International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume: 3 | Issue: 4 | May-Jun 2019 Available Online: www.ijtsrd.com e-ISSN: 2456 - 6470

Authenticate Aadhar Card Picture with Current Image using Content-Based Image Processing

Nehali M. Ghosalkar

ASM Institute of Management & Amp; Computer Studies (IMCOST), Thane, Maharashtra, India

How to cite this paper: Nehali M. Ghosalkar "Authenticate Aadhar Card Picture with Current Image using Content-Based Image Processing" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-3 | Issue-4, June 2019, pp.1255-1260, URL: https://www.ijtsrd.com/papers/ijtsrd25070.pdf

IJTSRD25070

ABSTRACT

This paper proposes to give review on algorithms which helps to match human face on Aadhar card with their current image using Content based Image Retrieval (CBIR).The concert of Content-Based Image Retrieval (CBIR) system is depends on competent feature extraction and accurate repossession of similar images. Content based image retrieval is the work for retrieving the images from the large collection of database on the basis of their own visual content. This paper express the method to obtain better retrieval efficiency from current picture of person which will give maximum matching score with same person Aadhar card photo.

Keywords: *CBIR, FLANN, ORB, Image Processing, Image Comparison*

Introduction

Image comparison is one of the essential processes in the field of image processing. Sometimes it is necessary to compare two images to estimate the similarity and dissimilarity between them. Often, images have to be compared, to facilitate choices of visualization and reproduction parameters respectively. Assessment of image similarity is an important problem of image analysis. Procedures of similarity between two images are useful for the comparison of algorithms dedicated to noise reduction, image matching, image coding and reinstatement. This paper presents feature base comparison methods and tools which are used for comparing images.

Normally there exist two approaches for penetrating and to retrieving images. The first one is based on textual information done manually by a human. This is called concept-based or text-based image indexing.

[8] A human describes and valuate the images according to the image content, the caption, or the background information. However, the illustration of an image with text requires significant effort and can be expensive, dreary, time consuming. To overcome the limitations of the text-based approach, the second approach known as Content-Based Image Retrieval (CBIR) techniques are used. In a CBIR system, images are mechanically indexed by visualizing their respected features such as color, texture, and shape.[8][7]

EXISTING TECHNIQUE:

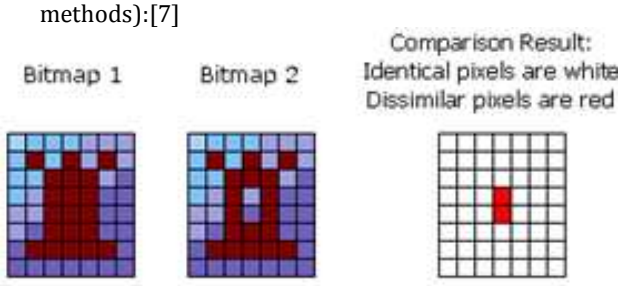
Methods:

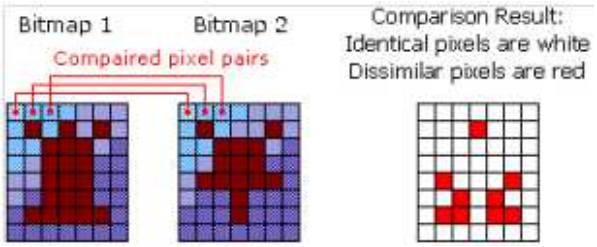
Pixel-by-pixel comparison:

The comparison train gets the color of pixels that have the same coordinates within the image and compares this color. If the color of each pixel of both images coincides, Test Complete considers the two images to be identical.[8]

1. **Pixel Tolerance** (Pixel Tolerance in scripting methods): Defines the allowed number of dissimilar pixels. If the number of different pixels is less than or equal to Pixel Tolerance, then Test Complete considers the images to be identical. For instance, in the image below, one bitmap differs from another by two pixels. If the Pixel Tolerance value were 2, Test Complete would consider these bitmaps to be identical.[8]

2. **Color Tolerance** (Color Tolerance in scripting methods):[7]

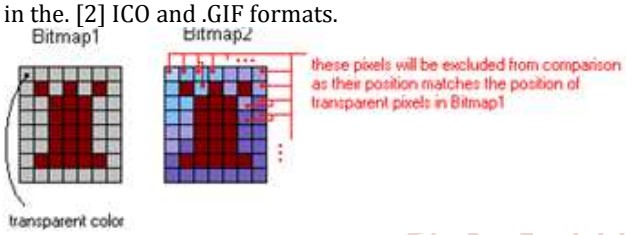




specify an satisfactory color difference at which two pixels should be treated as equal. The color difference is represented as an integer value within the range of 0-255 that specifies an acceptable difference for each color component (red, green and blue) of the compared pixels.If the difference between intensities of each of their color components does not exceed the specified value then two pixels are considered identical. When Color Tolerance is 0, which is the default value, the compared pixels are considered identical only if they have exactly the same color. When Color Tolerance is 255, pixels of any color are considered identical.[8][13]

3. **"Transparent" Color** (Transparent in scripting methods):

If this parameter is True (enabled), Test Complete treats the color of the upper-left pixel of the baseline image as a transparent color and does not compare all the pixels that have the same coordinates in the compared image as the transparent pixels have in the baseline image. This parameter is similar to the way transparency is implemented in the. [2] ICO and .GIF formats.



For instance, if the top-left pixel is gray, then all gray pixels of the first image will match pixels of any color that have the same coordinates in the second image.

Feature-Based Image Comparison:

In computer vision terminologies, image feature is a mathematical description of the raw image. Normally speaking, comparing the image features is more efficient and accurate than comparing raw images. Three steps are implicated for feature-based image comparison: feature detection, feature description, and feature matching.[13]

1. Feature detection is to detect feature points such as corner points, from the image.
2. In SIFT (Scale Invariant Feature Transform), the feature descriptor is a histogram of the leaning local gradient around the key point. The bins of the histogram are stored in a vector typically with 128 entries. In SURF (Speeded-Up Robust Feature), the descriptor is composed of the Haar wavelet responses with typically 64 entries.[8][18][19]
3. The third step feature matching, both SIFT and SURF use Euclidean distance [3][18][19]

Pre-processing methods:

- Geometric Adjustments
- Radiometric/Intensity Adjustments
- Intensity Normalization
- Homomorphic Filtering
- Illumination Modeling
- Linear Transformations of Intensity
- Sudden Changes in Illumination
- Speckle Noise
- Predictive models:
- Spatial models
- Temporal model.[12][1]

ASIFT:

A New Framework for Fully Affine Invariant Image Comparison

This method presents affine-SIFT (ASIFT), simulate all image views accessible by varying the two camera axis orientation parameters, namely, the latitude and the longitude angles, left over by the SIFT method. Then it covers the SIFT method for the other four parameters. The resulting method will be mathematically proved to be fully affine invariant.[6][7]

Tools

ImageMagick:

ImageMagick is a free and open-source software set for displaying, converting, and editing raster image and vector image files. It can read and write over 200 image file formats. ImageMagick is licensed under the Apache 2.0 license. The software mainly consists of a number of command-line interface utilities for manipulat images. ImageMagick does not have a robust graphical user interface to edit images as do Adobe Photoshop and GIMP, but does include – for Unix-like operating systems – a basic native X Window GUI (called IM Display) for interpretation and manipulating images and API libraries for many programming languages. The difference is highlighted in red color.[8]

Perceptual Diff:

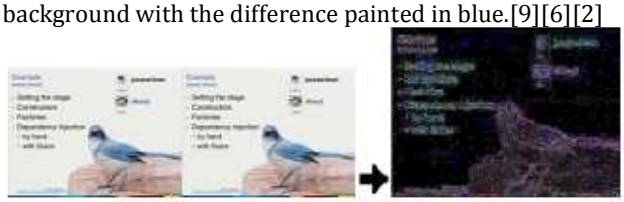
Perceptual Diff is an open source command line image comparison tool.

The package can be used in two different ways:

- per command line; just as the original project
- through a class in your code

The command-line tool can be found in the bin directory. You can run the application with node./bin/ perceptualdiff.js <image1> .png <image2> .png.

Use image1 and image2 as the images you want to compare. The default options for the output file will only show a black background with the difference painted in blue.[9][6][2]



1. **Image Diff:**

Image Diff is another GUI based image comparison freeware tool which is easy. After installation, run the program, click on the "Left" button to select the first image then clicking on the "Right" button to select the second image. Optionally, you can increase the threshold level if the images contain a lot of small differences which you'd like imageDiff to ignore. Click the Compare button and the differences can be shown in either 4 different modes (Monochrome-Ray, Predator, and Thermal).[7][6][8]

2. **ImageJ:**

ImageJ is a public domain, Java-based image processing program. ImageJ was designed with an open architecture that provides extensibility via Java plugins and recordable

macros. Analysis and processing plugins can be developed using ImageJ's built-in editor and a Java compiler. ImageJ can exhibit, edit, analyze, process, save, and print 8-bit color and grayscale, 16-bit integer, and 32-bit floating point images. It can read many image file formats, including TIFF, PNG, GIF, JPEG, BMP, DICOM, and FITS, as well as raw formats.[3][5][2]

3. OpenCV:

OpenCV (Open Source Computer Vision) is a library used for real-time computer vision. This is cross-platform and free for use under the open-source BSD license. OpenCV's application areas include:[9][11][2]

- 2D and 3D feature toolkits
- Ego motion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Motion understanding
- Object identification
- Segmentation and recognition
- Stereopsis stereo vision: depth perception from 2 cameras
- Structure from motion (SFM)
- Motion tracking
- Augmented reality

To support some of the above areas, OpenCV includes a statistical machine learning library that contains:

- Boosting (meta-algorithm)
- Decision tree learning
- Gradient boosting trees
- Expectation-maximization algorithm
- k-nearest neighbor algorithm
- Naive Bayes classifier
- Artificial neural networks
- Random forest
- Support vector machine (SVM)[7]

PROBLEM STATEMENT:

As all over above cases we need to match size of Images then only it will be compare accurately in this procedure we can say that text, shape ,color of two images should match.

The Diagonal and vertical of images position get distract if they are not in proper position, that means if image is in rotation format it will generate an mismatch result then you should put it in straight to match there all diagonal points exactly.

Feature comparison technique like SURF,SIFT,BREF performed poorly with Rotation and also SURF and STIF required license.

If the extention of Image not get matched then also it will create an error that is, JPG-PNG,JPEG-TIFF etc different types of images will not carry forward with their properties to match with onther one so here it will generate Wrong output

PROPOSED METHODOLOGY:

As per research shows OpenCv library providing huge support to Image Processing. I will try to expose them which will help to compare face and get their percentage or marks of matching.

We will use here OpenCv Feature Based image comparison technique with FLANN which one of Kmatcher. Here we are following step by considering some inbuilt libraries which helps us to reached our goal.[11]

Steps That can Follow:

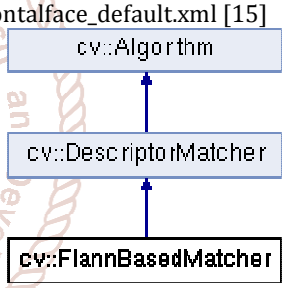
Step 1: First allow correspondence libraries to work on like, OpenCV , Matplotlib, sys , os, numpy, imutils[23]

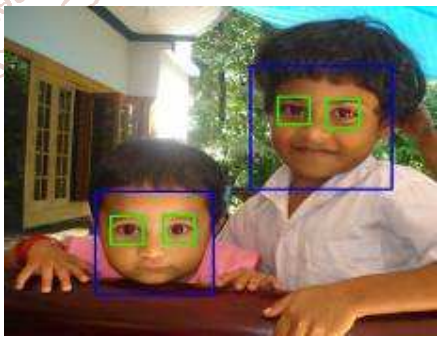
Step 2: We are applying haarcascad xml file for detecting a face and eye. The Haar Cascade is trained by superimposing the positive image over a set of negative images. The training is generally done on a server and on various stages. Better results are obtained by using high quality images and increasing the amount of stages for which the classifier is trained.[11]

OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in [22][23] opencv/data/haarcascades/ folder some of them are as follow:

~/OpenCV/opencv/data/haarcascades$ ls
haarcascade_eye.xml [14]

haarcascade_frontalface_default.xml [15]





Step3: We will use ORB descriptor. This algorithm was brought up by Ethan Rublee, Vincent Rabaud, Kurt Konolige and Gary R. Bradski in their paper ORB: An efficient option to SIFT or SURF in 2011. As the title says, it is a good alternative to SIFT and SURF in computation cost, matching performance and mainly the patents.

ORB is basically a fusion of FAST keypoint detector and BRIEF descriptor with many modifications to enhance the performance. First it use FAST to find keypoints, then apply Harris corner measure to find top N points among them. It also use pyramid to produce multiscale-features.[11]

Step 4: We now match two image descriptor with KNN(Key Nearest Neighbor) matcher which is FLANN . cv::FlannBasedMatcher interface in order to perform a quick and efficient matching by using the Clustering and Search in Multi-Dimensional Spaces module.[10]
FLANN stands for Fast Library for Approximate Nearest Neighbors. It contains a collection of algorithms optimized for fast nearest neighbor search in large datasets and for high dimensional features. It works faster than BFMatcher for large datasets. We will see the second instance with FLANN based matcher.

For FLANN based matcher, we need to pass two dictionaries which specifies the algorithm to be used, its related parameters etc. First one is IndexParams. For various algorithms, the information to be passed is explained in FLANN docs. As a summary, for algorithms like SIFT, SURF etc. you can pass following:

FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
While using ORB, you can pass the following. The commented values are recommended as per the docs, but it didn't provide required results in some cases. Other values worked fine.:

FLANN_INDEX_LSH = 6
index_params= dict(algorithm = FLANN_INDEX_LSH,
table_number = 6, # 12
key_size = 12, # 20
multi_probe_level = 1) #2

Second dictionary is the SearchParams. It specifies the number of times the trees in the index should be recursively traversed. Higher values gives better precision, but also takes more time. If you want to change the value, pass search_params = dict(checks=100).[16][17][22]

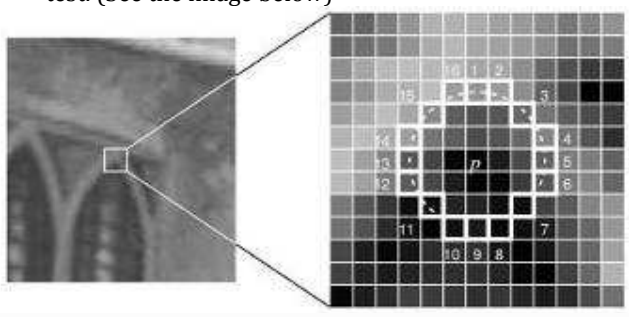
PROPOSED ALGORITHM:
ORB (Oriented FAST and Rotated BRIEF)[11]

A. FAST (Features from Accelerated Segment Test)

This algorithm was projected by Edward Rosten and Tom Drummond in their paper "Machine learning for high-speed corner detection" in 2006 (Later revised it in 2010).

Feature Detection using FAST

1. Select a pixel p in the image which is to be identified as an interest point or not. Let its intensity be Ip.
2. Select appropriate threshold value t.
3. Consider a circle of 16 pixels around the pixel under test. (See the image below)



4. Now the pixel p is a corner if there exists a set of n contiguous pixels in the circle (of 16 pixels) which are all brighter than I_p + t, or all darker than I_p − t. (Shown as white dash lines in the above image). n was chosen to be 12.
5. A high-speed test was planned to prohibit a large number of non-corners. This test examines only the four pixels at 1, 9, 5 and 13 (First 1 and 9 are tested if they are too brighter or darker. If so, then checks 5 and 13). If p is a corner, then at least three of these must all be brighter than I_p + t or darker than I_p − t. If neither of these is the case, then p cannot be a corner. The full segment test criterion can then be applied to the passed candidates by examining all pixels in the circle. This detector in itself exhibits high performance, but there are several weaknesses:[20][21]
   - It does not reject as many candidates for n < 12.
   - The choice of pixels is not optimal because its efficiency depends on ordering of the questions and distribution of corner appearances.
   - Results of high-speed tests are thrown away.
   - Multiple features are detected adjacent to one another.

First 3 points are addressed with a machine learning approach. Last one is addressed using non-maximal suppression.

Machine Learning a Corner Detector

1. Select a set of images for training (preferably from the target application domain)
2. Run FAST algorithm in every images to find feature points.
3. For every feature point, store the 16 pixels around it as a vector. Do it for all the images to get feature vector P.
4. Each pixel (say x) in these 16 pixels can have one of the following three states:

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \le I_p - t & \text{(darker)} \\ s, & I_p - t < I_{p\to x} < I_p + t & \text{(similar)} \\ b, & I_p + t \le I_{p\to x} & \text{(brighter)} \end{cases}$$

5. Depending on these states, the feature vector P is subdivided into 3 subsets, Pd, Ps, Pb.
6. Define a new boolean variable, Kp, which is true if p is a corner and false otherwise.
7. Use the ID3 algorithm (decision tree classifier) to query each subset using the variable Kp for the knowledge about the true class. It selects the x which yields the most information about whether the candidate pixel is a corner, measured by the entropy of Kp.
8. This is recursively applied to all the subsets until its entropy is zero.
9. The decision tree so created is used for fast detection in other images.

Non-maximal Suppression

Detecting multiple interest points in adjacent locations is another problem. It is solved by using Non-maximum Suppression.

1. Compute a score function, V for all the detected feature points. V is the sum of absolute difference between p and 16 surrounding pixels values.
2. Consider two adjacent keypoints and compute their V values.
3. Discard the one with lower V value.[20][8][16]

B. BREIF(Binary Robust Independent Elementary Features)

SIFT uses 128-dim vector for descriptors. Since it is using floating point numbers, it takes basically 512 bytes. Similarly SURF also takes minimum of 256 bytes (for 64-dim). Creating such a vector for thousands of features takes a lot of memory which are not feasible for resouce-constraint applications especially for embedded systems. Larger the memory, longer the time it takes for matching.

SIFT descriptors provides better speed-up because finding hamming distance is just applying XOR and bit count, which are very fast in modern CPUs with SSE instructions. But here, we need to find the descriptors first, then only we can apply hashing, which doesn't solve our initial problem on memory. BRIEF comes into picture at this moment. It provides a shortcut to find the binary strings directly without finding descriptors. It takes smoothened image patch and selects a set of n_d (x,y) location pairs in an unique way (explained in paper). Then some pixel intensity comparisons are done on these location pairs. For eg, let first location pairs be p and q. If I(p) < I(q), then its result is 1, else it is 0. This is applied for all the n_d location pairs to get a n_d-dimensional bitstring. This n_d can be 128, 256 or 512. OpenCV supports all of these, but by default, it would be 256 (OpenCV represents it in bytes. So the values will be 16, 32 and 64). So once you get this, you can use Hamming Distance to match these descriptors.

Now for descriptors, ORB use BRIEF descriptors. But we have already seen that BRIEF performs poorly with rotation. So what ORB does is to "steer" BRIEF according to the orientation of keypoints. For any feature set of n binary tests at location (x_i, y_i), define a 2 \times n matrix, S which contains the coordinates of these pixels. Then using the orientation of patch, \theta, its rotation matrix is found and rotates the S to get steered(rotated) version S_\theta.

ORB discretize the angle to increments of 2 \pi /30 (12 degrees), and construct a lookup table of precomputed BRIEF patterns. As long as the keypoint orientation \theta is consistent across views, the correct set of points S_\theta will be used to compute its descriptor.[21][19]

RESULT & PERFORMANCE ANALYSIS:

- Though all the process of Proposed Methodology. In haarcascade_eye.xml , haarcascade_frontalface_default.xml we get an output like this:

```
face_cascade
=cv2.CascadeClassifier('D:\\path_to_haarface\\haarcasc
ade_frontalface_default.xml')

eye_cascade
=cv2.CascadeClassifier('D:\\path_to_eyedetect\\haarcas
cade_eye.xml')
```

- ORB will detect the points to match and FLANN matcher will match all coordinates which is found by ORB.

- We will get Maximum Matching point count to identify whether face are matching or not.

Like; 8/10 mathching point will say images are somehow same.

CONCLUSION & FUTURE WORK:

Our proposed CBIR system was evaluated by different images query. The execution results presented the success of the proposed method in retrieving and matching similar images from the images database and outperformed the other CBIR systems in terms of average precision and recall rates. This can be represented from the precision and recall values calculated from the results of retrieval where the average precision and recall rates were 0.882 and 0.7002 respectively. In the future, filtering techniques will be employed to get more accurate results in the content based image retrieval system.

PUBLICATIONS & REFERENCES:

[1] An efficient similarity measure for content based image retrieval using memetic algorithm, Mutasem K. Alsmadi, Egyptian Journal of Basic and Applied Sciences 4 (2017) 112–122

[2] A review on various approaches for content based image retrieval based on shape, texture and color features Ankur Gupta. International Journal of Academic Research and Development ISSN: 2455-4197 Impact Factor: RJIF 5.22 www.academicsjournal.com Volume 3; Issue 1; January 2018; Page No. 177-180

[3] Classification of biomedical images using content based image retrieval systems. [Zhang et. al., Vol.5 (Iss.2): February, 2018], ISSN: 2454-1907 DOI: 10.5281/zenodo.1186565

[4] A Conceptual Study on Image Matching Techniques. Global Journal of Computer Science and Technology Vol. 10 Issue 12 (Ver. 1.0) October 2010 Page.

[5] Image Quantification Learning Technique through Content based Image Retrieval Dr. R. Usha Rani#1

[6] Rotation Invariant Content Based Image Retrieval System for Medical Images. International Journal on Future Revolution in Computer Science & Communication Engineering ISSN: 2454-4248 Volume: 4 Issue: 3

[7] [7] Various Approaches of Content Based Image Retrieval Process: A Review agan Madaan. International Journal of Scientific Research in Computer Science, Engineering and Information Technology © 2018 IJSRCSEIT | Volume 3 | Issue 1 | ISSN : 2456-3307

[8] A Review on Different Categories of CBIR Methods Latha1, Dr. Y. Jacob Vetha Raj2. International Journal of Scientific Research in Computer Science, Engineering and Information Technology © 2018 IJSRCSEIT | Volume 3 | Issue1 | ISSN : 2456-3307

[9] Image Comparison Methods & Tools: A Review. 1st National Conference on, EMERGING TRENDS IN INFORMATION TECHNOLOGY[ETIT], 28th -29th December 2015 pg 35-42

[10] How-To: Python Compare Two Images by Adrian Rosebrock on September 15, 2014 in Image Processing, Tutorials.

[11] k-NN classifier for image classification by Adrian Rosebrock on August 8, 2016 in Machine Learning, Tutorials.

[12] https://opencv-python-tutroals.readthedocs.io//lenatest/py_tutorials/py_feature2d/py_orb/py_orb.html

[13] https://www.igi-global.com/dictionary/content-based-image-retrieval-cbir/5587.

[14] Cattle Identication using Muzzle Print Images based on Texture Features Approach Conference Paper · January 2014 DOI: 10.13140/2.1.3685.1202

[15] https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_eye.xml

[16] https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml

[17] https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html

[18] https://docs.opencv.org/3.4/d5/d6f/tutorial_feature_flann_matcher.html

[19] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html#sift-intro

[20] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html

[21] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_fast/py_fast.html

[22] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_brief/py_brief.html

[23] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html