ISSN: 0975-8585

Research Journal of Pharmaceutical, Biological and Chemical Sciences

Logo Matching Using Advanced Context Dependent Unique Component Analysis.

S Mary Immaculate*, and S Prince Mary.

Faculty of Computing, Sathyabama University, Chennai, Tamil Nadu, India.

ABSTRACT

This paper deals with finding fake logo by matching and recognizing it with the original logo. This is done by convert image of logo into bit map and then converts the bitmap of images into byte array. To compute the unique or hash value for image of logo then compare the each image hash value Taking the hash value of byte array which belongs to the image of logo to be verified check it with the original image index value of the corresponding byte array. From the relation between hash values of both the logo image and the one being considered we could decide it whether it is fake logo or original logo. If the hash values of the entire byte array are exactly matching with the actual logo then it is considered to be original logo, otherwise it is the fake logo. This process is achieved by using Hashing Algorithm.

Keyword: Logo matching, Hashing, Hash SHA-256

**Corresponding author*

INTRODUCTION

Emblems are a special class of visual objects extremely important to assess the identity of something or someone. In industry and commerce, they have essential role to recall customers and their expectations associated with a particular product or service. This economical relevance has motivate the companies in soliciting smart image analysis solutions to verify emblem archives to find evidence of similar already existing logos, discover either inappropriate or non-authorized use of their logo, and reveal the malicious use of logos that have small variations with respect to the originals so to deceive customers. Thus Fake logo detection is necessary and is one of the major research issues of this kind. Earlier logo matching is done by shape matching and object recognition. Later, the two stage algorithm based on spatial-spectral saliency and partial spatial context is used. The existing method is based on context dependent logo matching. In which contextual features are extracted for fake and original images and matched. In this project Advanced Context Dependent Unique Component Analysis is proposed to increase the accuracy of logo matching. In this technique, visual and textual features are extracted and similarity between fake and original is checked with unique component analysis.

Related Work

The previous work on [2] “Shape matching and object recognition using shape contexts,” and [3]“ANSIG—“An analytic signature for permutation-invariant two-dimensional shape representation,” have used different global descriptors of the full logo image either accounting for logo contours or exploiting shape descriptors such as shape context. Previously they are used two algorithms which are spatial-spectral saliency and partial spatial context that accounts for local contexts of key points. They considered spatial-spectral saliency to avoid the impact of cluttered background and speed up the logo detection and localization. Appropriate metrics is accomplished among available methods, using two publicly available funds us data sets. In addition, used normalization method which recorded acceptable results when applied for colour normalization Drawback of this method is only suitable for fully visible logo in the image; it’s not corrupted by noise and is not subjected to changes. According to this, they cannot be applied to real world images. The major limitation of this approach is image resolution and their solution has revealed to be very sensitive to occlusions.

[4]“Unconstrained logo and trademark retrieval in general color image database using color edge gradient co-occurrence histograms”., R.Phan, J.Chia and D.Androutsos,2008 And [5]“Content based retrieval of logo and trademarks in general color image databases using color edge gradient co-occurrence histograms” R.Phan, J.Chia and D.Androutsos,2010 have considered pairs of color pixel in the edge neighborhoods and accumulated differences between pixels at different spatial distances into a color edge co-occurrence histogram. This global descriptor permits to perform fast approximate detection of logo. Drawback of this approaches Unsuited to deal with incomplete information or transformed versions of the original logo.

[6]Context-dependent kernel design for object matching and recognition H.Sahbi, J.Y.Audibert, J.Rabarisoa and R.Kerivan, 2008 And [7] Context-dependent kernels for object classification H.Sahbi, J.Y.Audibert and R.Kerivan, 2011.They are present a novel solution for logo detection and recognition its based on the definition of a “Context-dependent similarity” (CDS) kernel that directly incorporates the spatial context of local features. Drawback of this approaches not restricted to any a priori alignment model. All these methods assume that a logo picture is fully visible in the image, It’s not corrupted by noise and is not subjected to transformations. It cannot be applied to real world images.

Proposed System

In this paper, we present a new solution for logo matching which is based on the definition of “Context-Dependent Unique”. In matching process is done by convert the image of logo into bit of image. Bit of image can be converted into byte value then stored in array. By used hashing algorithm to compute the hash /unique value for each image. This matching process will be very accurate. The solution is proved to be highly effective and responds to the requirements of logo matching in real world images. The probability success of matching is high.

Hashing Technique

Hashing is a key that represents the original or unique value. There are different types of hashing function like MD2,MD4,MD5 and SHA.A hash function works for database storage and retrieval .In proposed system, to compare the original image with test image using the shorter hashed key than to find it using the original value.

To index the original value or key by using hash function .This hashing is always a one-way operation and its not possible to "reverse engineer" the hash function by analyzing the hashed values. It should not generate the same hash value from two different inputs. If it does, this is known as a collision.

SHA-256

In proposed system going to use the SHA 256 Managed class it has 256 bits and used to provide the 128 bits of security against collision attacks. The Compute Hash for a byte array of input data and generate the 256 bit hash of input data. The hash is used as a unique value it has fixed size and representing a large amount of data. Hashes of different input data should match then corresponding inputs are match. Some changes to the data result in large to produce the unpredictable changes in the hash value.

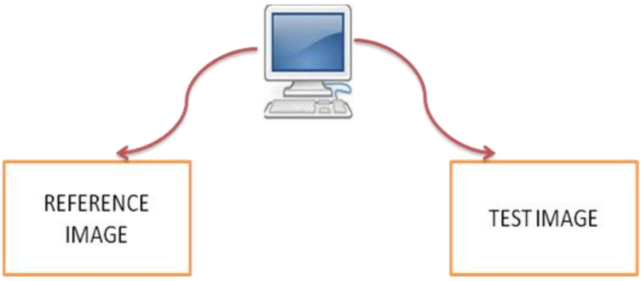
The SHA-256 compression function operates on a 512-bit message block and a 256- bit intermediate hash value. A 256-bit block cipher algorithm is use to encrypts the intermediate hash value using the message block as key. Hence there are two main components to describe: (1) the SHA-256 compression function, and (2) the SHA-256 message schedule.

We will use the following notation:

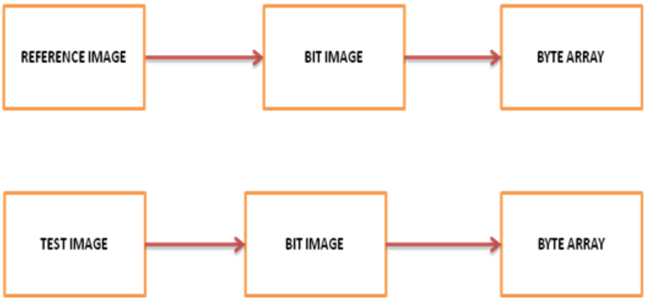
Bitwise XOR , Bitwise AND, Bitwise OR, Bitwise Complement , Mod 232 Addition , Right Shift by n Bits , Right Rotation by N Bits

Step By Step Processing:

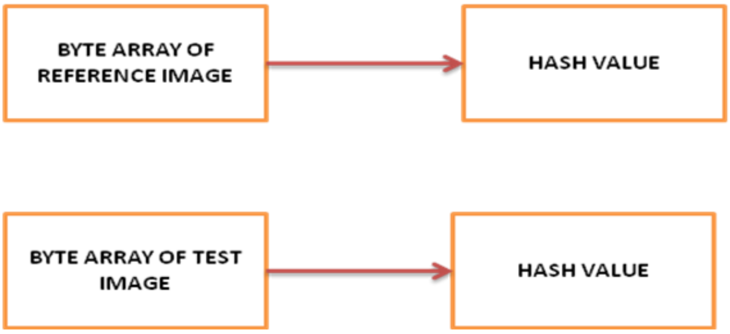
STEP 1: Load Logo Image to fetch the input image from source or system.



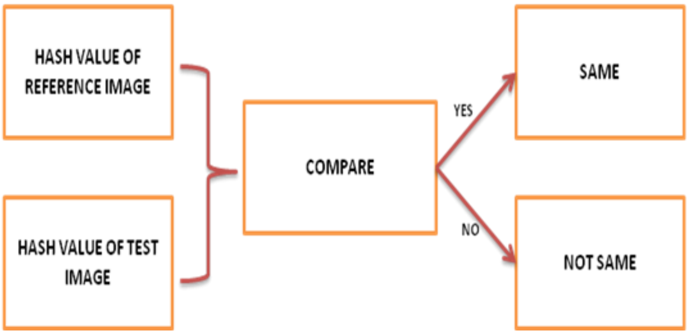
STEP 2: Convert Logo Image into Byte Arrayto convert the image into bit map then convert the bit of images into byte array.



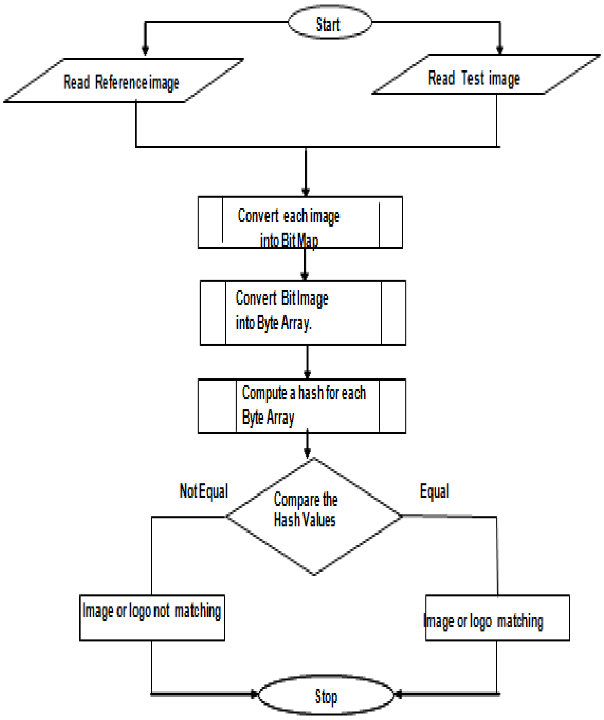
STEP 3: Compute Hash Value to find compute the hash value for each byte array by using Hash Algorithm (SHA256)



STEP 4: Compare Logo to compare the hash value of reference image with hash value of test image for find the both images are same or not.



Processing Flow: In the below diagram show the flow of execution of process .it's very useful to understanding the flow of execution.


Start
Read Reference image
Read Test image
Convert each image into BitMap
Convert BitImage into Byte Array.
Compute a hash for each Byte Array
Not Equal
Compare the Hash Values
Equal
Image or logo not matching
Image or logo matching
Stop

RESULT AND DISCUSSION

Finding the fake logo / image is done by convert image of logo into bit map and then converts the bitmap of images into byte array. To compute the unique or hash value for image of logo then compare the each image hash value Taking the hash value of byte array which belongs to the image of logo to be verified check it with the original image index value of the corresponding byte array. From the relation between hash values of both the logo image and the one being considered we could decide it whether it is fake logo or original logo.

Performance Analysis

Logo Matching: Fetch or load the Image

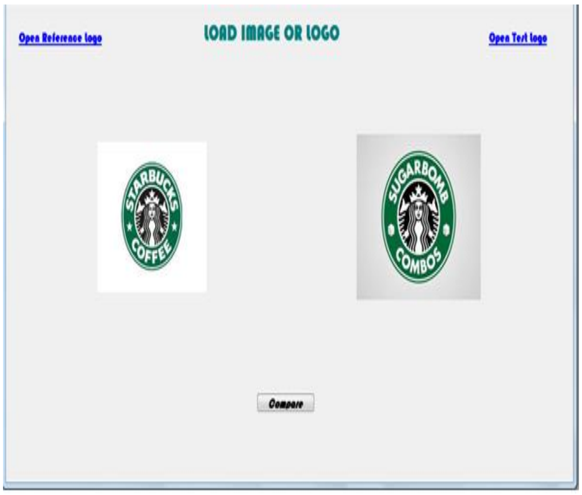
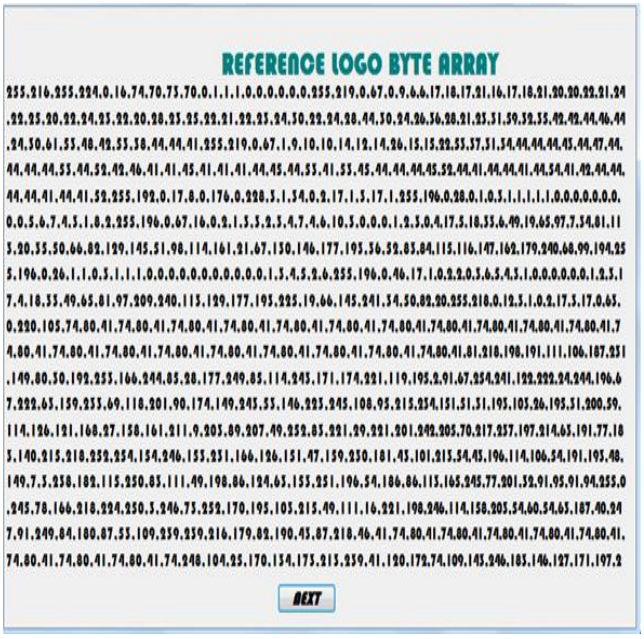
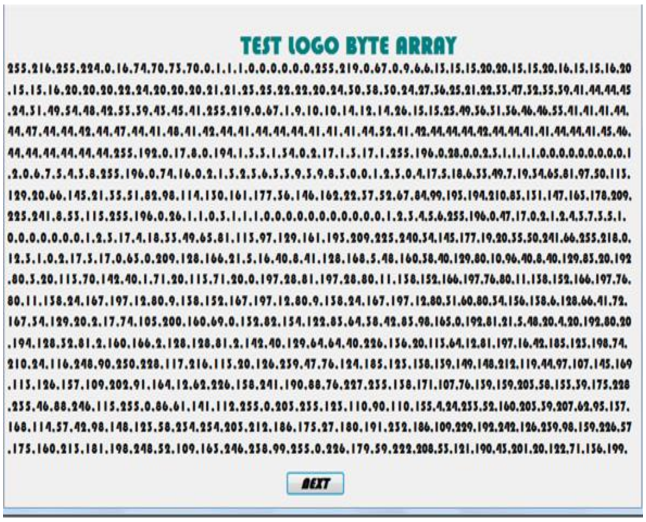


Figure 1: Fetch or load the reference image and test image Byte Array Value



```
255.216.255.224.0.16.74.70.73.70.0.1.1.1.0.0.0.0.0.0.255.219.0.67.0.9.6.6.17.18.17.21.16.17.18.21.20.20.22.21.24
.22.25.20.22.24.25.22.20.28.23.25.22.21.22.23.24.30.22.24.28.44.30.24.26.36.28.21.23.31.59.32.35.42.42.44.46.44
.24.30.61.53.48.42.53.38.44.44.41.255.219.0.67.1.9.10.10.14.12.14.26.15.15.22.53.37.31.34.44.44.44.45.44.47.44.
44.44.44.53.44.52.42.46.41.41.45.41.41.41.44.45.44.53.41.53.45.44.44.44.45.52.44.41.44.44.41.44.54.41.42.44.44.
44.44.41.44.41.52.255.192.0.17.8.0.176.0.228.3.1.34.0.2.17.1.3.17.1.255.196.0.28.0.1.0.3.1.1.1.1.1.0.0.0.0.0.0.0.
0.0.5.6.7.4.3.1.8.2.255.196.0.67.16.0.2.1.3.3.2.3.4.7.4.6.10.3.0.0.0.1.2.3.0.4.17.5.18.33.6.49.19.65.97.7.34.81.11
3.20.35.50.66.82.129.145.51.98.114.161.21.67.130.146.177.193.36.52.83.84.115.116.147.162.179.240.68.99.194.25
5.196.0.26.1.1.0.3.1.1.1.0.0.0.0.0.0.0.0.0.0.0.1.3.4.5.2.6.255.196.0.46.17.1.0.2.2.0.3.6.5.4.3.1.0.0.0.0.0.0.1.2.3.1
7.4.18.33.49.65.81.97.209.240.113.129.177.193.225.19.66.145.241.34.50.82.20.255.218.0.12.3.1.0.2.17.3.17.0.63.
0.220.105.74.80.41.74.80.41.74.80.41.74.80.41.74.80.41.74.80.41.74.80.41.74.80.41.74.80.41.74.80.41.74.80.41.7
4.80.41.74.80.41.74.80.41.74.80.41.74.80.41.74.80.41.74.80.41.74.80.41.74.80.41.81.218.198.191.111.106.187.231
.149.80.30.192.253.166.244.85.28.177.249.85.114.243.171.174.221.119.195.2.91.67.254.241.122.222.24.244.196.6
7.222.63.159.233.69.118.201.90.174.149.243.53.146.223.245.108.95.215.254.151.51.31.193.103.26.195.31.200.59.
114.126.121.168.27.158.161.211.9.203.89.207.49.252.83.221.29.221.201.242.205.70.217.237.197.214.63.191.77.18
3.140.215.218.252.254.154.246.153.231.166.126.151.47.159.230.181.43.101.213.54.43.196.114.106.54.191.193.48.
149.7.3.238.182.115.250.83.111.49.198.86.124.63.153.251.196.54.186.86.113.165.245.77.201.32.91.95.91.94.255.0
.245.78.166.218.224.250.3.246.73.252.170.195.103.215.49.111.16.221.198.246.114.158.203.54.60.54.63.187.40.24
7.91.249.84.180.87.53.109.239.239.216.179.82.190.43.87.218.46.41.74.80.41.74.80.41.74.80.41.74.80.41.74.80.41.
74.80.41.74.80.41.74.80.41.74.248.104.25.170.134.173.213.239.41.120.172.74.109.143.246.183.146.127.171.197.2
```

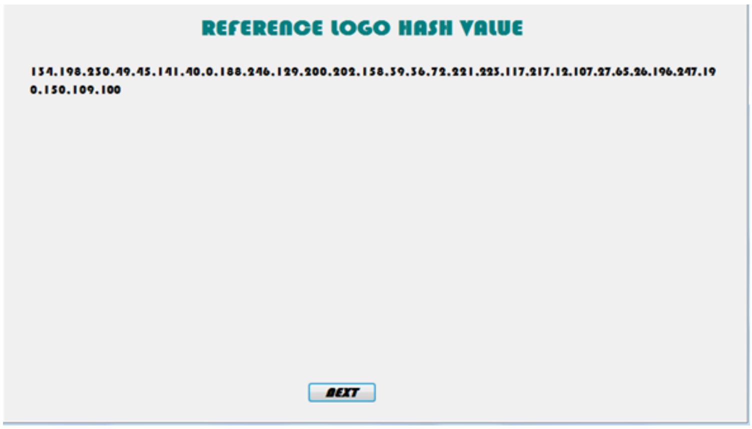
Figure 2: Byte Array Value Reference Logo



TEST LOGO BYTE ARRAY

```
255.216.255.224.0.16.74.70.73.70.0.1.1.1.0.0.0.0.0.0.255.219.0.67.0.9.6.6.13.15.15.20.20.15.15.20.16.15.15.16.20
.15.15.16.20.20.20.22.24.20.20.20.21.21.23.25.22.22.20.24.30.38.30.24.27.36.25.21.22.33.47.32.35.39.41.44.44.45
.24.31.49.54.48.42.53.39.43.45.41.255.219.0.67.1.9.10.10.14.12.14.26.15.15.25.49.36.31.36.46.46.53.41.41.41.44.
44.47.44.44.42.44.47.44.41.48.41.42.44.41.44.44.44.41.41.41.44.52.41.42.44.44.44.42.44.44.41.41.44.44.41.45.46.
44.44.44.44.44.44.255.192.0.17.8.0.194.1.3.3.1.34.0.2.17.1.3.17.1.255.196.0.28.0.0.2.3.1.1.1.1.0.0.0.0.0.0.0.0.0.1
.2.0.6.7.5.4.3.8.255.196.0.74.16.0.2.1.3.2.3.6.3.3.9.3.9.8.3.0.0.1.2.3.0.4.17.5.18.6.33.49.7.19.34.65.81.97.50.113.
129.20.66.145.21.35.51.82.98.114.130.161.177.36.146.162.22.37.52.67.84.99.193.194.210.83.131.147.163.178.209.
225.241.8.53.115.255.196.0.26.1.1.0.3.1.1.1.0.0.0.0.0.0.0.0.0.0.1.2.3.4.5.6.255.196.0.47.17.0.2.1.2.4.3.7.3.5.1.
0.0.0.0.0.0.0.1.2.3.17.4.18.33.49.65.81.113.97.129.161.193.209.225.240.34.145.177.19.20.35.50.241.66.255.218.0.
12.3.1.0.2.17.3.17.0.63.0.209.128.166.21.5.16.40.8.41.128.168.5.48.160.38.40.129.80.10.96.40.8.40.129.83.20.192
.80.3.20.113.70.142.40.1.71.20.113.71.20.0.197.28.81.197.28.80.11.138.152.166.197.76.80.11.138.152.166.197.76.
80.11.138.24.167.197.12.80.9.138.152.167.197.12.80.9.138.24.167.197.12.80.31.60.80.34.156.138.6.128.66.41.72.
167.34.129.20.2.17.74.105.200.160.69.0.132.82.154.122.83.64.38.42.83.98.165.0.192.81.21.5.48.20.4.20.192.80.20
.194.128.32.81.2.160.166.2.128.128.81.2.142.40.129.64.64.40.226.136.20.113.64.12.81.197.16.42.185.123.198.74.
210.24.116.248.90.250.228.117.216.113.20.126.239.47.76.124.185.123.138.139.149.148.212.119.44.97.107.145.169
.113.126.157.109.202.91.164.12.62.226.158.241.190.88.76.227.235.138.171.107.76.139.159.203.58.153.39.175.228
.235.46.88.246.115.255.0.86.61.141.112.255.0.203.235.123.110.90.110.155.4.24.233.52.160.203.39.207.62.95.137.
168.114.57.42.98.148.123.58.234.254.203.212.186.175.27.180.191.232.186.109.229.192.242.126.239.98.159.226.57
.175.160.213.181.198.248.52.109.163.246.238.99.255.0.226.179.59.222.208.53.121.190.43.201.20.122.71.136.199.
```

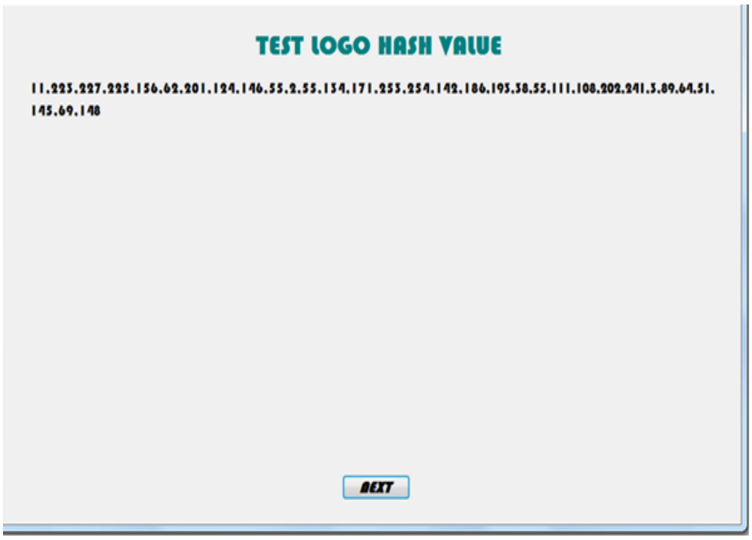
Figure 3: Byte Array value test Logo Hash Value



REFERENCE LOGO HASH VALUE

```
134.198.230.49.45.141.40.0.188.246.129.200.202.158.39.36.72.221.223.117.217.12.107.27.65.26.196.247.19
0.150.109.100
```

Figure 4: To Compute the Hash Value_Reference_Logo



TEST LOGO HASH VALUE

```
11.223.227.225.156.62.201.124.146.55.2.55.134.171.253.254.142.186.193.38.55.111.108.202.241.3.89.64.51.
145.69.148
```

Figure 5: To Compute the Hash Value_Test_logo Result: Logo/Image Matching

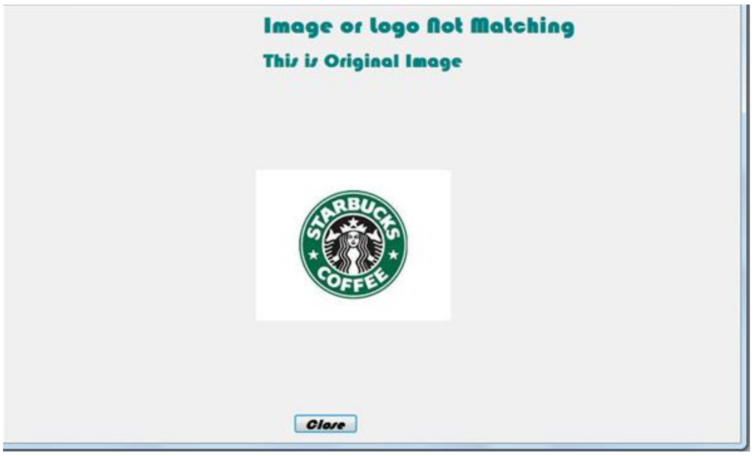


Image or Logo Not Matching

This is Original Image

Figure 5: Logo / Image Matching

CONCLUSION

Hashing technique is used to find the fake logo / image. This technique is depends on hash value of logo / images for compare the original with fake images/logo. The result is used to increase the accuracy of logo matching and highly effective and responds the requirements of logo detection and recognition in real world image. The probability of success of matching and detection is high and very secure. It is used for the purpose of avoid duplicate Logos in the market.

REFERENCES

[1] Hichem Sahbi, Lamberto Ballan Giuseppe Serra, and Alberto Del Bimbo. IEEE Trans Image Proc 2013;22(3).

[2] S Belongie, J Malik and J Puzicha. IEEE Trans Pattern Anal Match Intell 2002;24(4):509-522.

[3] J Rodriguez, P Aguiar and J Xavier. IEEE Conf Comput Vis Pattern Recognit Anchorage Ak 2008:1 – 8.

[4] R Phan, J Chia and D Androutsos. Proc IEEE Int Conf Acoust Speech Signal Process, Las Vegas, NV, 2008:1221 - 1224.

[5] R Phan ,J Chia and D Androutsos. Comput Vis Image Understand 2010;114(1):66 -84.

[6] H Sahbi, JY Audibert, J Rabarisoa and R Kerivan. IEEE Conf Comput Vis Pattern Recognit Anchorage, AK, 2008:1-8.