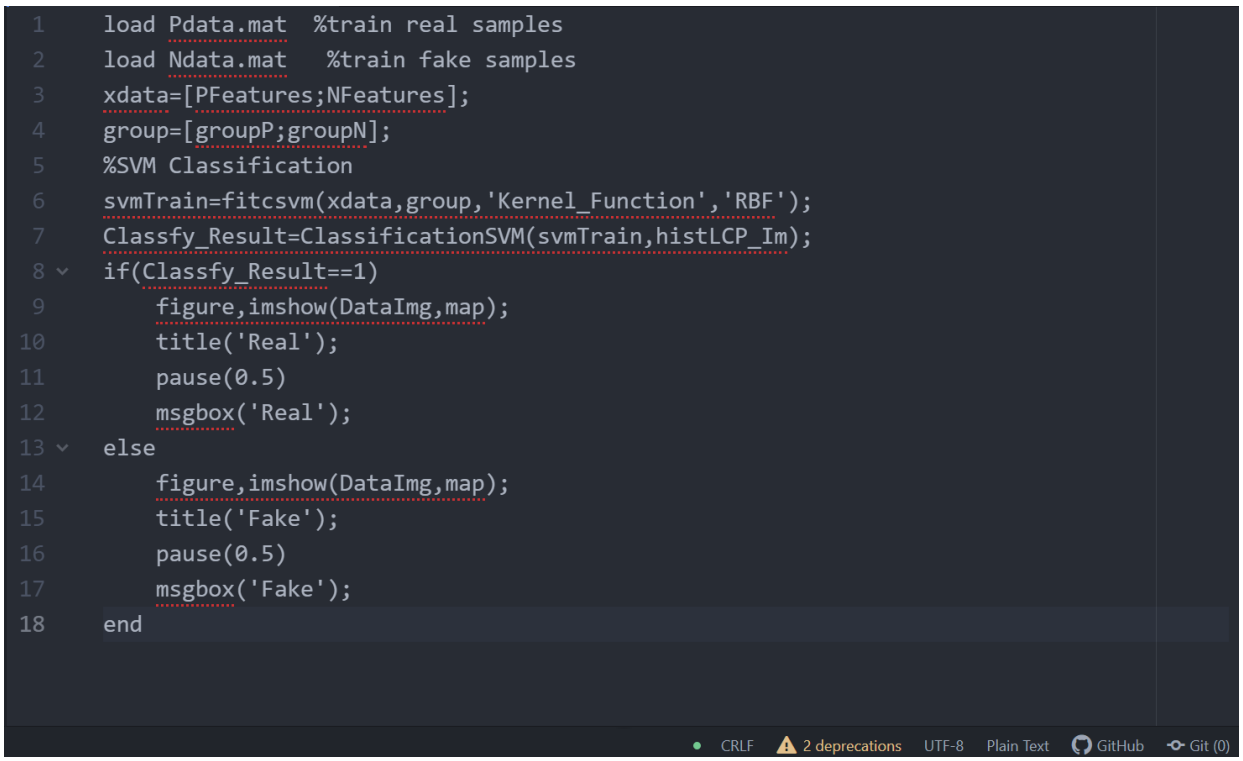
https://wiseai.dev/blogs

https://blog.devgenius.io/face-recognition-based-on-lbph-algorithm-17acd65ca5f7

https://www.geeksforgeeks.org/singular-value-decomposition-svd/

SVM Model-

Below is our attached SVM code snippet-



```
load Pdata.mat  %train real samples
load Ndata.mat   %train fake samples
xdata=[PFeatures;NFeatures];
group=[groupP;groupN];
%SVM Classification
svmTrain=fitcsvm(xdata,group,'Kernel_Function','RBF');
Classfy_Result=ClassificationSVM(svmTrain,histLCP_Im);
if(Classfy_Result==1)
    figure,imshow(DataImg,map);
    title('Real');
    pause(0.5)
    msgbox('Real');
else
    figure,imshow(DataImg,map);
    title('Fake');
    pause(0.5)
    msgbox('Fake');
end
```