Department of Electronics and
Electrical Communications Engineering
Faculty of Engineering - Cairo University

Fake currency recognition using Edge Detection

By "Heptacone" team

Presented by

Name	Sec.	ID.
شهد محمد عطيه عبدالغفار	2	9220406
حبيبه حاتم محمد العطار	2	9220254
سهيله احمد محمد عبدالمجيد	2	9220377
مؤمن وائل سيد عباس	3	9220616
مؤيد احمد عبدالفتاح	3	9220617
مايكل هاني سمير جرجس	3	9220637
يوسف محمود سعيد البابلي	4	9221018

Submitted to: Dr. Samah El-Tantawy

2023

Abstract

Human curiosity is endless, and this curiosity made him make use of linear algebra and other mathematical topics to create edge detection technology. Edge detection is a fundamental tool in image processing, machine vision, and computer vision, it is used to be specific-in feature detection and feature extraction. In our project, we are keen on showing the history of edge detection and its different applications. We will discuss fake currency as an application in which edge detection is used. We will discuss how linear algebra takes part in it. We will also display some algorithms used in it and how some parameters affect it.

Table of contents

List of figures

List of graphs

List of equations

• *Introduction*

Edge detection depends on locating edges and sharp changes in the image. Edges and sharp changes refer to points where light and colors change. Light and color changes reveal the sharp lines of the processed image. These are precisely the places we call edges here: the points of changing light and brightness on the digital image. A digital image becomes an edge in multiple and different ways, like: Horizontal Edges, Vertical Edges, and Diagonal Edges.

Detecting the edges of the image allows the observer to examine the light and color changes in the image in detail. It also means that the algorithm, i.e., machine vision and computer vision, can better process and understand the image.

There are many ways to detect the presence of edges, including Prewitt edge detection, Sobel edge detection, Laplace edge detection, and Canny edge detection.

edge detection has many advantages to choose it like, we can use it in the medical field as it is used to detect abnormalities in the human body for example detection of tumors. Also, we can use it in many applications like fingerprint recognition, vehicle detection, satellite images, and robotic vision.

- ***Literature Review***

1. A study and comparison about different edge detection techniques:

Edge detection is defined as the process of locating the sharp discontinuities in pixel intensity or can be characterized as the boundary, therefore the study of various edge-detecting techniques is of great importance for many applications in modern technologies.

There are many ways to perform edge detection, but most methods can be categorized into two main groups. The gradient-based edge detection uses the gradient method which detects edges by looking for the maximum and minimum in the first derivative, and the Laplacian-based edge detection which searches for zero crossings in the second derivative of the image to find edges.

This study focused on 4 main edge detection techniques 3 of which are gradient-based edge detectors, **the Sobel operator** which uses 3x3 convolution kernels that are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid and produce separate measurements of the gradient component in each direction (**Gx, Gy**) and then be combined to find the absolute magnitude of the gradient at each point, **Prewitt's operator** which is similar to Sobel operator and is used for detecting vertical and horizontal edges, and **Robert's cross operator** which is a 2x2 convolution kernel as well which detects cross edges.

And the **Canny edge detection algorithm** which is known to be one of the more efficient techniques in edge detection, is the canny edge detection built on previous edge detection techniques like Sobel and enhanced it.

In conclusion, the gradient-based algorithms had a major drawback in of its over-sensitivity towards noise while the Canny method displayed better performances in almost every scenario but displayed some drawbacks in the fact that it is much more computationally expensive compared to the other techniques.[1]

- Visual Comparison of various edge detection Algorithms in figure[1],[2]:

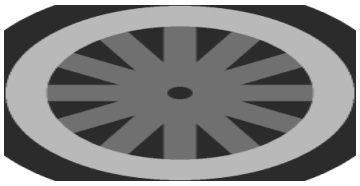


Figure 1: Image used for edge detection analysis

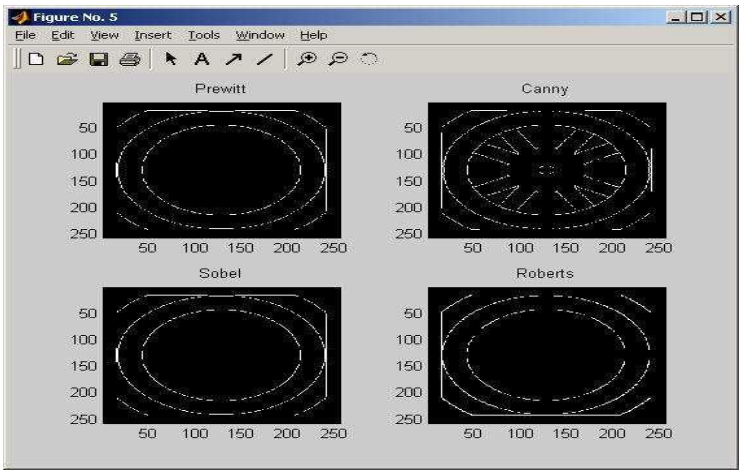


Figure 2: Results of edge detection

2. Overview of Edge Detection Algorithms Based on Deep Learning:

Edge detection is a method that plays an important role in image analysis. How to extract the edge of an image quickly and accurately has always been a research concern. Many existing research and studies have shown that edge detection is of great significance in many fields such as image feature extraction, feature description, and target recognition.

Depending on positioning and extraction, researchers have put a variety of edge detection methods through time. If we look at the time sequence of research progress, the methods can be divided into two main categories: classic traditional algorithms and algorithms based on deep learning as Figure [3] shows.

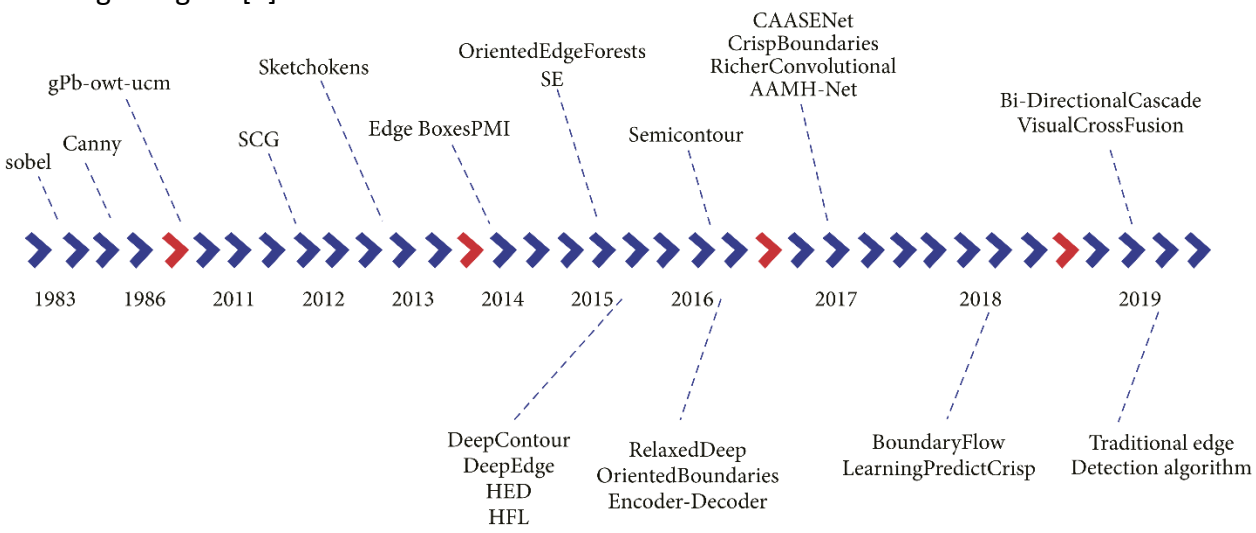


Figure 3: Edge Detection Algorithms Based on Deep Learning

Classic edge detection methods (**Roberts**, **Prewitt**, **Sobel**, and **Canny** operator) often use low-level features (such as color, brightness, texture, and gradient) as the priority of edge detection and depend on both first and second-order edge detection.

Although the classic edge detection methods have made great progress, they have many limitations. With the development of deep learning technology, and as edge extraction becomes more complex, more modern methods were needed. The most important difference between deep learning and traditional edge detection methods is that the features it uses are automatically learned from big data instead of manual design.

Deep learning edge detection algorithms are divided into two main algorithms:

1. Fully supervised learning edge detection algorithms.
2. Weakly supervised learning edge detection algorithms.

➢ Fully Supervised Learning Edge Detection Algorithm

Fully supervised learning is to use samples of known categories, adjusts the parameters of the classifier, and trains to obtain an optimal model to achieve the required performance. Then use this trained model to assign all inputs to corresponding outputs and review the outputs to achieve the purpose of edge detection. At present, most edge detection algorithms are implemented using full supervision.

This paper divides this algorithm into five categories: spectral clustering, multiscale fusion, network reconstruction, codec-based, and subpixel convolution edge detection algorithm. Methods based on spectral clustering and subpixel have high detection accuracy, but poor anti-noise performance, methods based on neural networks, and codec are the opposite as they solve the anti-noise performance but have low detection accuracy.

➢ Weakly Supervised and Unsupervised Learning Edge Detection Algorithm

Weakly supervised target detection refers to the process that the detection results only cover a small part of the target object, and the target detection task can be achieved through the class label of the sample. Unsupervised learning means that the data learned by the model has no labels. So, the goal of unsupervised learning is to understand the characteristics and laws of the data through these samples.[2]

3. Fingerprint recognition:

Fingerprint has been used widely now, such as authentication for mobile phones and to monitor working hours. But the recognition of the that was used was low, so it has been noticed that the edge detection technique applied to the fingerprint image can enhance the quality of the image. Knowing that we study the four edge detection techniques: Sobel, Prewitt, Robert, and Canny. And for faster classification, we also apply two dimensionality reduction techniques: principal component analysis and linear discriminant analysis. Then we identify fingerprint images with the algorithm support vector machine using a linear kernel function.

Experimental results showed that the pre-processing of fingerprint images using canny edge detection with principal component analysis can increase the recognition rate from 64.3% to 88%. Using canny edge detection with linear discriminant analysis improved fingerprint image recognition from 73.8% to 88%.[3]

Figure 4: Input image

Figure 5: reprocessed image

4. Automated new license plate recognition in Egypt:

The traffic transportation system in Egypt has been widely improved over the last years due to using new techniques in controlling traffic, especially in highly concentrated cities such as Cairo, Giza, Alexandria…etc. Automating license plate recognition is one of the techniques that enhances control over Egyptian transportation.

License plate recognition (**LPR**) was an effective form of Automatic Vehicle Identification (**AVI**) system. **Edge detection** technology, especially the **Sobel technique**, played an important role in this technology as it was used to extract the rectangular plate out of the vehicle. The process is divided into several steps. Starting from extraction then recognition and ending with database communication.

The extraction step has several sub-steps that include image capturing and processing the image using edge detection, filtering, and segmentation. After the image is captured, it's converted to a grey-scale image, and edge detection is applied by using a Sobel edge detector. Morphological algorithms, dilation, and erosion are used. In dilation, every background pixel that is touching an object pixel is changed into an object pixel. Dilation makes the objects larger, and erosion is averse to dilation. Dilation was used in vertical and horizontal lines to detect the structure of the rectangle. Filtering and smoothing eroded images are the next step in extraction to remove unwanted objects. The last step in extraction is the rectangle check test to make sure that the selection process for the plate was correct with the right dimensions.

After plate extraction comes to the segmentation to be able to recognize the plate data's main components of such as the upper region color of the plate, and plate numbers and letters. Lastly comes the database communication step for the data extracted and having all the vehicles driving records instantly and automatically.[4]

Figure 6: The capture of original image

Figure 7: The grayscale image

(a) (b)

Figure 8: (a) Sobel edge detection, (b) dilation

- ***Mathematical Formulation of the problem and Solution Methodology***

➢ System Overview:

Previously we talked about what others have done using the different edge detection methods. Now we are going to talk about our application using edge detection which is **Fake currency recognition**. The counterfeit currency is a severe problem which a lot of people suffer from its consequences because having it is illegal, and its penalty can reach 25 years according to the Egyptian Law No.202 from Penal Code. So, one of the best ways to face this problem is edge detection, edges in an image represent the regions that have the changes in the pixel values.

In practical we will use **Canny edge detection operator** along with others image processing techniques to get more accurate and fast results like gray scale conversion, segmentation, extraction, etc. All these functions will be done from MATLAB or any compiler from different coding languages like Python or C++. We will have some main steps in our system which are :

1. Input and read an image that contains the currency. It could be from scanner or camera and the format could be in BMP or JPEG.
2. Perform a pre-processing method like gray scale conversion , smoothening image, etc. then perform the main image processing methods like edge detection, segmentation, characteristic extraction and then compare the image with the real one using pattern matching. All of that is done by MATLAB, Python or C++.
3. Print the result.

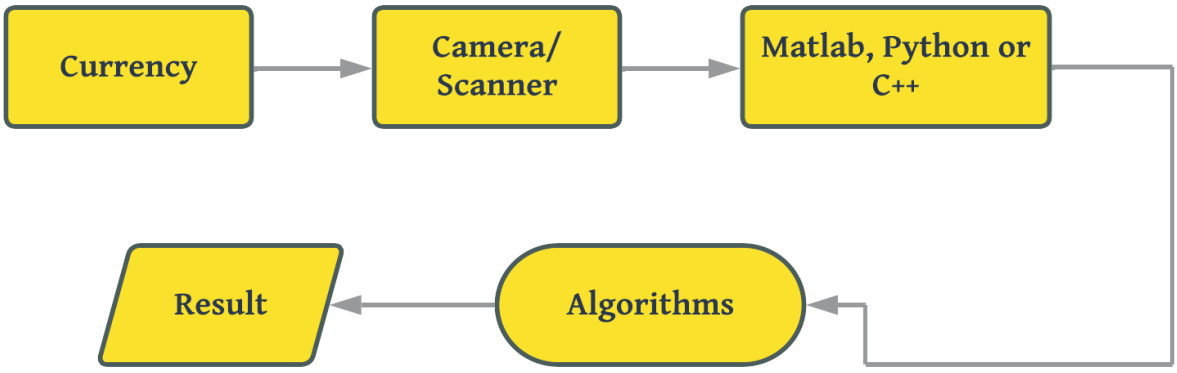


Figure 9: Flow of process edge detection

➢ *Algorithm*

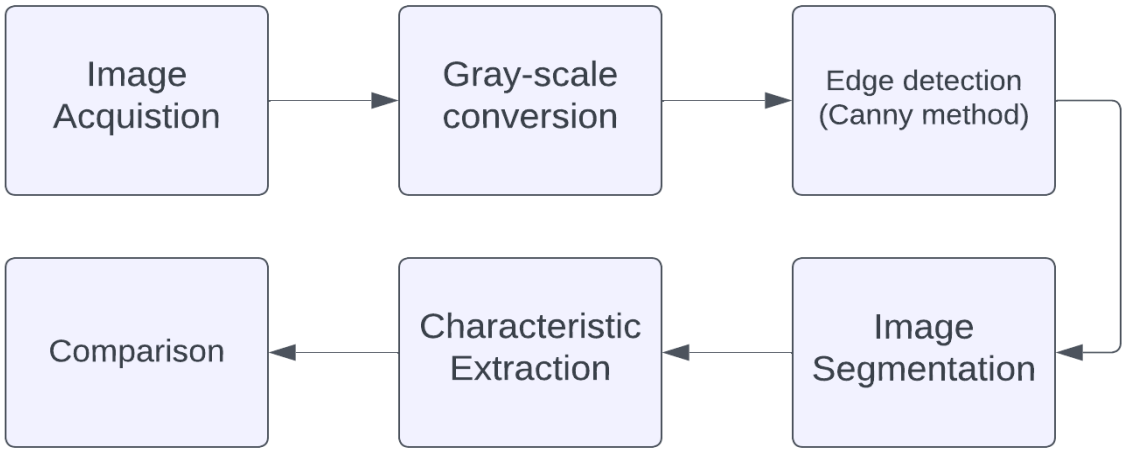


Figure 10: Algorithm for fake currency

1. Image Acquisition:

After scanning the image from scanner or getting it from a camera, it is formatted by any supported format in Python, we will use two input images at which the first image is the original image used as reference currency note, the second is for testing that will be verified by the system. The application can be developed under MATLAB or any coding language like Python or C++. But we used Python for its simplicity.

2. Gray-scale Conversion:

Now we want to turn the acquired image into grayscale. It is a process to convert a colored image (mainly from RGB) into a gray image, which has only black and white color, based on the pixels' values.

Figure 11: Gray-scale image

We apply gray scale as it has many advantages (e.g., it reduces the complexity of the code, it saves some space within the memory, and it only carries the intensity information which make it easier to process instead of three-color components Red(R), Green(G), Blue(B))

3. Edge Detection:

Then we need to extract the characteristics of an image like its edges or boundaries. Edges represent sharp changes in pixel intensity, which can indicate the presence of object boundaries, edges, or texture variations. Edge detection algorithms typically scan an image and process each pixel in relation to its neighboring pixels, looking for areas where the intensity changes sharply. The output of an edge detection process is a binary image of the edges detected, which can be used for further analysis or processing.

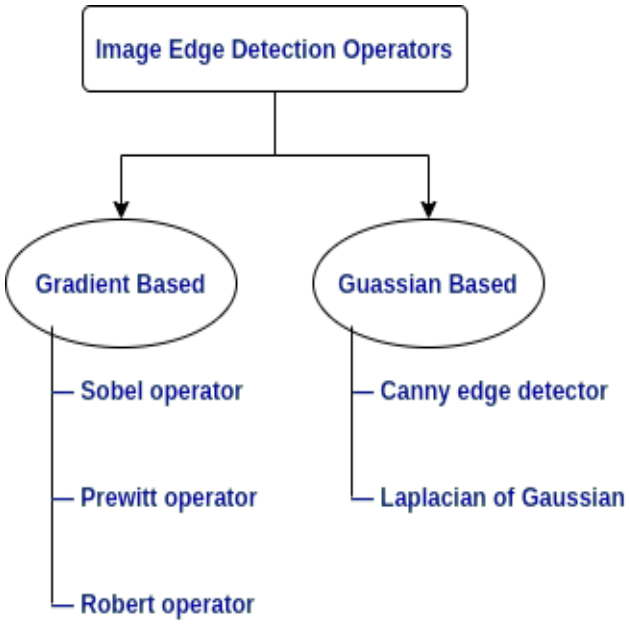


Figure 12: Types of edge detection operators

There are a lot of edge detection algorithms but mainly they are two types: Gaussian-based & Gradient-based. The main difference between them is that Gaussian-based edge detection involves smoothing the image with a Gaussian filter and then taking the gradient to detect edges. This produces smoother edges, but it may not be a highly effective method in noisy images. Gradient-based edge detection calculates the gradient directly from the image, resulting in sharper edges, but it can be more sensitive to noise. The choice between the two techniques depends on the application's requirements, with Gaussian-based edge detection being better for smooth edges and gradient-based edge detection being better for sharp edges.

In our case we will use **the Canny edge detection method** as it has the best result related to our application **Fake currency recognition.**

❖ Canny Method

Introduction:

The canny method is considered an improved method from the Sobel method. It takes the Sobel image and applies more enhancement and treatment to make more clear and Sharper edges.

How it works:

1- Grayscale conversion: Converting the original image into black-and-white.

2- Noise reduction (Image smoothening):

Applying a Gaussian filter to reduce noise from the photo to make sure that each pixel is similar to its neighboring pixels. Gaussian kernel filter of size (3x3, 5x5, 7x7…….) is applied to the photo, and the smaller the size of the filter the less blurry it becomes. The following (equation 1) represents the Gaussian kernel filter applied to the pixels of the photo

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{x^2+y^2}{2\sigma^2}}$$ (Equation 1)

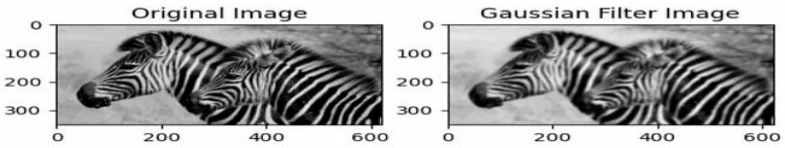


Figure 13: Noise reduction

3-Finding Intensity Gradient:

The gradients could be founded from the derivative or the change in intensity between neighboring pixels of the vertical gradient Sobel-Y (Gy) and the horizontal gradient Sobel-X (Gx) and then we can represent this gradient as a vector that has a magnitude as represented in (equation 2) and direction(orientation) as represented in (equation 3)that we can calculate for every pixel. This method removed all the unwanted pixels and only Left the edges.

(s_x, s_y) Gradient Vector :

Magnitude = $\sqrt{s_x^2 + s_y^2}$ (Equation 2) Direction = $\theta = \tan^{-1}\frac{s_y}{s_x}$ (Equation 3)

The following figure shows the x- gradient, y-gradient, and magnitude gradient which shows how the magnitude gradient showed the edges of the photo

Figure 14: x-gradient, y-gradient, and magnitude gradient images

4-Non-maximum Suppression:

The resulting edges are thick, so we will use this technique to make the edges one pixel thick. We simply neglect the edge points that do not clearly contribute to the visibility of the edge. Using the gradient direction(angle) calculated in the previous step (which is perpendicular to the edge) we compare all pixels along this direction and find the maximum gradient and suppress all other pixels to zero as represented in (equation 4)

$$M(x,y) = \begin{cases} |\nabla S|(x,y) & if\ |\nabla S|(x,y) > |\Delta \dot{S}|(\dot{x},\dot{y}) \& |\Delta S|(x,y) > |\Delta \ddot{S}|(\ddot{x},\ddot{y}) \\ 0 & otherwise \end{cases} \quad \text{(Equation 4)}$$

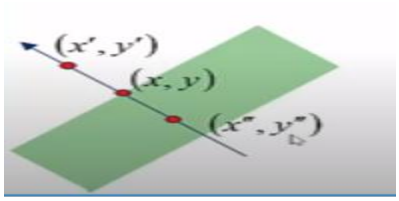


Figure 15: Non-Maximum Suppression

5-Hysteresis threshold:

We finally want to make sure that we removed all the weak response edges(noises) and leave only dominant edges, so we set out two thresholds upper and lower. When the pixel is above the high threshold it is relevant and is named an “edge pixel,” while if it is below the lower threshold, it is not relevant and is named a “non-edge pixel” and is set to 0. There is a case where the pixel is in between the two thresholds and in this case if the pixel is connected to the edge pixels, then it is relevant and will be considered as part of the edge detected object and named edge pixel. If it is not connected, then it will not be relevant.

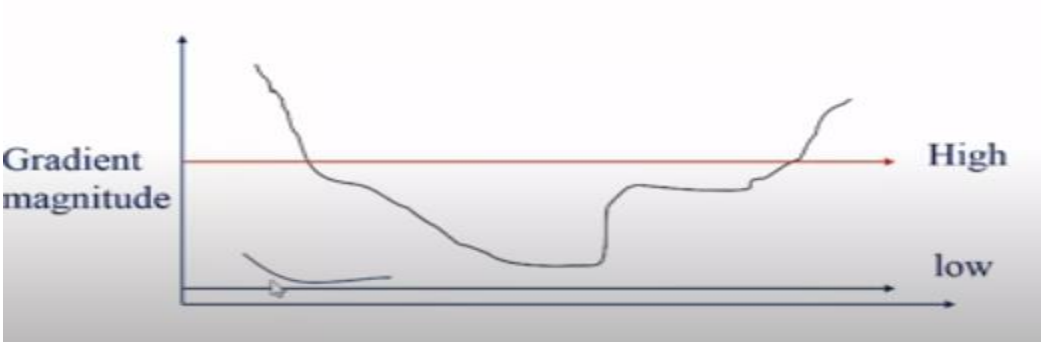


Figure 16: the idea of hysteresis thresholding

After these five steps, we can have a canny edge-detected photo that can be used in many applications such as fake currency detection that will be used to evaluate our methodology.

4. Image Segmentation:

In this process, the image is divided into a set of pixels also known as super pixels, where each pixel is classified into a part. These parts are known as segments. We take the important segments to be processed instead of processing the entire image.

5. Characteristic Extraction:

We extract the features of the image (e.g., security circles, see through register, intaglio microtext, continuous intaglio lines, latent image, optical variable magnetic ink (**OVMI**)[5]) by the edge-based image segmentation.

6. Comparison:

We can use **calculation of intensity** to detect if it is real or fake currency. We calculate the intensity of each extracted feature. If the calculated intensity is greater than the threshold of 70%, then it is classified as original note. Otherwise, it is considered a fake one.

Another common approach is to use computer vision algorithms like **HOG distance** to compare the features of the detected edges in the binary image with those of the reference image. This can include comparing the shape, size, and position of the edges, as well as their texture and orientation.

Also, we can use machine learning techniques to classify the image as genuine or counterfeit. This involves training a machine learning model on a dataset of genuine and counterfeit currency images, and then using this model to classify new images as either genuine or counterfeit based on the features extracted from the Canny edge detection.

● *Experimental Work: Results and Analysis*

➢ Measures with comparative assessment

We will use two measures to detect the fake currency:

1. Intensity difference:

In this measure using our python program with the OpenCV library, we will calculate the mean intensity of the Canny edge-detection for the real currency image and the fake one and compare them with the threshold. The threshold is a value we assign to the code which we try to adjust to get the best accuracy of detecting the fake currency . If the real mean value minus the test mean value is less than the threshold, then the test image is real, otherwise the test image is fake.

2. Euclidean distance using HOG feature:

In this measure, we will calculate the Euclidean distance which can be used as a measure of the dissimilarity between two image features, where the image features are represented as vectors as shown in figure 19. The Euclidean distance between the two vectors represents the magnitude of the difference between the two features, where a smaller value means the features are more similar and a larger value means the features are more dissimilar. And there is also a threshold for it we try to adjust to get the best accuracy of detecting the fake currency.

If the HOD Euclidean distance is less than the threshold, then the test image is real, otherwise the test image is fake.

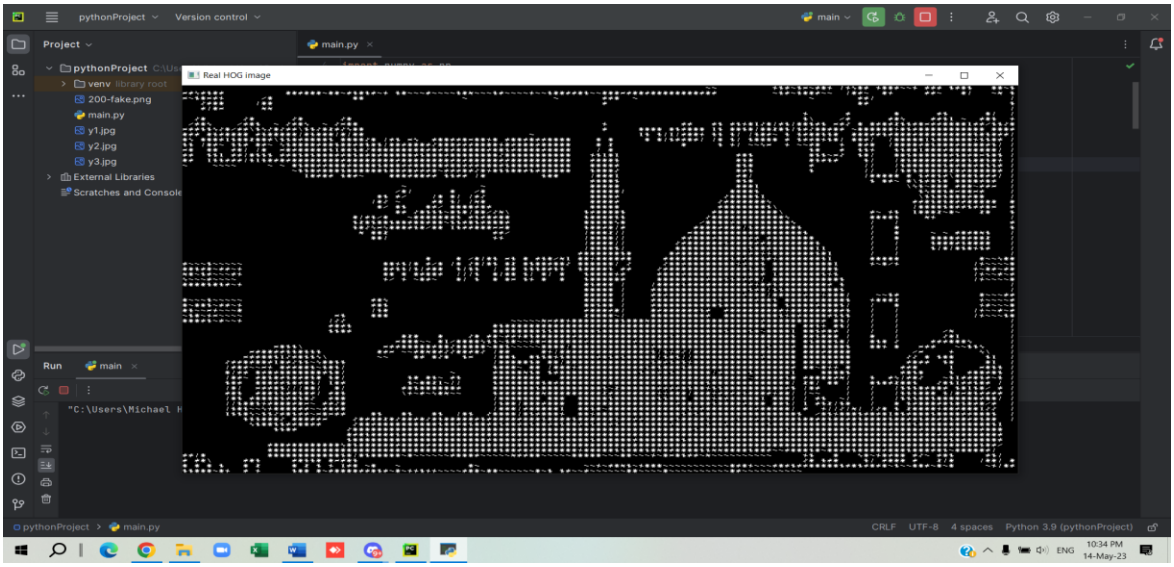


Figure 17: HOG feature extraction

After applying our methodology as explained in the pervious topic and by using the previously discussed 2 measurements our output will be as following :

- Displaying real & test image without any image processing

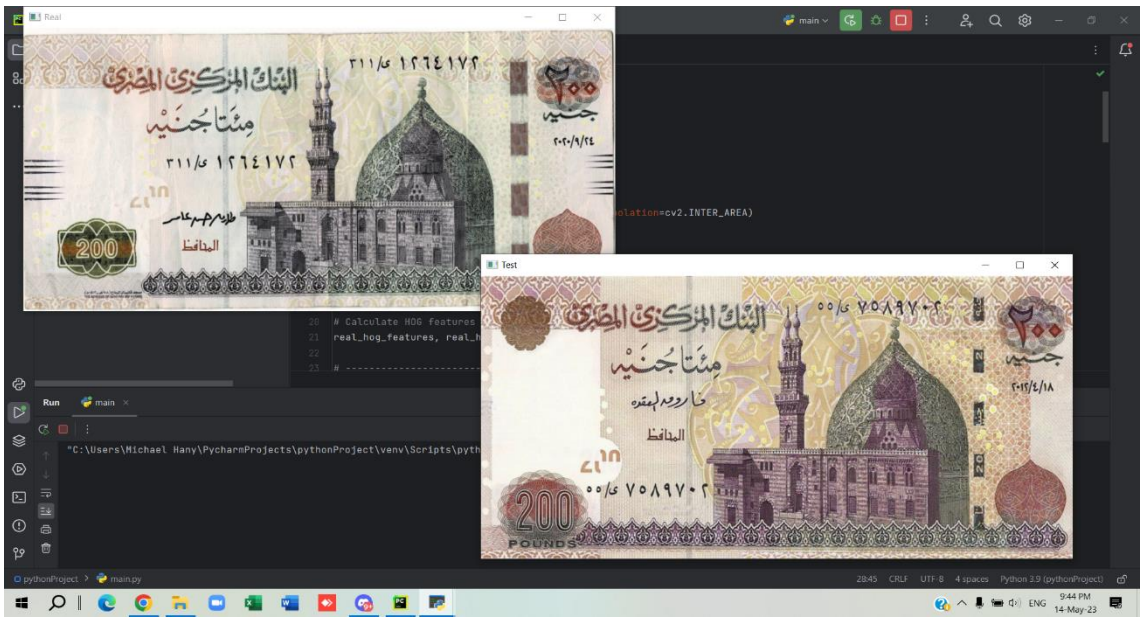


Figure 18: Real & Test image without any image processing

- Displaying a grayscale and blur version of the real & test image

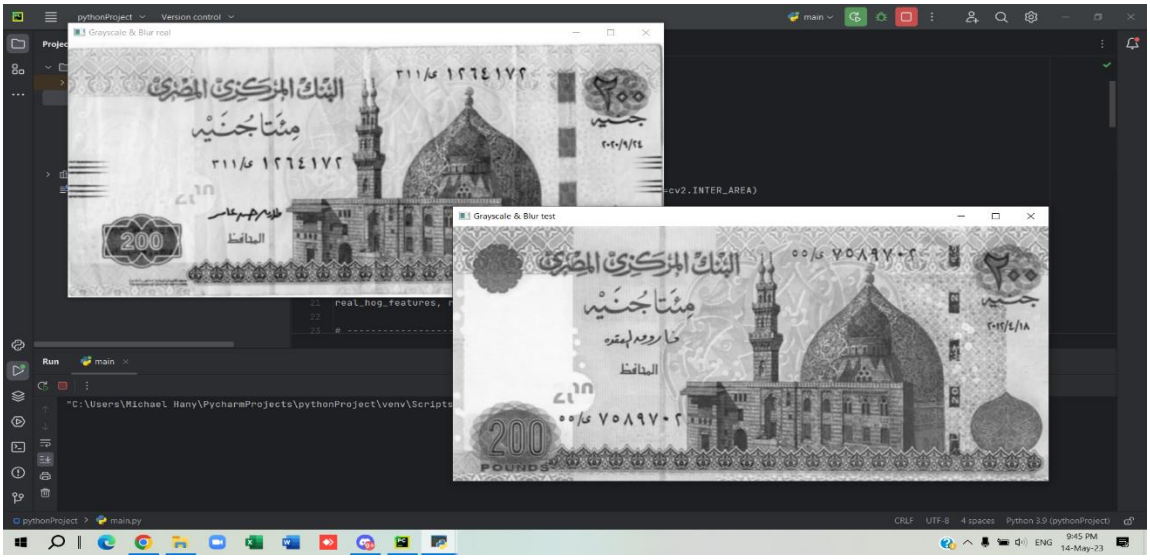


Figure 19: Displaying a grayscale and blur version of the real & test image

- Displaying the Canny edge-detected version of the real & test image

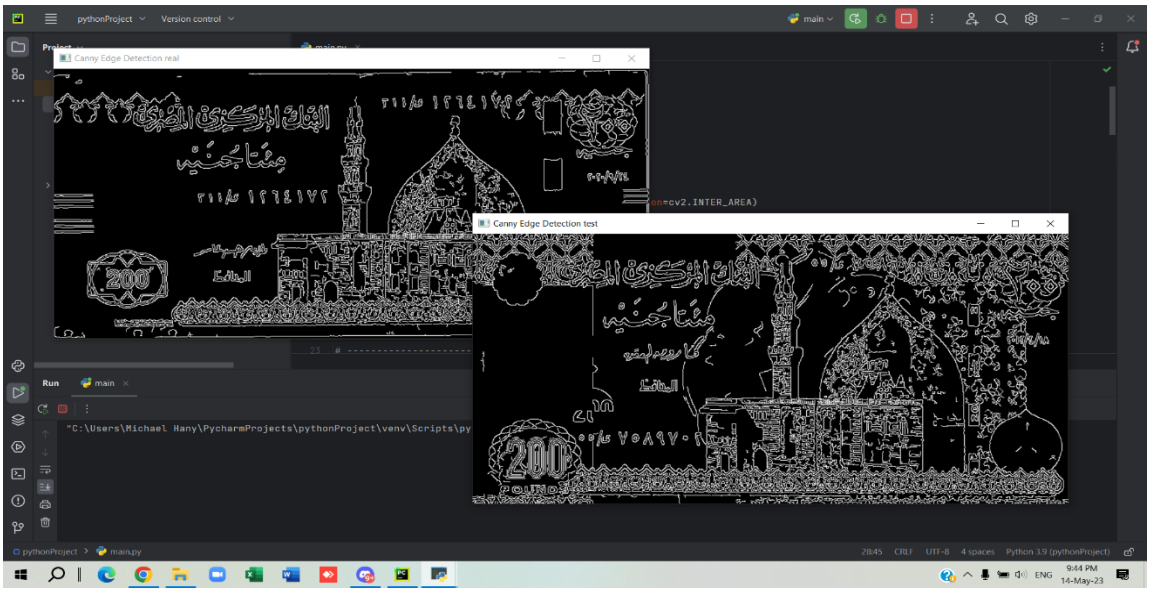


Figure 20: Canny edge-detected version of the real & test image

- Displaying the HOG feature technique to extract the real & test image features

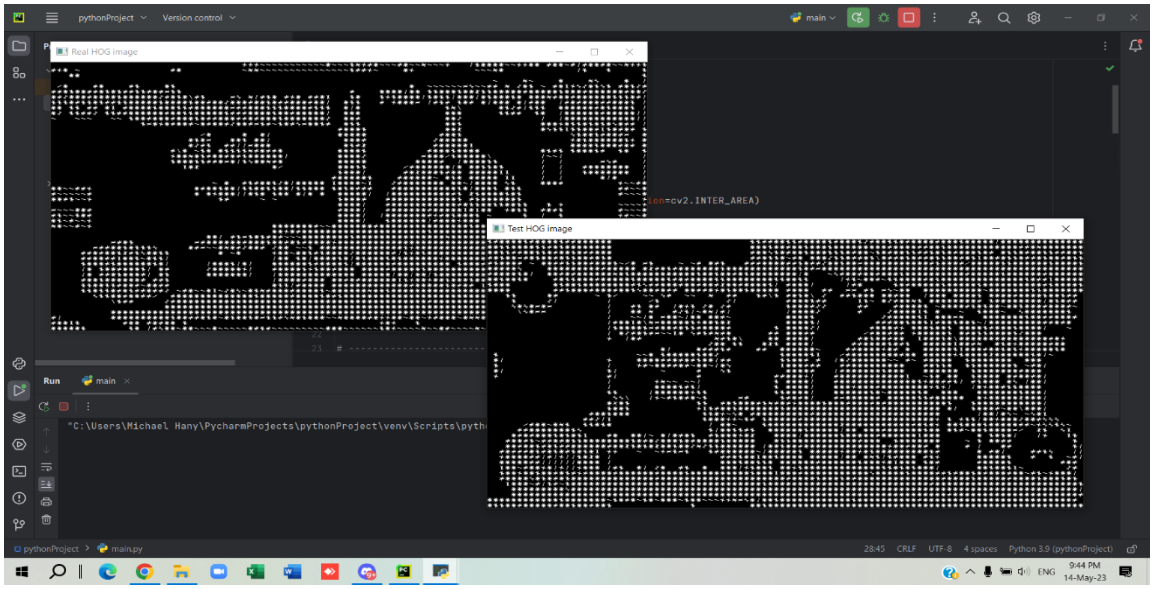


Figure 21: The HOG feature technique to extract the real & test image features

- Print if the test image is a real currency or fake based on the 2 measurements (Intensity difference & Euclidean distance)

```
The test currency is fake based on intensity difference.
The test currency is fake based on HOG distance.
```

➢ Scenarios and comparative assessment:

Our program is designed with python to detect the fake currency using canny edge detection and feature extraction. The main objective of the program is to evaluate how far can the edge detection and feature extraction technology can be applied in such an application and what are the obstructions that we might face that our program might fail against. The test will be applied as follows:

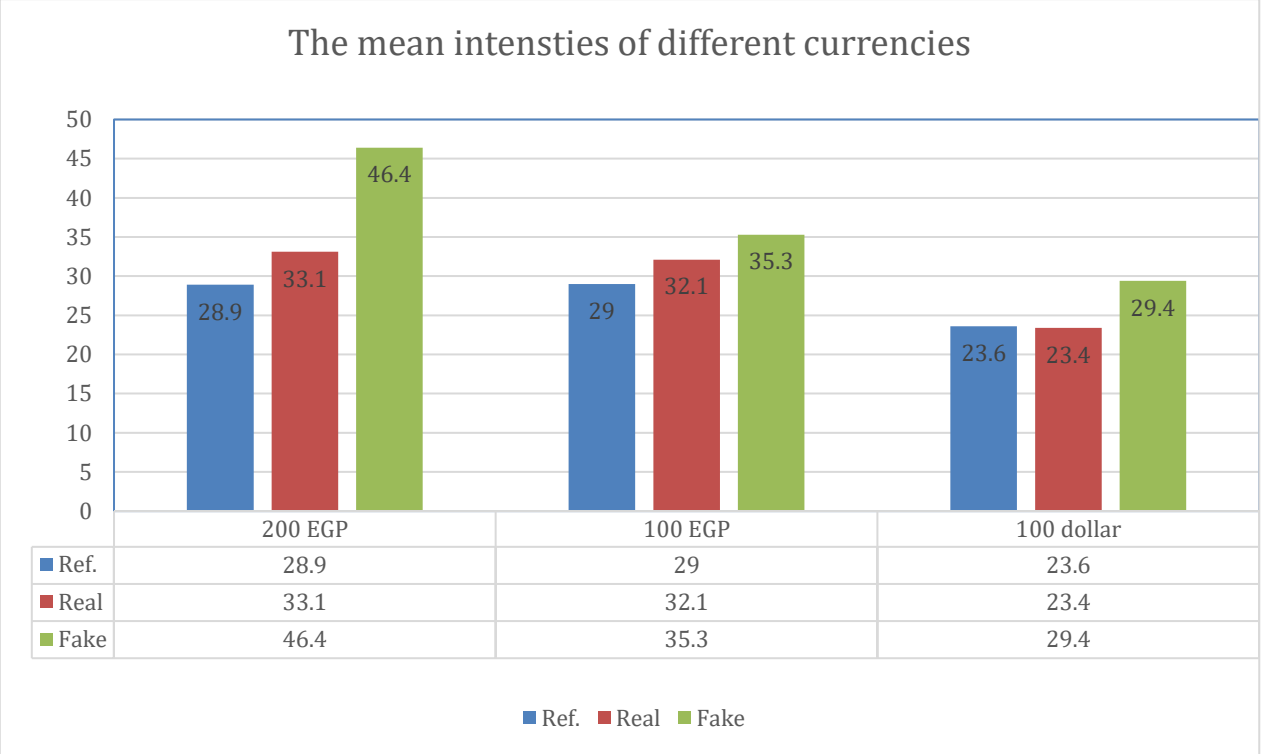
1-Our data set is composed of three copies for each currency; two are real but pictured from different sources and we will consider one of them as reference, and one is fake. Each of the two photos (real and fake) will be compared with the reference.

2- The program will run and give three outputs:

- Whether the currency is real or fake.
- Intensity difference (measure 1)
- HOG difference (measure 2)

The comparative assessment will be based on the results of the two measures and the expected result of the currency (real or fake) based on the input images. We are expecting that the intensity difference and HOG difference of the real image with respect to the reference image are less than that of the fake image with respect to the reference image. The larger the data set the better we can test our program on more conditions.

➢ Data representation

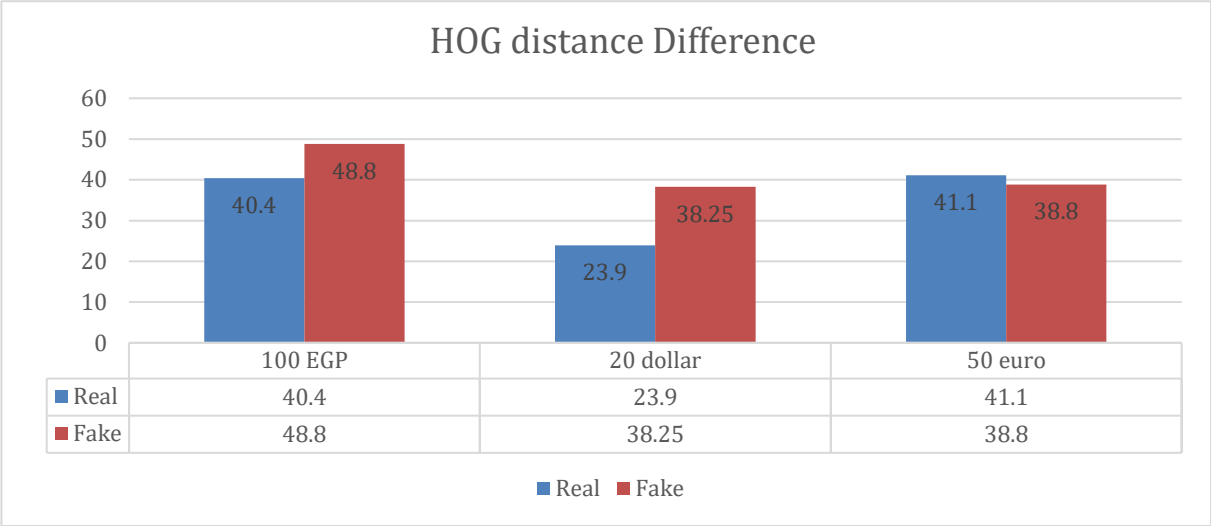


Graph 1: the mean intensities of different currencies

The above figure shows the different mean intensities after applying the canny edge detection computed for 3 different images, 2 of them are real notes and we took one of them as a reference, and 1 fake note.

Our observations are:

- The real and reference notes are very close in their mean intensities while the fake is not as close.
- Different currencies have different average mean intensities.
- The threshold for each currency is different



Graph 2: HOG distance difference

This graph shows the difference between HOG distances of different currencies (how dissimilar are the notes).

We used a reference real note for every different currency and calculated the hog distance between that currency and another real note and a fake note.

Observations:

- The hog distances are relatively big because the chosen cells to compare between pixels are small. That makes high accuracy comparison though the quality of the images isn't perfect since we got them from the web.
- The 50 euro is wrong as we tested two 50-euro notes with lower quality to see the quality impact on the results.

Note: After experimenting on different images of different notes we noticed that the quality of the image greatly affects the mean intensity value and the HOG difference which then affects the final result.

• *Conclusion*

✓ In order to detect the fake currency, we used two methods:

1. Intensity difference: in this method, we assign the threshold value to the code. If the real mean value minus the test mean value is less than the threshold, then the test image is real, otherwise, it is fake.

2. Euclidean distance using HOG feature: we will calculate the Euclidean distance which can be used as a measure of the dissimilarity between two image features. Images features are represented as features. The Euclidean distance between the two vectors represents the magnitude of the difference between the two features. A smaller value means similar features.

Our data set is composed of three copies for each currency: two are real but pictured from different sources and we will consider one of them as a reference, and one is fake. Each of the two photos (real and fake) will be compared with the reference. We are expecting that the intensity difference and HOG difference of the real image with respect to the reference image is less than that of the fake image with respect to the reference image.

✓ The results showed that:

1. Intensity difference: The real and reference notes are very close in their mean intensities while the fake is not as close. The threshold for each currency is different.

2. Euclidean distance using HOG feature: The hog distances are relatively big because the chosen cells to compare between pixels are small. That makes high accuracy comparison though the quality of the images isn't perfect.

✓ Recommendation

Finally, we recommend using the Euclidean distance method since it showed more accurate results and because the intensity difference method is more sensitive to the quality of images which affects the results.

• *Future work*

In future work, there are several potential enhancements to explore for the project on fake currency recognition using Canny edge detection. These include acquiring large datasets, integrating machine learning techniques to complement edge detection, analyzing specific counterfeit features, optimizing for real-time implementation, and evaluating performance in challenging scenarios. These advancements would contribute to more robust and effective counterfeit currency detection.

• *References*

[1] Maini, R., & Aggarwal, H. (2009, February). "Study and comparison of various image edge detection techniques.". *International journal of image processing (IJIP). [On-line]. 3(1)*, pp. 1–11. Available: https://www.academia.edu/download/31159717/Maini.pdf [March 23, 2023].

[2] Tian, B., & Wei, W. (2022, September). "Research Overview on Edge Detection Algorithms Based on Deep Learning and Image Fusion.". *Security and Communication Networks 2022. [On-line]. 2022*. Available: https://www.hindawi.com/journals/scn/2022/1155814/ [March 24, 2023].

[3] Chanklan, R., Chaiyakhan, K., Hirunyawanakul, A., Kerdprasop, K., & Kerdprasop, N. (2015, January). "Fingerprint recognition with edge detection and dimensionality reduction techniques.". *International Journal of Applied Pattern Recognition. [On-line]. 3(1)*, pp. 81–21. Available: https://www.semanticscholar.org/5ff9/4db197469c08ebb63d14e25982a86a83fd90.pdf [March 23, 2023].

[4] M.A. Massoud, M. Sabee, M. Gergais and R. Bakhit. (2013, January). "Automated new license plate recognition in Egypt.". *Alexandria Engineering Journal. [On-line]. 52(3)*, pp. 319-326. Available: https://www.sciencedirect.com/science/article/pii/S1110016813000276 [March 24, 2023].

[5] Central Bank of Egypt. (2023, March). “Built in Security Features at Egyptian Banknote.” [On-line]. Available: https://www.cbe.org.eg/en/banknote [April 12, 2023].

• *Appendix*

➢ *The Python Code*

```
# import the required library
import cv2
from skimage.feature import hog
import numpy as np

# Real currency image (Reference image)
real_image = cv2.imread("100dollar-ref.png")

# get the new dimensions to resize the image
width = 800
height = 400
dim = (width, height)
real_re_img = cv2.resize(real_image, dim, interpolation=cv2.INTER_AREA)

# Convert the real image to Grayscale & Blur
real_gray = cv2.cvtColor(real_re_img, cv2.COLOR_BGR2GRAY)
real_blur = cv2.GaussianBlur(real_gray, (3, 3), 0)
# Canny Edge detection the real image
real_edges = cv2.Canny(image=real_blur, threshold1=100, threshold2=200)
# Calculate HOG features for real image
real_hog_features, real_hog_image = hog(real_edges, orientations=8, pixels_per_cell=(8,8),
cells_per_block=(1, 1), visualize=True)

#-----------------------------------------------------------------------------------------#

# Read the test image
img = cv2.imread('100dollar-fake.png')

# get the new dimensions to resize the image
width = 800
height = 400
dim = (width, height)
resized_img = cv2.resize(img, dim, interpolation=cv2.INTER_AREA)

# Display original & test image
cv2.imshow('Real', real_re_img)
cv2.imshow('Test', resized_img)
cv2.waitKey(0)

# Convert to grayscale
test_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
# Blur the image for better edge detection
test_blur = cv2.GaussianBlur(test_gray, (3, 3), 0)

# Display grayscale & Blur image
cv2.imshow('Grayscale & Blur test', test_blur)
cv2.imshow('Grayscale & Blur real', real_blur)
cv2.waitKey(0)

# Canny Edge Detection for test image
test_edges = cv2.Canny(image=test_blur, threshold1=100, threshold2=200)

# Calculate HOG features for test image
test_hog_features, test_hog_image = hog(test_edges, orientations=8, pixels_per_cell=(8,8),
cells_per_block=(1, 1), visualize=True)

```

```
# Display Canny Edge Detection Image
cv2.imshow('Canny Edge Detection test', test_edges)
cv2.imshow('Canny Edge Detection real', real_edges)
cv2.waitKey(0)

# Display HOG features image
cv2.imshow('Test HOG image', test_hog_image)
cv2.imshow('Real HOG image', real_hog_image)
cv2.waitKey(0)

# Calculate the mean intensity of the pixels in the edge-detected images
real_mean = cv2.mean(real_edges)[0]
test_mean = cv2.mean(test_edges)[0]

# Set a threshold for the intensity difference
threshold = 1

# Determine if the test currency is real or fake based on intensity difference
if abs(real_mean - test_mean) < threshold:
    print("The test currency is real based on intensity difference.")
else:
    print("The test currency is fake based on intensity difference.")


# pad test_hog_features with zeros to make it the same size as real_hog_features
padding = np.zeros(real_hog_features.size - test_hog_features.size)
test_hog_features = np.concatenate((test_hog_features, padding))

# compute the distance between the two feature vectors
hog_distance = np.linalg.norm(real_hog_features - test_hog_features)

# Set a threshold for the HOG distance
hog_threshold = 45

# Determine if the test currency is real or fake based on HOG distance
if hog_distance < hog_threshold:
    print("The test currency is real based on HOG distance.")
else:
    print("The test currency is fake based on HOG distance.")

cv2.waitKey(0)
cv2.destroyAllWindows()

```