Feature Extraction from an Image and Color Detection Using OpenCV

CPE-695-A

(Applied Machine Learning)

Submitted By:

Divyesh Mehta

Ashutosh Gajankush

Submitted To:

Prof: Rong Duan

Date: 5th May 2016

Table of Contents

List of Figures

CHAPTER I

INTRODUCTION

1.1 Motivation

Feature extraction and matching is at the base of many computer vision problems, such as object recognition or structure from motion. Current methods for assessing the performance of popular image matching algorithms are presented and rely on costly descriptors for detection and matching. Specifically, the method assesses the type of images under which each of the algorithms reviewed herein perform to its maximum or highest efficiency. The efficiency is measured in terms of the number of matches founds by the algorithm and the number of type I and type II errors encountered when the algorithm is tested against a specific pair of images. Current comparative studies asses the performance of the algorithms based on the results obtained in different criteria such as speed, sensitivity, occlusion, and others.

1.2 Feature Extraction

In machine learning, pattern recognition and in image processing, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is related to dimensionality reduction.

When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g. the same measurement in both feet and meters, or the repetitiveness of images presented as pixels), then it can be transformed into a reduced set of features (also named a features vector). This process is called feature extraction. The extracted features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data.

In machine learning and pattern recognition, a feature is a measurable property of a phenomenon being observed. Choosing discriminating and independent features is key to any pattern recognition algorithm being successful in classification. For instance, if our goal is to detect a cow in an image while any pixel of the image can be seen as a feature, not all of these are informative and useful for our goal. In real applications usually tens of thousands of features are measured while only a very small percentage of them carry useful information towards our learning goal. Therefore, we usually need an algorithm that compress our feature vector and reduce its dimension. Two groups of methods which can be used for dimensionality reduction are: 1) Feature extraction methods where apply a transformation on the original feature vector to reduce its dimension from d to m. 2) Feature selection methods that select a small subset of original features. In this work, we want to compare the linear discriminant analysis (LDA) which is a traditional feature extraction method with a forward selection based method (which is an instance of the feature selection algorithms) and find under which conditions, one of these algorithms works better.

1.2 Feature Construction

Feature construction is one of the key steps in the data analysis process, largely conditioning the success of any subsequent statistics or machine learning endeavor. In particular, one should beware of not losing information at the feature construction stage. It may be a good idea to add the raw features to the preprocessed data or at least to compare the performances obtained with either representation. We argue that it is always better to err on the side of being too inclusive rather than risking it to discard useful information. The medical diagnosis example that we have used before illustrates this point. Many factors might influence the health status of a patient. To the usual clinical variables (temperature, blood pressure, glucose level, weight, height, etc.), one might want to add diet information (low fat, low carbonate, etc.), family history, or even weather conditions. Adding all those features seems reasonable but it comes at a price: it increases the dimensionality of the patterns and thereby immerses the relevant information into a sea of possibly irrelevant, noisy or redundant features.

1.3 Feature selection

We are decomposing the problem of feature extraction in two steps: feature construction, briefly reviewed in the previous section, and feature selection, to which we are now directing our attention. Although feature selection is primarily performed to select relevant and informative features, it can have other motivations, including: 1. general data reduction, to limit storage requirements and increase algorithm speed; 2. feature set reduction, to save resources in the next round of data collection or during utilization; 3. performance improvement, to gain in predictive accuracy; 4. data understanding, to gain knowledge about the process that generated the data or simply visualize the data

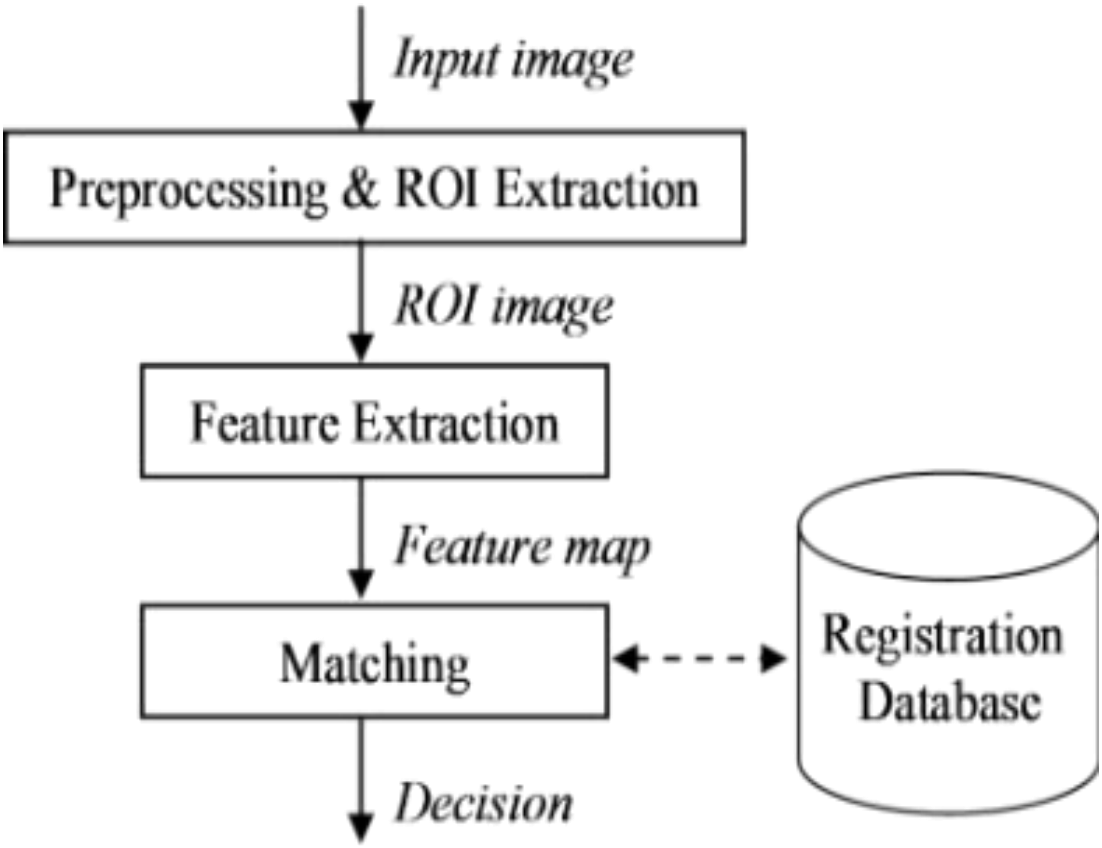

Fig 1: General Block Diagram of Feature Extraction.

CHAPTER II

HOG Algorithm

2.1 Introduction to HOG

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

Navneet Dalal and Bill Triggs, researchers for the French National Institute for Research in Computer Science and Automation (INRIA), first described HOG descriptors at the 2005 Conference on Computer Vision and Pattern Recognition (CVPR).

2.2 Implementation of the HOG

Implementation of the HOG descriptor algorithm is as follows:

1. Divide the image into small connected regions called cells, and for each cell compute a histogram of gradient directions or edge orientations for the pixels within the cell.
2. Discretize each cell into angular bins according to the gradient orientation.
3. Each cell's pixel contributes weighted gradient to its corresponding angular bin.
4. Groups of adjacent cells are considered as spatial regions called blocks. The grouping of cells into a block is the basis for grouping and normalization of histograms.
5. Normalized group of histograms represents the block histogram. The set of these block histograms represents the descriptor.

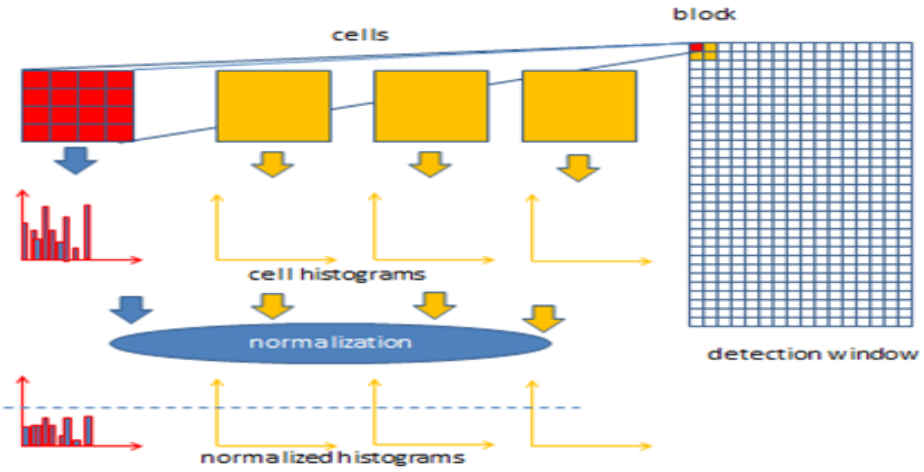

Fig 2. HOG Algorithm Implementation Scheme

2.3 Computation of HOG

Computation of the HOG descriptor requires the following basic configuration parameters:

- Masks to compute derivatives and gradients
- Geometry of splitting an image into cells and grouping cells into a block
- Block overlapping
- Normalization parameters

The recommended values for the HOG parameters are:

- 1D centered derivative mask [-1, 0, +1]
- Detection window size is 64x128
- Cell size is 8x8
- Block size is 16x16 (2x2 cells)

2.4 Object detection using Color Detection

Object detection and segmentation is the most important and challenging fundamental task of computer vision. It is a critical part in many applications such as image search, scene understanding, etc. However it is still an open problem due to the variety and complexity of object classes and backgrounds.

The easiest way to detect and segment an object from an image is the color based methods . The object and the background should have a significant color difference in order to successfully segment objects using color based methods.

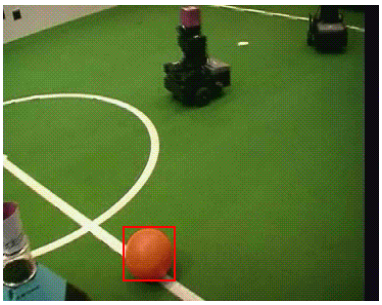

Fig 3: Image Showing Pink Color Detection

2.5 Real Time Object Tracking

Tracking moving objects based on color information is more robust than systems utilizing motion cues. In order to maintain the lock on the object as the surrounding conditions vary, the color model needs to be adapted in real-time.

For our project we are using Laptop camera for detecting the green color in real-Time. The upper and the lower limit of the color is given and based on that the green color is detected. The color is tracked and a nice red track is seen in the real time.

CHAPTER III

TECHNOLOGIES

3.1 OpenCV:

OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android.

Fig 4: OpenCV Logo

OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 9 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

3.2 Python Programming:

Python is an interpreted, object-oriented programming language similar to PERL, which has gained popularity because of its clear syntax and readability.

Fig.5 Python Logo

Python is said to be relatively easy to learn and portable, meaning its statements can be interpreted in a number of operating systems.

CHAPTER IV

APPROACH AND APPLICATIONS

4.1 APPROACH

4.1.1 HOG Approach:

General algorithm for training an object detector using Histogram of Oriented Gradients is as follows:-

- Sample P positive samples from your training data of the object(s) you want to detect and extract HOG descriptors from these samples.

- Sample N negative samples from a negative training set that does not contain any of the objects you want to detect and extract HOG descriptors from these samples as well. In practice N >> P.

- Train a Linear Support Vector Machine on your positive and negative samples.

- Apply hard-negative mining. For each image and each possible scale of each image in your negative training set, apply the sliding window technique and slide your window across the image. At each window compute your HOG descriptors and apply your classifier. If your classifier (incorrectly) classifies a given window as an object (and it will, there will absolutely be false-positives), record the feature vector associated with the false-positive patch along with the probability of the classification. This approach is called hard-negative mining.

- Take the false-positive samples found during the hard-negative mining stage, sort them by their confidence (i.e. probability) and re-train your classifier using these hard-negative samples. (Note: You can iteratively apply steps 4-5, but in practice one stage of hard-negative mining usually [not always] tends to be enough. The gains in accuracy on subsequent runs of hard-negative mining tend to be minimal.)

- Your classifier is now trained and can be applied to your test dataset. Again, just like in Step 4, for each image in your test set, and for each

scale of the image, apply the sliding window technique. At each window extract HOG descriptors and apply your classifier. If your classifier detects an object with sufficiently large probability, record the bounding box of the window. After you have finished scanning the image, apply non-maximum suppression to remove redundant and overlapping bounding boxes.

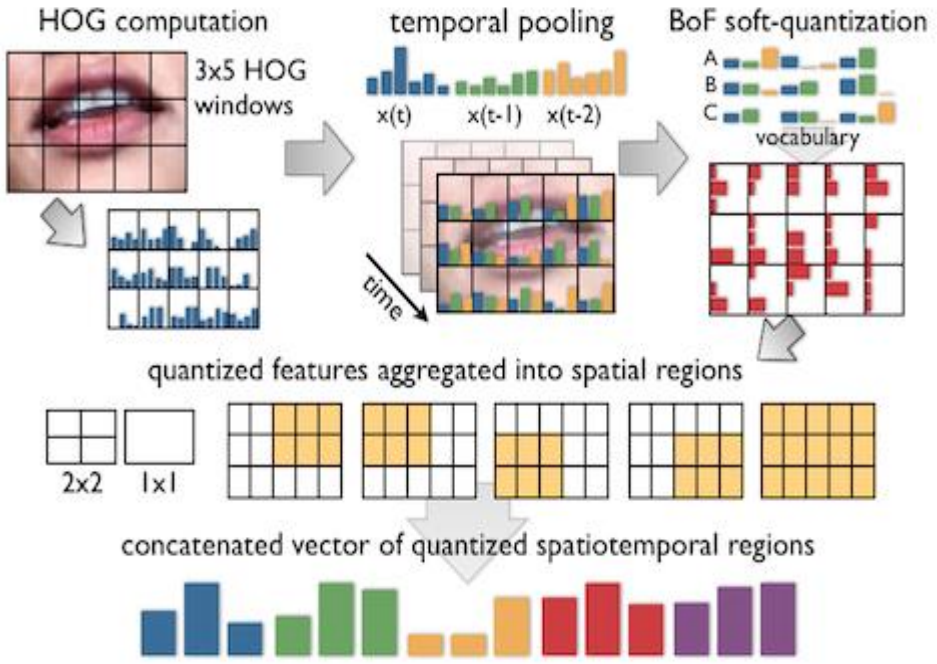

Fig 6: HOG Algorithm Steps Overview

4.1.2 Approach for Color detection

OpenCV usually captures images and videos in 8-bit, unsigned integer, BGR format. In other words, captured images can be considered as 3 matrices; BLUE, GREEN and RED (hence the name BGR) with integer values ranges from 0 to 255.

In the Fig 7, each small box represents a pixel of the image. In real images, these pixels are so small that human eye cannot differentiate.

Usually, one can think that BGR color space is more suitable for color based

segmentation. But HSV color space is the most suitable color space for color based image segmentation.

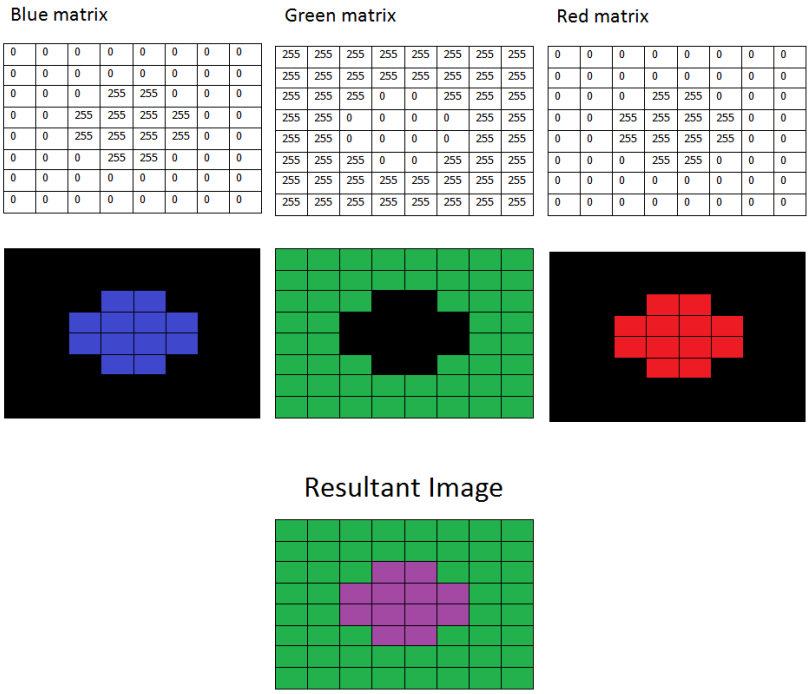

Fig 7 How BGR image is formed

HSV color space is also consists of 3 matrices, HUE, SATURATION and VALUE. In OpenCV, value range for HUE, SATURATION and VALUE are respectively 0-179, 0-255 and 0-255.

HUE represents the color, SATURATION represents the amount to which that respective color is mixed with white and VALUE represents the amount to which that respective color is mixed with black.

Hue values of basic colors

- Orange 0-22
- Yellow 22- 38
- Green 38-75
- Blue 75-130
- Violet 130-160
- Red 160-179

These are approximate values.

4.2 Applications:

Feature Extraction can have many applications and can be used in various fields. Few examples are as follows:-

Robotics

- This technique can be further utilized in the Robotics field whereby we can train Robots to perform particular actions based on the feature extracted.

- This Robot can be used in the areas where it is not possible for human to go. Therefore training robots to do a particular task based on the features extracted can be very useful.

Fig 8: Robot Training

➢ Security

- Based on the real time images, System can detect any intruder entering the premises or any malfunction in any instruments in a lab and can alert the Security.

Fig.9: Security Camera

CHAPTER V

CONCLUSION

5.1 Conclusion:

In this project, we successfully implemented Human Detection, Color based object detection and the real time tracking of an Object based on Color Detection. We also learned and implemented HOG algorithm. It also gave us experience in using OpenCV and Python.

5.2 Future Scope:

- Object Detection: All the objects in an image can be detected and labelled accordingly. This will help us to interpret image fully.
- Real-time Object Detection: All the Objects can be traced in real time and will be very useful in Automation System and Security.

References

1. https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

2. http://docs.opencv.org/trunk/d8/db1/group__datasets__pd.html#gsc.tab=0

3.http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours#findcontours

4.http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=drawcontours#drawcontours