2nd Report

Face Detection Techniques And Algorithms

Supervised by:
Prof. Oussama El Issati
Prepared by:
Ait Said Noureddine
Ennouali Mohamed Amine

May 30, 2017

Contents

Chapter 1

Face detection techniques and algorithms

1.1 Introduction

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

Object detection algorithms are used in face detection and face recognition. It is also used in tracking objects, for example tracking a ball during a football match, tracking movement of a cricket bat, tracking a person in a video.

The concept is that every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point (i.e. the center) are sought. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin color and distance between eyes can be found[1].

1.2 Face detection and tracking algorithms

Cascading is an ensemble learning based on the concatenation of several Classifiers, using all information collected from the output from a given classifier as additional information for the next classifier in the cascade. Cascading is a multistage system.

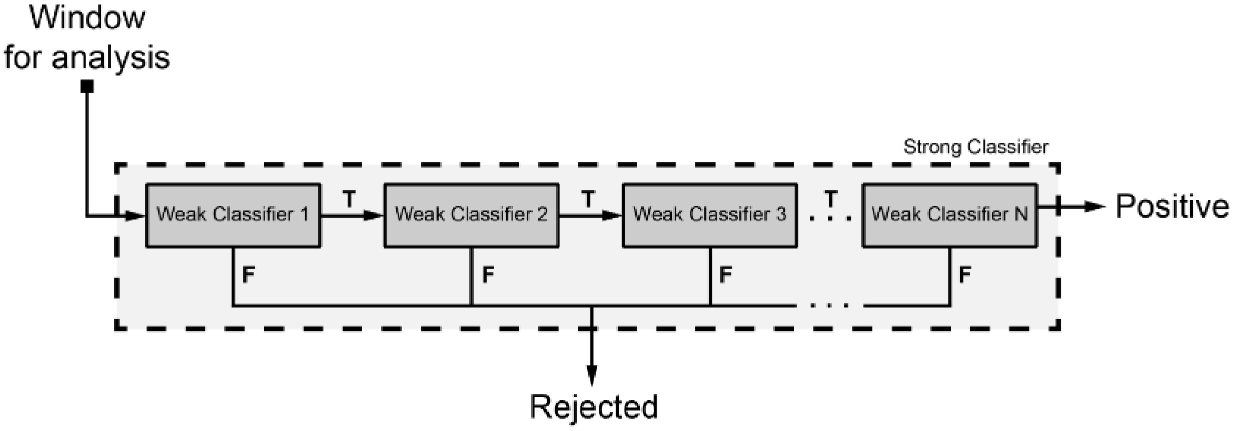
Cascading classifiers are trained with several hundred "*positive*" sample views of a particular object and arbitrary "*negative*" images of the same size.

After the classifier is trained it can be applied to a region of an image and detect the object in question. To search for the object in the entire frame, the search window can be moved across the image and check every location for the classifier. This process is most commonly used in image processing for object detection and tracking, primarily facial detection and recognition.

1.2.1 Cascade classification based algorithm: Viola and Jones

The first cascading classifier is the face detector of *Viola and Jones (2001).* The requirement was that the classifier be fast in order to be implemented on low CPU systems, such as cameras and phones[1].

Figure 1.1: Concept of cascade classification[5]



Characteristics of the algorithm

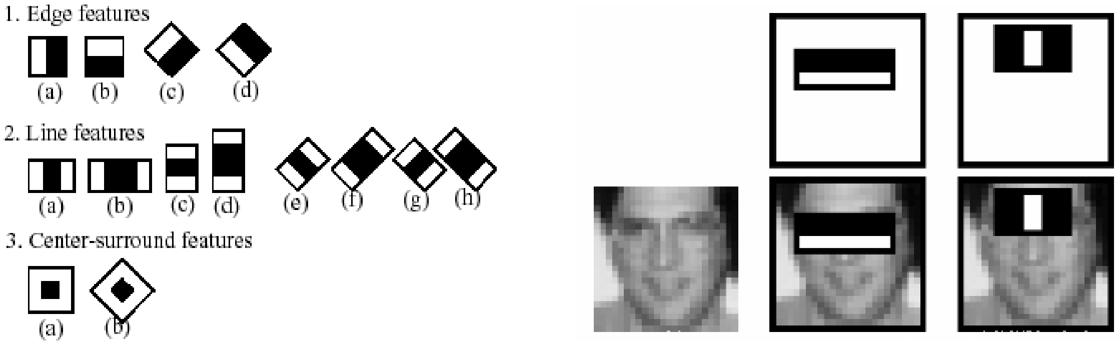
- **Robust:** the algorithm has a very high detection rate (true-positive rate) and very low false-positive rate[1].
- **Real Time:** At least 2 frames per second are processed thus making it a quick and an efficient algorithm. The algorithm comprises of four stages:

  1. Haar Features Selection
  2. Creating Integral Image
  3. Adaboost Training Algorithm
  4. Cascade Classifiers

HAAR Features

Haar-like features are digital image features used in object detection[2]. Haar features are similar to convolution kernels which are used to detect the presence of a feature in the given image.

Figure 1.2: Haar features applied to detect faces[4].



A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. [4]

There is a huge number of possible sizes and locations of each kernel. For example using a 24x24 window results over 160000 features. For each feature calculation, we need to find sum of pixels under white and black rectangles. To solve this, they introduced the integral images to simplify calculation of sum of pixels[3].

Integral image

he algorithm introduces the concept of integral image to find the sum of all the pixels under a rectangle with just 4 corner values instead of summing up all the values.

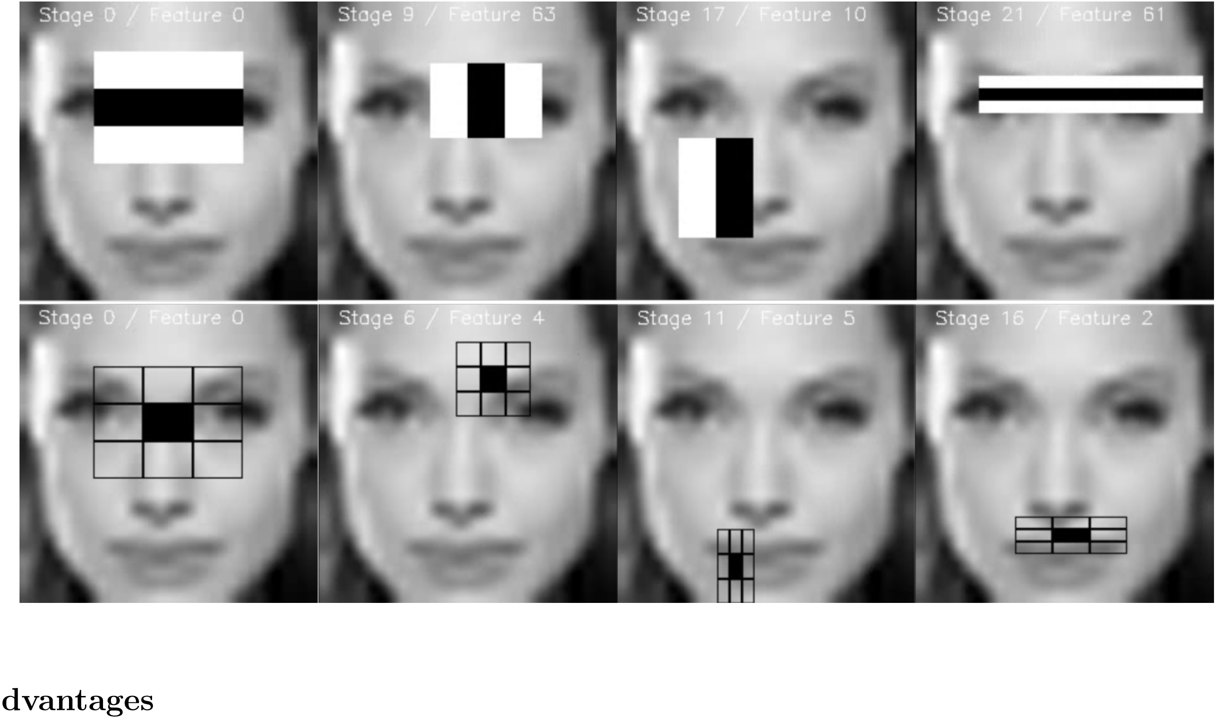
Adaboost

Since not all the features are relevant, we have to select only the best features using Adaboost. Which is a machine learning algorithm which helps in finding only the best features among all the 160000+ features. After these features are found, a weighted combination of all these features is used in evaluating and deciding if any given window (24x24) has a face or not. Each of the selected feature is considered to be included if they can at least perform better than random guessing. The little classifiers are called "weak classifiers", Adaboost constructs a strong classifier as a linear combination of these weak classifiers.

HAAR classifier generating

The process of generating a classifier includes two stages: the training and the detection stage. The detection stage using either HAAR or LBP based models. The library **OpenCV** comes with preprogramed applications that we can use to generate a classifier in *.xml* file, this file contains all the features that will be used to look for faces in future applications, OpenCV comes also with pre-generated *.xml* files ready to use.

Figure 1.3: The process of classifier generation using OpenCV.



Advantages

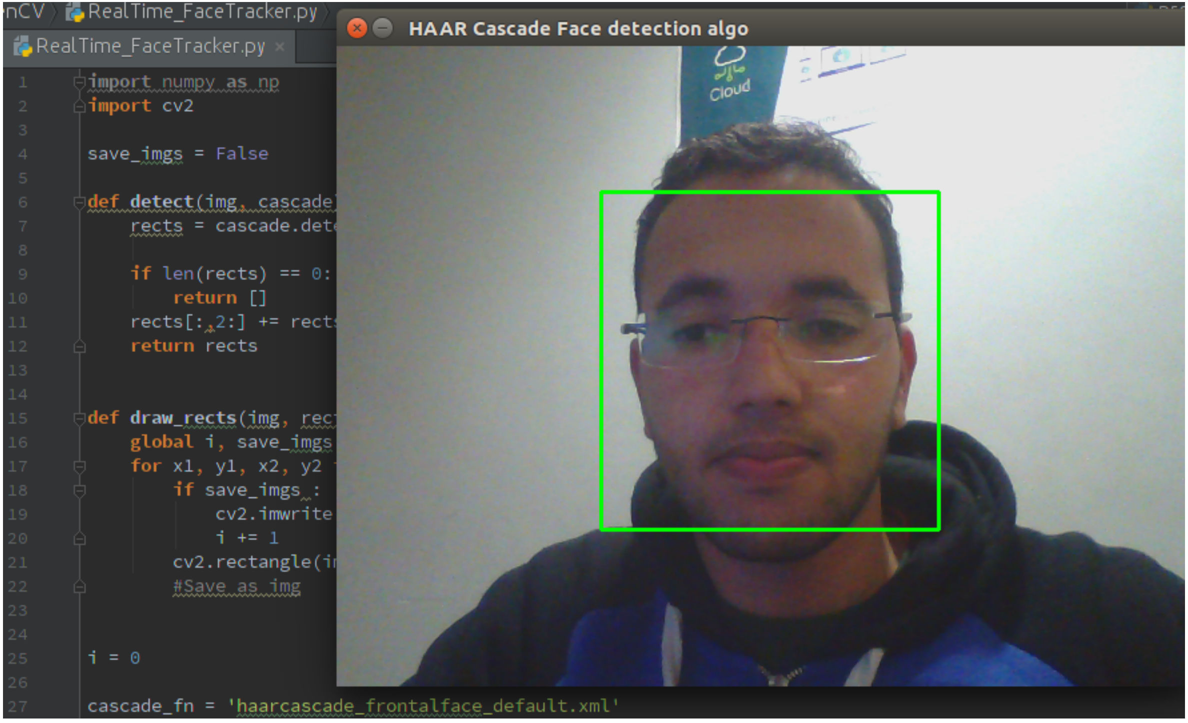
- Fast features are computed very quickly.
- The features are scaled instead of scaling the image.
- This is a generic detection scheme which can be used to detect other objects like hands, buildings, etc.

Disadvantages

- The detector is effective only in the case of frontal images of the face.
- If the face is turned 45 degrees, it fails to detect the face.
- It is sensitive to lighting conditions.
- Due to overlapping sub-windows, we might face the problem of multiple objects being detected as face.

Results

Figure 1.4: Default OpenCV HAAR cascade classifier test.



```
import numpy as np
import cv2

save_imgs = False

def detect(img, cascade
    rects = cascade.dete

    if len(rects) == 0:
        return []
    rects[:,2:] += rect
    return rects


def draw_rects(img, rec
    global i, save_imgs
    for x1, y1, x2, y2
        if save_imgs :
            cv2.imwrite
            i += 1
        cv2.rectangle(i
        #Save as img


i = 0

cascade_fn = 'haarcascade_frontalface_default.xml'
```

1.2.2 CAMshift Algorithm

CAMshift is a tracking algorithm, which is based on MeanShift algorithm, what CAMshift does meanShift in every single frame of a video, and record the results we got by MeanShift. CamShift algorithm includes these three parts[6] :

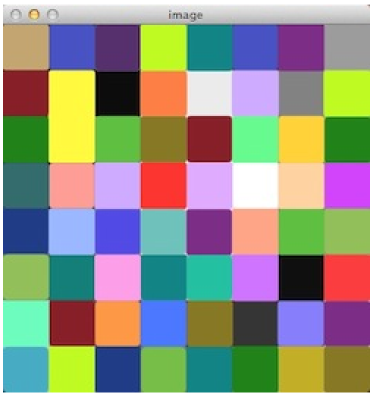
1. Back projection
2. Applying MeanShift
3. Tracking

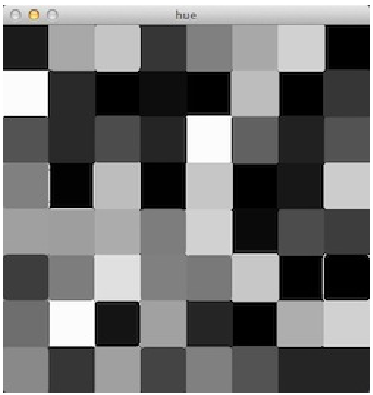
Back Projection

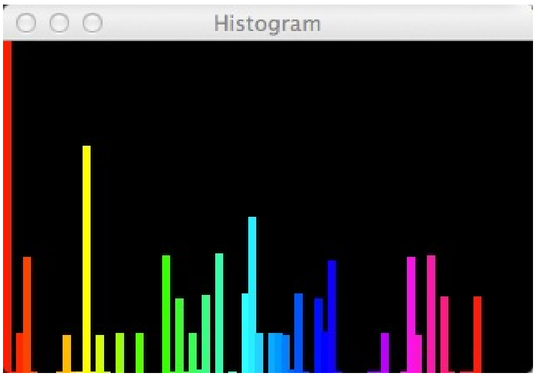
Back projection is a method using the histogram of an image to show up the probabilities of colors that may appear in each pixel. First we transform the picture space to HSV space which is a cylindric color base (or any space which include an H channel that represent the hue of each pixel, of course, value of hue is between 0 to 180). Secondly, we split the H channel out, as a single grayscale image, and get its histogram, and normalize it. Thirdly, we use “calcBackProject()” function to calculate the back projection of the image.

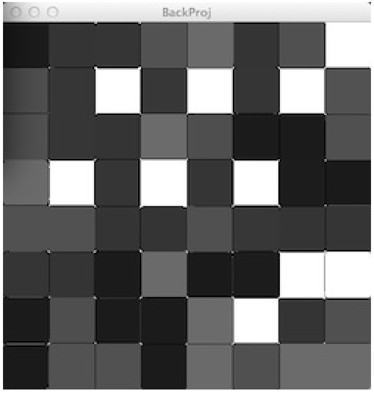
Example:

This is an example to explain how we get the back projection. We transform the picture into HSV space and the second image shows the hue channel. The third image shows its histogram. We calculate the weight of each color in the whole picture using histogram, and change the value of each pixel to the weight of its color in whole picture, the result of this step is shown in the last image.









Applying MeanShift

MeanShift is an algorithm which finding modes in a set of data samples representing an underlying probability density function, so the whole algorithm is:

1. Initialize the sphere, including the center and radius.
2. Calculate the current mass center.
3. Move the sphere’s center to mass center
4. Rrepeat step b and c, until converge, that is, current mass center after calculate, is the same point with center of sphere.

Track

The last step is tracking, if we have a video, or frames captured by our web camera, what we need to do is just use meanShift algorithm to every single frame, and the initial window of each frame is just the output window of the prior frame.

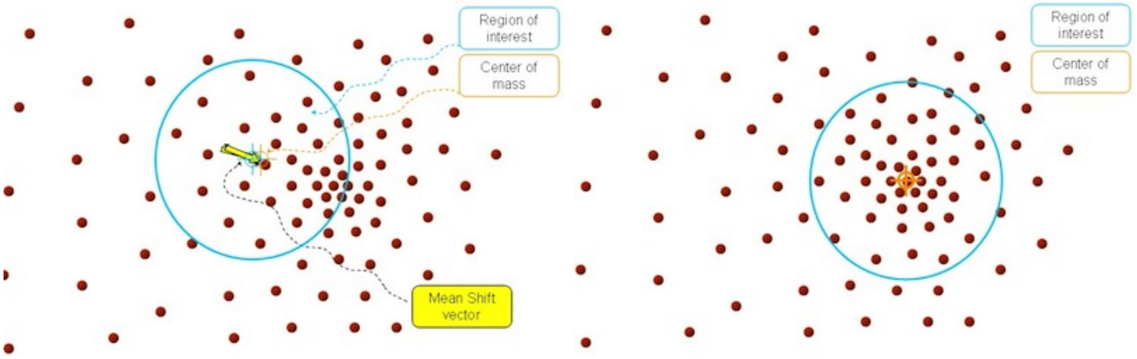


Figure 1.5: The meanshift method.

1.3 Conclusion

As a conclusion, CAMshift and HAAR Cascade are two different methods used for detection ; the first one is used for tracking the specified object detected with the second method, so even if there is a permanent movement of the object that we want to detect ,with the collaboration of this two methods the detection will be succeeded.

Chapter 2

Face recognition methods

2.1 Introduction

We have developed a near-real-time computer system that can locate and track a subject's head, and then recognize the person by comparing characteristics of the face to those of known individuals. The technology of face recognition has become mature within these few years. Systems using the face recognition, have become true in real life. In this report, we will see a comparative study of the most recent methods of face recognition. One of the approaches is using the eigenfaces method, fisherfaces method, LBPH and SIFT methods. After the implementation of the above methods, we learned the advantages and disadvantages of each approach and the difficulties of their implementation.

2.2 The Eigenfaces method

Eigenfaces are the name given to a set of eigenvectors when they are used in a computer vision problem of human face recognition. The approach of using eigenfaces for recognition was developed by Sirovich and Kirby (1987) and used by Matthew Turk and Alex Pentland in face classification. The eigenvectors are derived from the covariance matrix of the probability distribution over the high-dimensional vector space of face images.

2.2.1 Individual features

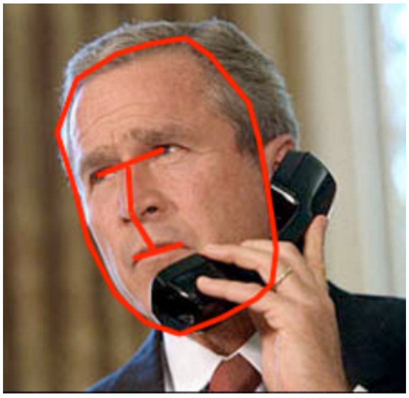
- Eyes, nose, mouth, head outline
- Position and size relationships

2.2.2 Advantages

As an appearance-based approach, eigenface recognition method has several advantages:

- Raw intensity data are used directly for learning and recognition without any significant low-level or mid-level processing.
- No knowledge of geometry and reflectance of faces is required.
- Data compression is achieved by the low-dimensional subspace representation.
- Recognition is simple and efficient compared to other matching approaches.

Figure 2.1: The Eigenfaces method principal features[5]



2.2.3 Disadvantages

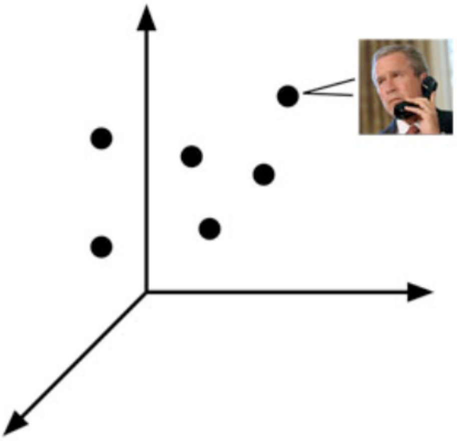
These advantages reflect the power of appearance-based approach in ease of implementation. However, the experimental results also demonstrate some serious limitations of eigenface representation method for face recognition under different conditions.

- very sensitive to scale.
- Recognition rate decreases for recognition under varying pose and illumination.
- Fragile and complex

Though the eigenfaces approach is shown to be robust when dealing with expression and glasses, these experiments were made only with frontal views. The problem can be far more difficult when there exists extreme change in pose as well as in expression and disguise. Additionally, all the face images tested in the experiments are taken with a uniform background. However, this condition may not be satisfied in most natural scenes, which will seriously deteriorate the recognition performance. In such cases, a segmentation process has to be considered.

2.2.4 The Eigenface approach

Figure 2.2: The Eigenfaces approach [5]



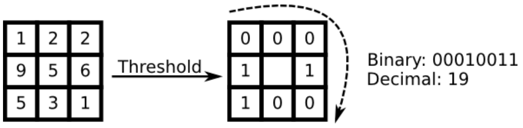
- Images are points in a vector space.
- We use PCA to reduce dimensionality of the face space.
- Compare projections onto face space to recognize faces.

2.3 The Fisherfaces method

Fisherface Concept-Differing from the Eigenface concept, the fisherface method tries to maximize the ratio of the between-class scatter versus the within-class scatter . The result of this shapes the projections so that the distances between the classes are at a maximum, while the distances between samples of the same class are at a minimum. A possible disadvantage is if the between-class scatter is large, then the within-class scatter might also still be of a relatively large value.

conclusion

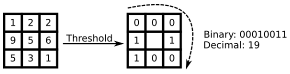
Eigenfaces and Fisherfaces take a somewhat holistic approach to face recognition. You treat your data as a vector somewhere in a high-dimensional image space. We all know high-dimensionality is bad, so a lower-dimensional subspace is identified, where (probably) useful information is preserved. The Eigenfaces approach maximizes the total scatter, which can lead to problems if the variance is generated by an external source, because components with a maximum variance over all classes aren't necessarily useful for classification. So to preserve some discriminative information we applied a Linear Discriminant Analysis and optimized as described in the Fisherfaces method. The Fisherfaces method worked great... at least for the constrained scenario we've assumed in our model. Now real life isn't perfect. You simply can't guarantee perfect light settings in your images or 10 different images of a person. So what if there's only one image for each person? Our covariance estimates for the subspace may be horribly wrong, so will the recognition. Remember the Eigenfaces method had a 96



The Fisherface approach

So in order to get good recognition rates you'll need at least 8(+-1) images for each person and the Fisherfaces method doesn't really help here. The above experiment is a 10-fold cross validated result carried. So some research concentrated on extracting local features from images. The idea is to not look at the whole image as a high-dimensional vector, but describe only local features of an object. The features you extract this way will have a low-dimensionality implicitly. But you'll soon observe the image representation we are given doesn't only suffer from illumination variations. Think of things like scale, translation or rotation in images - your local description has to be at least a bit robust against those things. the Local Binary Patterns methodology has its roots in 2D texture analysis. The basic idea of Local Binary Patterns is to summarize the local structure in an image by comparing each pixel with its neighborhood. Take a pixel as center and threshold its neighbors against. If the intensity of the center pixel is greater-equal its neighbor, then denote it with 1 and 0 if not. You'll end up with a binary number for each pixel, just like 11001111. So with 8 surrounding pixels you'll end up with $2^{8}possiblecombinations, calledLocalBinaryPatternsorsometimesreferredtoasLBPcodes.ForexampleaL$

Figure 2.3: LBPH principe[5]



2.3.1 SIFT Algorithm

Scale-invariant feature transform (SIFT) is an algorithm in computer vision to detect and describe local features in images. The algorithm was patented in the US by the University of British Columbia [7] and published by David Lowe in 1999.[8]

Applications include object recognition, robotic mapping and navigation, image stitching, 3D modeling, gesture recognition, video tracking, individual identification of wildlife and match moving.

Due to time limitation, we will not dive deep into this method in this document, we will only present some key notes that distinguish it from the other methods.

Overview

SIFT is simply an algorithm that extracts the key points that distinguish a specific object. This description, extracted from a training image, can then be used to identify the object when attempting to locate the object in a test image containing many other objects. To perform reliable recognition, it is important that the features extracted from the training image be detectable even under changes in image scale, noise and illumination. Such points usually lie on high-contrast regions of the image, such as object edges. The relative positions between these key features in the original scene shouldn't change from one image to another.

SIFT detects and uses a very large number of features from the images, which reduces the contribution of the errors caused by these local variations in the average error of all feature matching errors. This technique can robustly identify objects even among clutter and under partial occlusion, because the SIFT feature descriptor is invariant to uniform scaling, orientation, illumination changes, and partially invariant to affine distortion.[7]

How does SIFT work?

SIFT keypoints of objects are first extracted from a set of reference images[8] and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors. From the full set of matches, subsets of keypoints that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches. The determination of consistent clusters is performed rapidly by using an efficient hash table implementation of the generalized Hough transform. Each cluster of 3 or more features that agree on an object and its pose is then subject to further detailed model verification and subsequently outliers are discarded. Finally the probability that a particular set of features indicates the presence of an object is computed, given the accuracy of fit and number of probable false matches. Object matches that pass all these tests can be identified as correct with high confidence.[9]

An example of application

2.3.2

Bibliography

[1] *Wikipedia.*
https://en.wikipedia.org/wiki/Object_detection
https://en.wikipedia.org/wiki/Computer_vision

[2] P. Viola, M.J. Jones *"Robust Real-Time Face Detection, International Journal of Computer Vision"*, Vol. 57, No. 2, May 2004.

[3] *"Face Detection and Tracking: A Comparative Study Of Two Algorithms"* Sonia Mittal, Chirag Shivnani
http://csjournals.com/IJCSC/PDF7-1/11.%20Sonia.pdf

[4] *OpenCV Documentation Website*
https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html
http://docs.opencv.org/3.1.0/db/df8/tutorial_py_meanshift.html

[5] *"A comparison of Haar-like, LBP and HOG approaches to concrete and asphalt runway detection in high resolution imagery"*
http://epacis.net/jcis/PDF_JCIS/JCIS11-art.0101.pdf

[6] *"Continuously Adaptive Shift"*
http://eric-yuan.me/continuously-adaptive-shift/

[7] *U.S. Patent 6,711,293, "Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image", David Lowe's patent for the SIFT algorithm, March 23, 2004*
https://www.google.com/patents/US6711293

[8] *Lowe, David G. (1999). "Object recognition from local scale-invariant features" (PDF). Proceedings of the International Conference on Computer Vision. 2. pp. 1150–1157. doi:10.1109/ICCV.1999.790410.*
http://www.cs.ubc.ca/~lowe/papers/iccv99.pdf

[9] *Lowe, David G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints". International Journal of Computer Vision. 60 (2): 91–110. doi:10.1023/B:VISI.0000029664.99615.94.* http://citeseer.ist.psu.edu/lowe04distinctive.html