Orientation Robust Object Detection Using Histogram of Oriented Gradients

S. M. Amirul Islam

2013-1-60-048

Md. Abdullah-Al-Kafi

2013-1-60-033

A project submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering

Department of Computer Science and Engineering
East West University
Dhaka-1212, Bangladesh

December, 2017

Declaration

I, hereby, declare that the work presented in this project is the outcome of the investigation performed by me under the supervision of Dr. Taskeed Jabid , Assistant Professor, Department of Computer Science and Engineering, East West University. I also declare that no part of this project has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned	Signature
....	....
(Dr. Taskeed Jabid)	(**S. M. Amirul Islam**)
Supervisor	
	Signature
	....
	(**Md. Abdullah-Al-Kafi**)

Letter of Acceptance

This Project Report entitled *Orientation Robust Object Detection Using Histogram of Oriented Gradients* submitted by S. M. Amirul Islam (ID: 2013-1-60-048) and Abdullah-Al-Kafi (ID: 2013-1-60-033) to the Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh is accepted by the department in partial fulfillment of requirements for the Award of the Degree of Bachelor of Science and Engineering on December, 2017.

Supervisor

. . . .

(Dr. Taskeed Jabid
Assistant Professor, Department of Computer Science and Engineering
East West University, Dhaka, Bangladesh.

Chairperson

. . . .

(Dr. Ahmed Wasif Reza)
Chairperson and Associate Professor,
Department of Computer Science and Engineering, East West University.

Abstract

We study orientation robust object detection using the HOG feature set. We show that this method provides reasonably well accuracy for detecting objects with varying angle, poses and distance from the viewing plane. We calculate gradient magnitude and orientation of individual cells of the input images and get gradient vector from it. Then we divide the gradients vectors in predetermined bins depending on it's orientation. After that, we normalize the image blocks to get normalized vector. Concatenating all the normalized vectors gives our final feature vector. Finally we give the feature vector to a SVM to train our detector. Once the detector is trained, it is ready for testing. Our testing results show that the detector can detect objects with recall rate of 83% and precision rate of 97%.

Acknowledgments

As it is true for everyone, we have also arrived at this point of achieving a goal in our life through various interactions with and help from other people. We want to thank everyone who have supported us to complete this project. This work was carried out in the Department of Computer Science and Engineering at East West University, Bangladesh.

First of all, we would like to express my deepest gratitude to the Almighty for His blessings on us. Next, our special thanks go to our supervisor, "Dr. Taskeed Jabid", who gave us this opportunity, initiated us into the field of "Object Detection", and without whom this work would not have been possible. His encouragements, visionary and thoughtful comments and suggestions, unforgettable support at every stage of our BS.c study were simply appreciating and essential.

Last but not the least, We would like to thank our parents for their unending support, encouragement and prayers.

There are numerous other people too who have shown us their constant support and friendship in various ways, directly or indirectly related to our academic life. We will remember them in our heart and hope to find a more appropriate place to acknowledge them in the future.

S. M. Amirul Islam
December, 2017

Abdullah-Al-Kafi
December, 2017

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Object Detection is widely used in academic and industrial applications. Real life applications of object detection includes face detection, pedestrian detection and counting, product detection in manufacturing line, vehicle detection, security and intelligence, etc. As a sub-discipline of Computer Vision [1] and Image Processing. Object Detection has improved a lot in the last two decades. At present, several techniques exist to model and implement systems that can detect a specific object. Histogram of Oriented Gradients (HOG) has become a very popular approach to make object detector systems.

In this project, we implement a HOG based detector to identify orientation robust objects of varying shape, textures and viewing angle.

1.1 Motivation

It is a well-known fact that HOG based object detectors perform very well for detecting rigid objects. It delivers very good amount of accuracy from a small dataset. Unlike methods like CNN, this method does not take huge computational power, large dataset and long time to train. This makes HOG very convenient for applications where the system must perform under resource, time and cost constraints. We wanted to exploit these advantages of HOG and use it for detecting objects with different orientations.

1.2 Objective

Our objective is to show the validity of HOG as a Computer Vision technology that can perform well in real life applications. Many real world objects are semi-rigid in nature and they have different orientations. We will show that we can get considerably good amount of accuracy while using HOG for detecting objects with different orientations.

1.3 System Procedure

Our approach to implement the system revolves around several steps. First we use a set of training images to extract the feature set. [2] Then we feed the extracted features to a structural SVM. The SVM uses the features to train a detector for the given object. [3] Once the detector is trained, we can start testing images to identify our desired object.

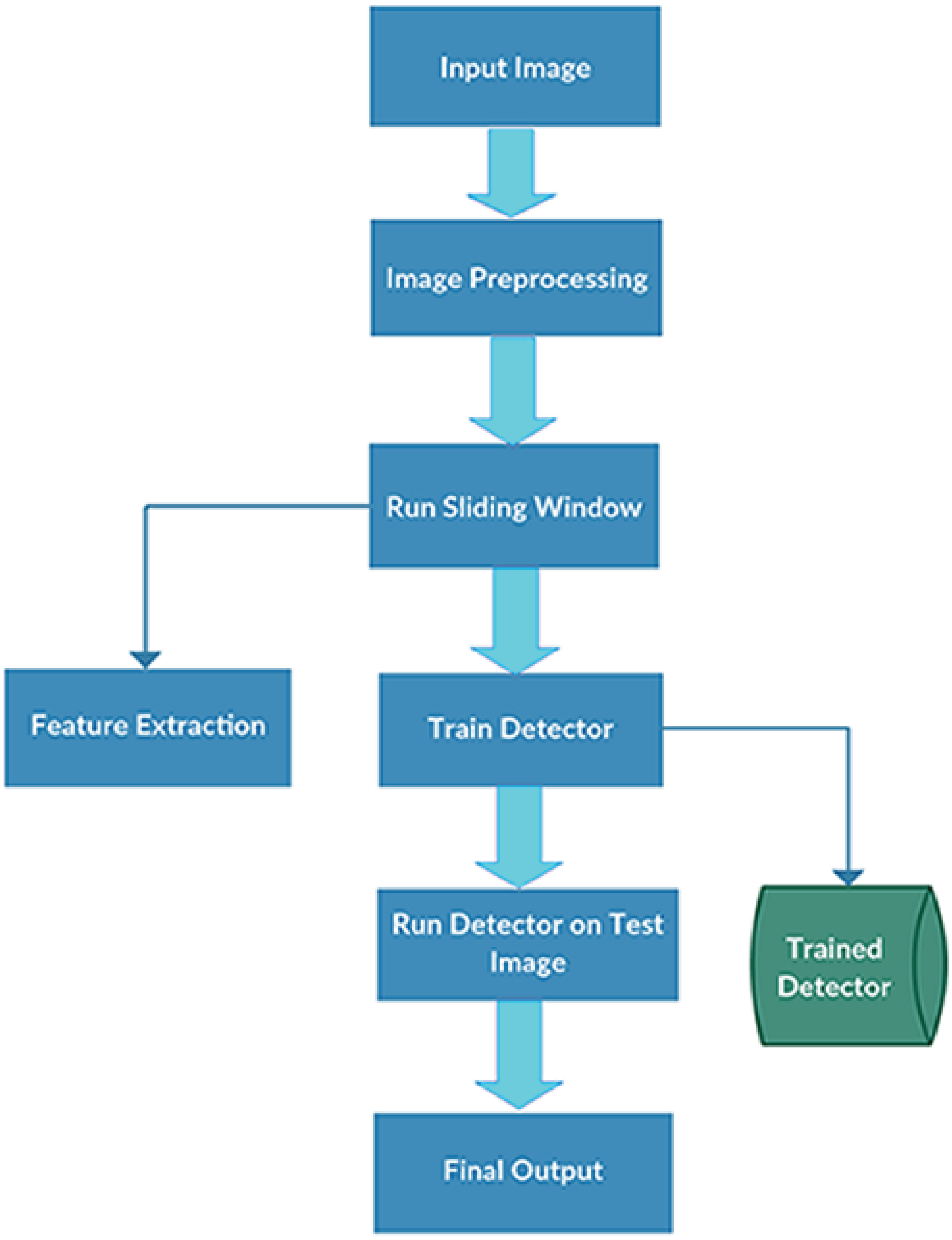

Figure 1.1: System Procedure

Chapter 2

Literature Review

The amount of literature in Object Detection is extensive. From the late 90's there have been many research papers that propose different approaches to object detection. Here we will discuss about only a few of the previous literary works that are relevant to our own work.

2.1 Viola-Jones Face Detector

Viola and Jones published their seminal work in 2001. [4] It was the first object detection framework that provided competitive detection rates in real-time. The primary motivation for this was face detection. It had high detection rate and very low false positive rate for face detection. This method is suitable for real time application, it can process two frames per second. The algorithm has four stages:

- Haar feature selection.
- Creating an integral image.
- Adaboost training.
- Cascading classifiers.

2.1.1 Haar Features

The haar features [5] can be extracted by calculating the difference in brightness between the white and black rectangles over a specific area. Viola and Jones used a two rectangle feature, more rectangles are also possible. Each feature is related to a special location in the sub-window.

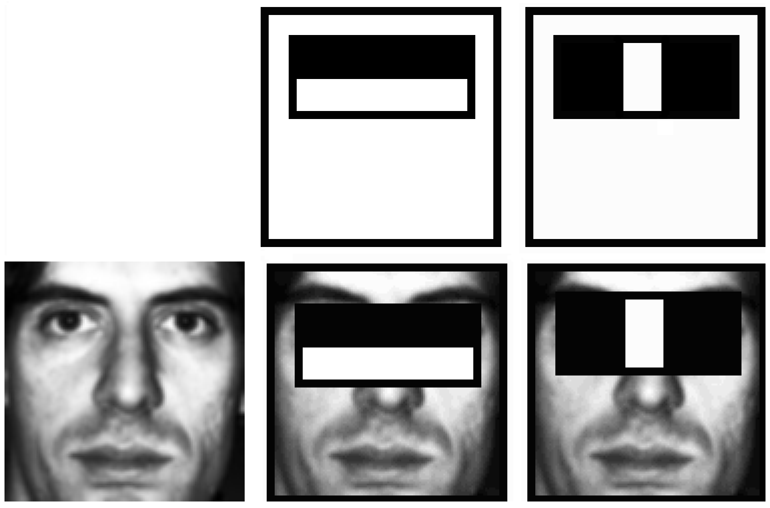

Figure 2.1: Haar Feature Extraction

2.1.2 Integral Image

Integral image is an image representation techniques that can evaluate rectangular features in constant time, which gives them a considerable speed advantage over more sophisticated alternative features. It computes the sum of values in a rectangle subset of a grid.

2.1.3 Adaboost Training

The Adaboost or Adaptive Boost is a very popular boosting algorithm. It takes the output of 'weak classifiers' and combines them to provide a more accurate 'strong classifier'.

2.1.4 Cascade Structure

An important part of the Viola-Jones detector is the cascade. It uses the boosted classifier that keeps almost all positive examples while rejecting most of the negative sub-windows. Most of the sub-windows are rejected in the early stage of the detector. Classifying a sub-window forms a degenerate decision tree, which is called a cascade. The cascade structure has an impact on the training process. Viola-Jones detector needs a lot of negative examples. To handle large amount of negative training data, Viola and Jones used a bootstrap process. A threshold was manually chosen at each node. A partial classifier was used to find unelected negative examples to train the next node. Each of the nodes are trained independently. However, recent works have showed that it might be beneficial not to completely separate the training process of different nodes.

2.2 Felzenwalb's Deformable Parts Model

Felzenwalb's object detection system is based on mixtures of multiscale deformable part models. [6]

The system is able to represent highly variable object classes and achieves state-of-the-art results in the PASCAL object detection challenges. While deformable part models have become quite popular, their value had not been demonstrated on difficult benchmarks such as the PASCAL datasets. The system relies on new methods for discriminative training with partially labeled data. Felzenwalb combines a margin sensitive approach for data-mining hard negative examples with a formalism known as latent SVM. A latent SVM is a reformulation of MI-SVM in terms of latent variables. A latent

SVM is semi-convex and the training problem becomes convex once latent information is specified for the positive examples. This leads to an iterative training algorithm that alternates between fixing latent values for positive examples and optimizing the latent SVM objective function.

Deformable part models such as pictorial structures provide an elegant framework for object detection. Yet it has been difficult to establish their value in practice. On difficult datasets deformable part models are often outperformed by simpler models such as rigid templates or bag-of-features. While deformable models can capture significant variations in appearance, a single deformable model is often not expressive enough to represent a rich object category.

In Felzenwalb's model, image is partitioned into blocks at multiple scales and compute histogram of gradient orientations in each block. Then train a template using a linear support vector machine. At test time, convolve feature map with template. Find local maxima of response. For multi-scale detection, repeat over multiple levels of a HOG pyramid. Single rigid template usually is not enough to represent a category. Many objects (e.g. humans) are articulated, or have parts that can vary in configuration. Many object categories look very different from different viewpoints, or from instance to instance.

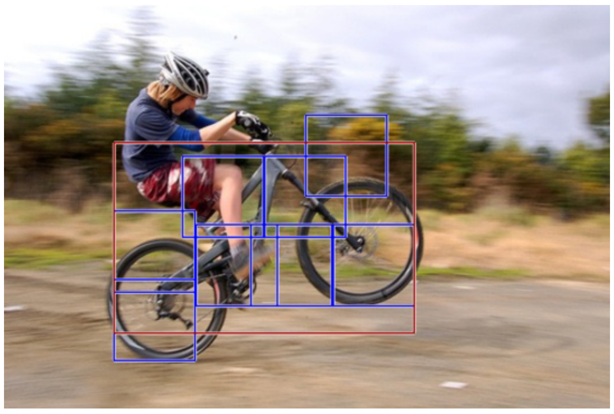

Figure 2.2: Bicycle Detection using Felzenswalb's Deformable Parts Model

2.2.1 Object Hypothesis

Multiscale Model. The resolution of part filters is twice the resolution of the root. The score of a hypothesis is the sum of filter scores minus the sum of deformation costs.

Detection. Define the score of each root filter location as the score given the best part placements.

Efficient Computation. Generalized distance transforms. For each "default" part location, find the score of the "best" displacement.

Training. Training data consists of images with labeled bounding boxes. Need to learn the filters and deformation parameters.

2.3 Dalal and Triggs' HOG for Human Detection

HOG is popular detector. This has been used lot in human detection. Detecting humans in images is a challenging task owing to their variable appearance and the wide range of poses that they can adopt. The first need is a robust feature set that allows the human form to be discriminated cleanly, even in cluttered backgrounds under difficult illumination. Locally normalized Histogram of Oriented Gradient (HOG) descriptors provide excellent performance relative to other existing feature sets.[7] Because local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions.

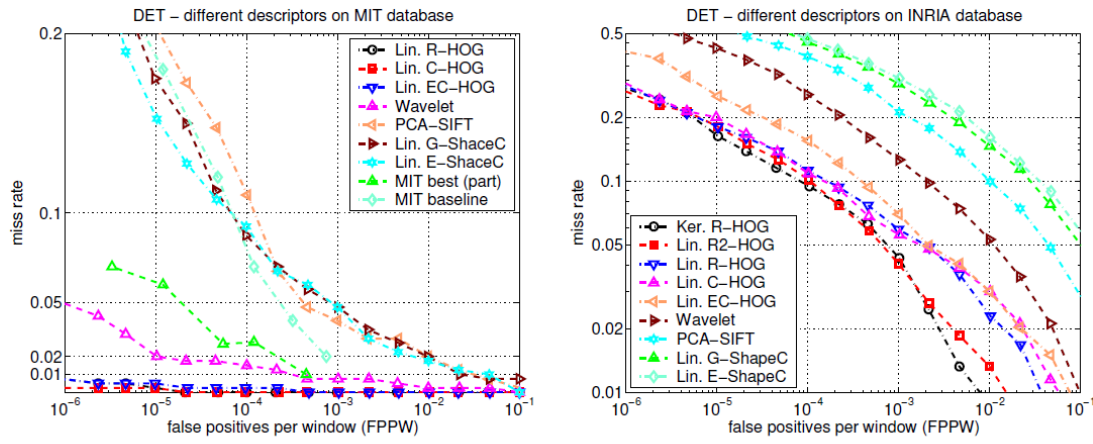

Figure 2.3: The performance of selected detectors on (left) MIT and (right) INRIA data sets.

2.3.1 Dataset

This detector is tasted on two different data sets. The first is the well-established MIT pedestrian database containing 509 training and 200 test images of pedestrians in city scenes (plus left-right reflections of these). It contains only front or back views with a relatively limited range of poses. Their best detectors give essentially perfect results on this data set, so they produced a new and significantly more challenging data set, 'INRIA', containing 1805 64x128 images of humans cropped from a varied set of personal photos.

2.3.2 Methodology

The method is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. The basic idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions. In practice this is implemented by dividing the image window into small spatial regions (cells), for each cell accumulating a local 1-D histogram of gradient directions or edge orientations over the pixels of the cell. The combined histogram entries from the representation. For better invariance to illumination, shadowing, etc., it is also useful to contrast normalize the local responses before using them. This can be done by accumulating a measure of local histogram ”energy” over somewhat larger spatial regions (blocks) and using the results to normalize all of the cells in the block. The full detector has an even lower false positive rate owing to non-maximum suppression.

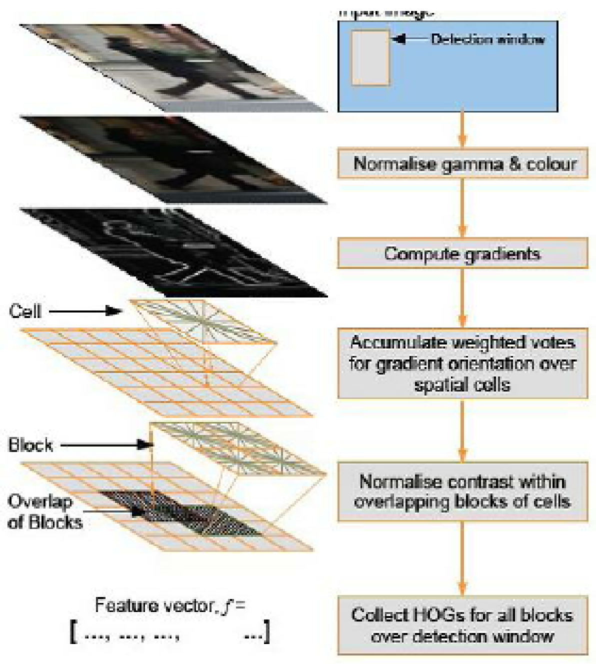
Detection window
Normalise gamma & colour
Compute gradients
Cell
Accumulate weighted votes for gradient orientation over spatial cells
Block
Overlap of Blocks
Normalise contrast within overlapping blocks of cells
Feature vector, f =
[..., ..., ..., ...]
Collect HOGs for all blocks over detection window

Figure 2.4: An overview of feature extraction and object detection chain

1239 of the images was selected as positive training examples, together with their left-right reflections (2478 images in all). A fixed set of 12180 patches sampled randomly from 1218 person-free training photos provided the initial negative set. For each detector and parameter combination a preliminary detector is trained and the 1218 negative training photos are searched exhaustively for false positives ('hard examples'). The method is then re-trained using this augmented set (initial 12180 + hard examples) to produce the final detector. The set of hard examples is subsampled if necessary, so that the descriptors of the final training set fit into 1.7 Gb of RAM for SVM training. This retraining process significantly improves the performance of each detector (by 5% at 10-4 False Positives per Window tested (FPPW) for default detector), but additional rounds of retraining make little difference so this is not used. To quantify detector performance

Detection Error Tradeoff (DET) curves are plotted on a log-log scale, i.e. miss rate (1-Recall or FalseNeg/TruePos+FalseNeg) versus FPPW. Lower values are better. DET plots are used extensively in speech and in NIST evaluations. They present the same information as Receiver Operating Characteristics (ROC's) but allow small probabilities to be distinguished more easily. They often use miss rate at 10-4 FPPW as a reference point for results.

This is arbitrary but no more so than, e.g. Area Under ROC. In a multiscale detector it corresponds to a raw error rate of about 0:8 false positives per 640x480 image tested. (The full detector has an even lower false positive rate owing to non-maximum suppression). DET curves are usually quite shallow so even very small improvements in miss rate are equivalent to large gains in FPPW at constant miss rate. For example, for default detector at 1e-4 FPPW, every 1% absolute (9% relative) reduction in miss rate is equivalent to reducing the FPPW at constant miss rate by a factor of 1.57.

2.3.3 Implementation

The implementation is done in several steps. We briefly discuss them in the sub-sections of this section.

2.3.3.1 Gamma/Color Normalization

There is an evaluation for several input pixel representations including grayscale, RGB and LAB color spaces optionally with power law (gamma) equalization. These normalizations have only a modest effect on performance, perhaps because the subsequent descriptor normalization achieves similar results. Color information is used when available. RGB and LAB color spaces give comparable results, but restricting to grayscale reduces performance by 1.5% at 10-4 FPPW. Square root gamma compression of each color channel improves performance at low FPPW (by 1% at 10-4 FPPW) but log compression is too strong and worsens it by 2% at 10-4 FPPW.

2.3.3.2 Gradient Computation

Detector performance is sensitive to the way in which gradients are computed, but the simplest scheme turns out to be the best. We tested gradients computed using Gaussian smoothing followed by one of several discrete derivative masks. Several smoothing scales were tested including $\sigma = 0$ (none). Masks tested included various 1-D point derivatives and cubic corrected as well as Sobel masks. Simple 1-D masks at $\sigma = 0$ works best. Using larger masks always seems to decrease performance, and smoothing damages it significantly. Using uncentered derivative masks also decreases performance presumably because orientation estimation suffers as a result of the x and y filters being based at different centers. For color images, separate gradients are calculated for each color channel, and take the one with the largest norm as the pixel's gradient vector.

2.3.3.3 Spatial / Orientation Binning

The next step is the fundamental nonlinearity of the descriptor. Each pixel calculates a weighted vote for an edge orientation histogram channel based on the orientation of the gradient element centered on it, and the votes are accumulated into orientation bins over local spatial regions that are called cells. Cells can be either rectangular or radial. To reduce aliasing, votes are interpolated bilinearly between the neighboring bin centers in both orientation and position. The vote is a function of the gradient magnitude at the pixel, either the magnitude itself, its square, its square root, or a clipped form of the magnitude representing soft presence/absence of an edge at the pixel. In practice, using the magnitude itself gives the best results. Taking the square root reduces performance slightly, while using binary edge presence voting decreases it significantly.

2.3.3.4 Normalization and Descriptor Blocks

Gradient strengths vary over a wide range owing to local variations in illumination and foreground-background contrast, so effective local contrast normalization turns out to be essential for good performance. A number of different normalization schemes is evaluated. Most of them are based on grouping cells into larger spatial blocks and contrast normalizing each block separately. The final descriptor is then the vector of all components of the normalized cell responses from all of the blocks in the detection window.

2.3.3.5 Detection Window

64x128 detection window includes about 16 pixels of margin around the person on all four sides.

2.3.3.6 Classifier

By default a soft (C=0.01) linear SVM [8] trained with SVMLight (slightly modified to reduce memory usage for problems with large dense descriptor vectors).

2.4 Max Margin Object Detection

Max margin object detection is a comparatively new object detection technique. It was proposed by David E. King in his research paper published in 2005. [9] In the paper he proposes a new method for object detection in images. The method does not perform sub-sampling over images, rather it does optimization over all sub windows. MMOD can be used to improve any object detection method that has linear learned parameters. The paper shows that a HOG filter that is learned through MMOD can outperform state-of-the-art deformable parts model on face detection dataset.

2.5 Neural Network Based Object Detection

Artificial neural network is a computing system inspired by the biological neural networks that constitute animal brains. Such systems learn to do tasks by considering examples, generally without task specific programming. For example, in image recognition, they might learn to identify images that contain cars by analyzing example images that have been manually labeled as "car" or "no car" and using the analytic results to identify cars in other images. They have found most use in applications difficult to express in a traditional computer algorithm using rule-based programming.

An ANN is based on a collection of connected units called artificial neurons. Each connection (synapse) between neurons can transmit a signal to another neuron. The receiving (postsynaptic) neuron can process the signal(s) and then signal downstream neurons connected to it.

An artificial neuron may have state, generally represented by real numbers, typically between 0 and 1 Neurons and synapses may also have a weight that varies as learning proceeds, which can increase or decrease the strength of the signal that it sends downstream. Further, they may have a threshold such that only if the aggregate signal is below (or above) that level is the downstream signal sent.

Typically neurons are organized in layers. Different layers may preform different kinds of transformations on their inputs. Signals travel from the first (input), to the last (output) layer, possibly after traversing the layers multiple times.

An artificial neuron is a device with many inputs and one output. The neurons has two modes of operation; the training mode and the using mode. In the training mode the neurons can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.

The original goal of the neural network approach is to solve problems in the same way that a human brain would. It has been used widely in computer vision, speech recognition, social network filtering, image processing, object detection and many other domain.

2.5.1 CNN

Convolutional neural network is a class of deep, feed forward artificial neural network that have successfully been applied to analyzing visual imagery. CNNs use a variation of multilayer perceptions designed to require minimal preprocessing.

Convolutional networks were inspired by biological processes in which the connectivity pattern between neurons is inspired by the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

2.5.1.1 Working Procedyre of CNN

In traditional feed-foward CNNs, we have training data where each element consists of a feature vector that we input to the NN in the "input layer," so with image recognition, we could just have each pixel be one input. Those are our feature vectors. Alternatively, we could manually create other – likely smaller – feature vectors.

A simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. We use three main types of layers to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer (exactly as seen in regular Neural Networks). These layers are stacked to form a full ConvNet architecture. The convolution and sub-sampling layers in a CNN works to extract features from their input. These features are then given to the next hidden layer to extract still more complex features, or are directly given to a standard classifier to output the final prediction (usually a Softmax, but also SVM [**?**] or any other can be used). In the context of image recognition, these features are images treats, like stroke patterns in the lower layers and object parts in the upper layers.

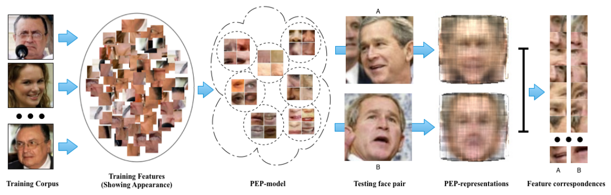

Figure 2.5: Feature extraction and mapping in CNN

In natural images these features tend to be the same at all locations. Recognizing a certain stroke pattern in the middle of the images will be as useful as recognizing it close to the borders.

After the replication (the "convolution" step) we add a sub-sample step, which can be implemented in many ways, but is nothing more than a sub-sample. The subsequent convolution and sub-sampling steps are computed over features extracted in the previous layer, instead of the raw pixels of the original image.

The convent architecture ensures that the learned "filters" produce strongest response to a spatially local input pattern.

2.5.1.2 Disadvantages of CNN

- High computational cost.
- Without a good GPU they are quite slow to train (for complex tasks).
- They need to use a lot of training data.

2.5.2 TensorFlow

TensorFlow is an open source software library for numerical computation using data flow graphs. [10] Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

It is a symbolic math library, and also used for machine learning applications such as neural networks. It also can be used with CNN in image processing or other domain.

Chapter 3
Proposed System Description

In our system we trained a semi-rigid object detector. A rigid object is a solid object in which deformity is close to zero. The distance between two given points in a rigid object remains same in time regardless external force exerted on it. On the other hand semi-rigid object is in which deformity is slightly more than rigid object. Cat's face is a good example of semi-rigid object. We trained our object detector to detect cat faces in images.

We used Histogram of Oriented gradients (HOG) to extract feature from training data and train with a SVM (Support Vector Machine) to build a detector.

3.1 Object Detection With HOG

To detect something in an image is tough work owing to their variable appearance and the wide range of poses that they can adopt. HOG is a well-known detector for detecting rigid object. Cat's face is a semi-rigid object. Cat's face can be found in various angles. Also there are various types of cat with various shape and color. In term of shape, ears, nose and whole face shape are important factors. HOG deals with them smoothly. Also with HOG slight variant in angled cat faces can be detected efficiently.

Locally normalized Histogram of Oriented gradient descriptor provide better performance relative to other existing feature sets.

3.2 Methodologies

To detect a cat's face we build our system to make histograms on data set then extract feature from them. But before feature extraction we did some image preprocessing. Then by using this feature we trained a SVM tool to detect cat's face. After training we used sliding window technique on the testing images to identify and isolate cat face in an image. It scans image with a fixed sized rectangular window. Then extracting features of that window and applying SVM to classify that the rectangular window bounds a cat's face. This process is repeated on successively scaled copies of the image so that cat's faces can be detected at any size. The sliding window technique is also used to extract right features from right bounding box for training.

At last non-maximal suppression is applied on the testing data to remove multiple detection and to find optimal detection of the same face.

3.3 Image Preprocessing

The training and testing data are preprocessed before extracting features. First we change the image to grayscale and enlarged the images by some factors. Then we down sample the images with image pyramid. Then we take the left-right mirror of the images. After mirroring the feature extraction is done.

3.3.1 Image Pyramid

At first we load the data listed in XML files. XML files list the images in dataset and also contain the positions of the face boxes. Face boxes are the rectangular boxes of faces from which the features are extracted. Then we enlarge the images with a factor of n (n=1, 2, 3). We do this because it allows us to detect smaller faces. In our work we enlarged images with a factor of two.

Pyramid, or pyramid representation, is a type of multi-scale signal representation developed by the computer vision, image processing and signal processing communities, in which a signal or an image is subject to repeated smoothing and subsampling. Pyramid representation is a predecessor to scale-space representation and multiresolution analysis. [11]

In image processing an image pyramid is a collection of images - all arising from a single original image- that successively downsampled or upsampled until some desired stopping point is reached.

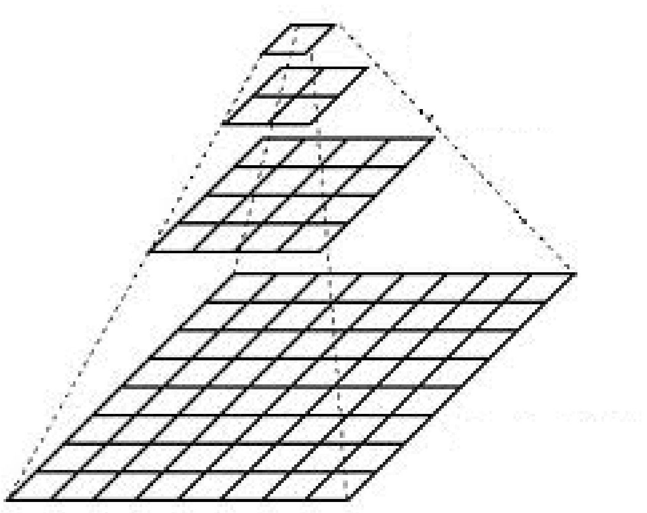

Figure 3.1: Pyramid Up

Utilizing an image pyramid allows us to find objects in images at different scales of an image. At the bottom or at the top we have the original image at its original size (in terms of width and height) and each layer the image is resized or subsampled. The image is progressively subsampled until some stopping criterion is met, which is normally minimum size has been reached and no further subsampling needs to take place.

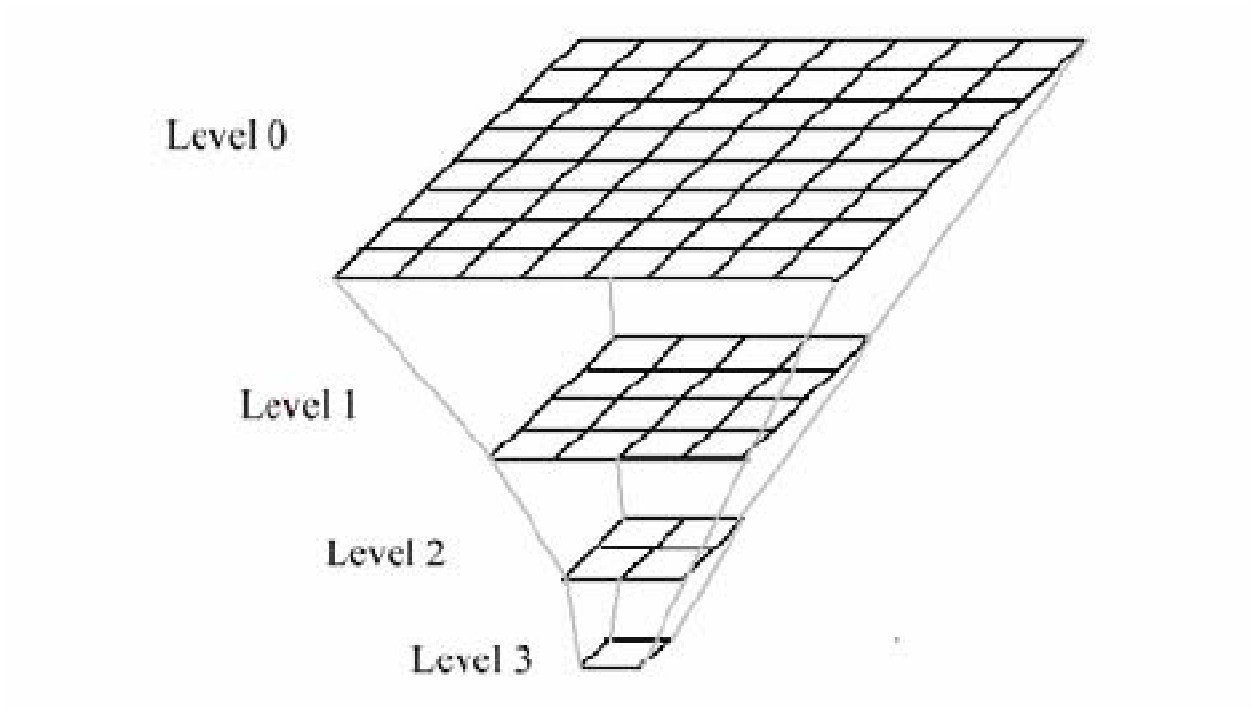

Figure 3.2: Pyramid Down

There are three types of image pyramids.

Gaussian Pyramid. In a Gaussian pyramid, subsequent images are weighted down using a Gaussian average and scaled down. Each pixel containing a local average that corresponds to a pixel neighborhood on a lower level of the pyramid. This technique is used especially in texture synthesis.

Laplacian Pyramid. A Laplacian pyramid is very similar to a Gaussian pyramid but saves the difference image of the blurred versions between each levels. Only the smallest level is not a difference image to enable reconstruction of the high resolution image using the difference images on higher levels. This technique can be used in image compression.

Steerable Pyramid. A steerable pyramid is an implementation of a multi-scale, multiorientation used for applications including image compression, texture synthesis and object recognition. It can be thought of as an orientation selective version of a Laplacian pyramid.

Hence, we are using the HOG descriptor for object classification so we do not use smoothing since smoothing tends to hurt classification performance.

In general, there is a trade-off between performance and the number of layers that we generate. The smaller the scale factor is, the more layers need to create and process - but this also gives an image classifier a better chance at localizing the object we want to detect in the image.

A large scale factor will yield less layers and perhaps might hurt our object classification performance; however we have obtained higher performance gains by processing less layer.

3.3.2 Image Mirroring

A mirror image (in a plane mirror) is a reflected duplication of an object that appears almost identical, but is reversed in the direction perpendicular to the mirror surface.

Since cat faces are generally left-right symmetric we can increase our training dataset by adding mirrored versions of each image in training data. So this step doubles the size of our training dataset. This is obviously optional but is useful in many object detection tasks.

3.4 Sliding Window

In the context of computer vision (and as the name suggests), a sliding window is rectangular region of fixed width and height that "slides" across an image.[?] For each of these windows, we would normally take the window region and apply an image classifier to determine if the window has an object that interests us - in this case, a face. Combined with image pyramids we can create image classifiers that can recognize objects at varying scales and locations in the image. These techniques, while simple, play an absolutely critical role in object detection and image classification.

The approach involves scanning the image with a fixed-sized rectangular window and applying a classifier to the sub-image defined by the window. Successively scaled copies of the image we apply sliding window technique for identifying and localizing objects. For each of these windows, we normally take the window region and apply HOG to determine if the window has an object that interests us. We used 80x80 sliding window for identifying and localizing objects.

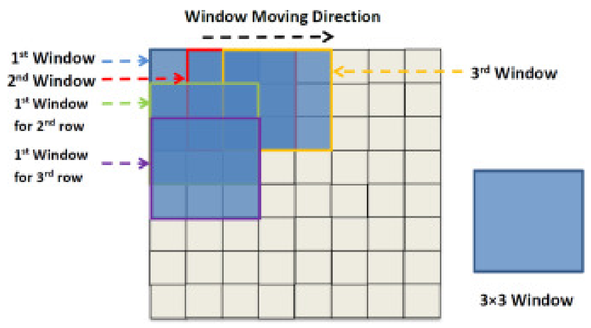

Figure 3.3: An Illustration of Sliding Window Technique

3.5 Feature Extraction

In the HOG feature descriptor, the distribution (histograms) of directions of gradients (oriented gradients) are used as features. Gradients (x and y derivatives) of an image are useful because the magnitude of gradients is large around edges and corners (regions of abrupt intensity changes) and we know that edges and corners pack in a lot more information about object shape than flat regions.

We compute centered horizontal and vertical gradients with respect to x axis and y axis without any smoothing. Then we compute gradient magnitude and gradient angle or orientation. For color image we can take the color channel with highest gradient magnitude for each pixel. Then we quantize the gradient orientation in 9 bins. Then we concatenate the histograms. [1]

3.5.1 Calculate Gradient Orientation and Magnitude

The gradient is a multi-variable generalization of the derivative. While a derivative can be defined on functions of a single variable, for functions of several variables, the gradient takes its place. The gradient is a vector-valued function, as opposed to a derivative, which is scalar-valued.

The first order derivative at any point in an image is computed using the magnitude of the gradient:

$$\nabla f = \begin{bmatrix} G_x \\ \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Where the magnitude is:

$$\nabla f = mag(\nabla f) = \sqrt{{G_x}^2 + {G_y}^2}$$

The direction of the gradient vector is the angle given by $\alpha(x, y)$

$$\alpha(x, y) = \tan^{-1}(\frac{G_y}{G_x})$$

We need to first calculate the horizontal and vertical gradients. This is easily achieved by filtering the image with the following masks.

-1	0	1

-1
0
1

The x-gradient fires on vertical lines and the y-gradient fires on horizontal lines. The magnitude of gradient fires where ever there is a sharp change in intensity. None of them fire when the region is smooth. The gradient image removed a lot of non-essential information (e.g. constant colored background), but highlighted outlines. At every pixel, the gradient has a magnitude and a direction. For color images, the gradients of the three channels are evaluated. The magnitude of gradient at a pixel is the maximum of the magnitude of gradients of the three channels, and the angle is the angle corresponding to the maximum gradient.

3.5.2 9-bin Histogram

The image window is divided into small spatial region called "cells". For each cell, we accumulate a local 1-D histogram forms the basic "orientation histogram" in the cell. This combined cell level 1-d histogram divides the gradient angle range into a fixed number of predetermined bins. The gradient magnitudes of the pixels in the cell are used to vote into the orientation histogram.

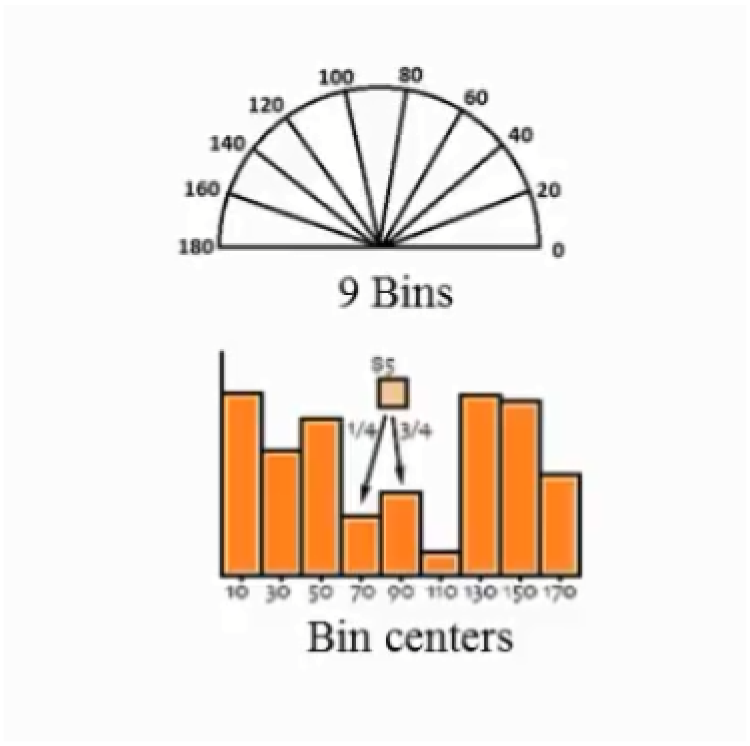

Figure 3.4: 9-bin Histogram

A bin is selected based on the direction, and the vote (the value that goes into the bin) is selected based on the magnitude. Interpolate linearly between neighboring bin centers. An example of interpolation is given below:

Let θ=85 degree, There is no bin for 85 degree. Close neighbor bins are 70 degree and 90 degree. Distance to the closest bins are 15 and 5 respectively. Hence ratios are:

$$\frac{5}{15+5} = \frac{5}{20} = \frac{1}{4} \quad \text{and} \quad \frac{15}{15+5} = \frac{15}{20} = \frac{3}{4}$$

At last we concatenate the histograms.

3.5.3 Normalizing Blocks

Gradients of an image are sensitive to overall lighting. If you make the image darker by dividing all pixel values by 2, the gradient magnitude will change by half, and therefore the histogram values will change by half. Ideally, we want our descriptor to be independent of lighting variations. In other words, we would like to "normalize" the histogram so they are not affected by lighting variations.

Let's say we have an RGB color vector [128, 64, 32]. The length of this vector is

$$\sqrt{128^2 + 64^2 + 32^2} = 146.64$$

This is also called the L2 norm of the vector. Dividing each element of this vector by 146.64 gives us a normalized vector [0.87, 0.43, and 0.22]. Now consider another vector in which the elements are twice the value of the first vector 2 x [128, 64, and 32] = [256, 128, and 64].

Now that we know how to normalize a vector, while calculating HOG we can simply normalize the 91 histogram the same way we normalized the 31 vector above. It is not a bad idea, but a better idea is to normalize over a bigger sized block of 1616. A 1616 block has 4 histograms which can be concatenated to form a 36 x 1 element vector and it can be normalized just the way a 31 vector is normalized. The window is then moved by 8 pixels and a normalized 361 vector is calculated over this window and the process is repeated.

3.5.4 Calculate HOG Feature Vector

Each 1616 block is represented by a 36*1 vector. So when we concatenate them all into one giant vector we obtain a 36*105 = 3780 dimensional vector.

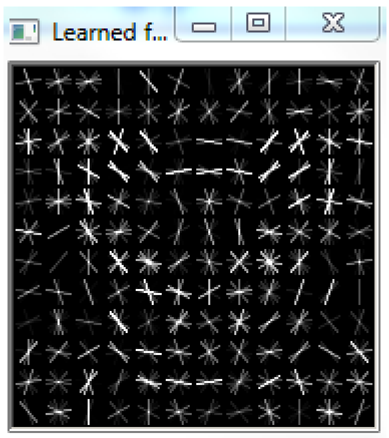

Figure 3.5: Feature Vector Obtained From Training Data (Cat Face)

3.6 Training Object Detector

We used a Support Vector Machine to train our desired detector. We used the extracted HOG features as the input to the SVM. Detailed work with SVMs is not under the scope of our project. However, we will provide a general discussion on SVMs in next few sections.

3.6.1 Support Vector Machine

Support vector machine is a supervised learning model with associated algorithms that analyzes data used for classification. It can be used in training detectors for object detection which is why it holds relevance to us. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm can build a model that assigns new examples to one category or the other. SVMs can efficiently perform both linear and non-linear classification. The original SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. In 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplane.

3.6.1.1 Linear SVM

Here we discuss the working mechanism of a Linear SVM. We are given a training dataset of n points of the form,

$$(x_1, y_1), , (x_n, y_n)$$

Where the yi are either 1 or -1 each indicating the class to which the point xi belongs. Each xi is p-dimensional real vector. We want to find the "maximum-margin hyperplane" that divides the group of points xi for which yi = 1 from the group of points for which yi = -1, which is defined so that the distance between the hyperplane and the nearest point xi from either group is maximized.

Any hyperplane can be written as the set of points x satisfying,

$$w.x - b = 0$$

where w is the (not necessarily normalized) normal vector to the hyperplane. This is much like Hesse normal form, except that w s not necessarily a unit vector. The value of b divided by w determines the offset of the hyperplane from the origin along the normal vector w.

If the training data are linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible. The region bounded by these two hyperplanes is called the "margin", and the maximum-margin hyperplane is the hyperplane that lies halfway between them. These hyperplanes can be described by the equations,

$w.x - b = 1$

and

$w.x - b = -1$

Geometrically, the distance between these two hyperplanes is, $\frac{2}{|w|}$ So to maximize the distance between the planes we want to minimize w, As we also have to prevent data points from falling into the margin, we add the following constraint: for each i either,

$$w.x - b \geq 1 \quad \text{if} \quad y_(i) = 1$$

or,

$$w.x - b \leq -1 \quad \text{if} \quad y_(i) = -1$$

These constraints state that each data point must lie on the correct side of the margin.

This can be rewritten as:

$y_i(w.x - b) \geq 1$ for all $1 \leq i \leq n(1)$

We can put this together to get the optimization problem:

Minimize $|(|w|)|$ subject to $y_i(w.x_i - b) \geq 1$,for $i = 1$ to n.

An easy-to-see but important consequence of this geometric description is that the max-margin hyperplane is completely determined by those xi which lie nearest to it. These xi are called support vectors.

To extend SVM to cases in which the data are not linearly separable, we introduce the hinge loss function

This function is zero if the constraint in (1) is satisfied, in other words, if Xi lies on the correct side of the margin. For data on the wrong side of the margin, the function's value is proportional to the distance from the margin. We then wish to minimize,

$$[\frac{1}{2}\sum_{i=1}^{n} max(0, 1 - y_i(w.x_i - b))] + \lambda||w||^2$$

where the parameter λ determines the tradeoff between increasing the margin-size and ensuring that the Xi lie on the correct side of the margin. Thus, for sufficiently small values of λ, the soft-margin SVM will behave identically to the hard-margin SVM if the input data are linearly classifiable, but will still learn if a classification rule is viable or not.

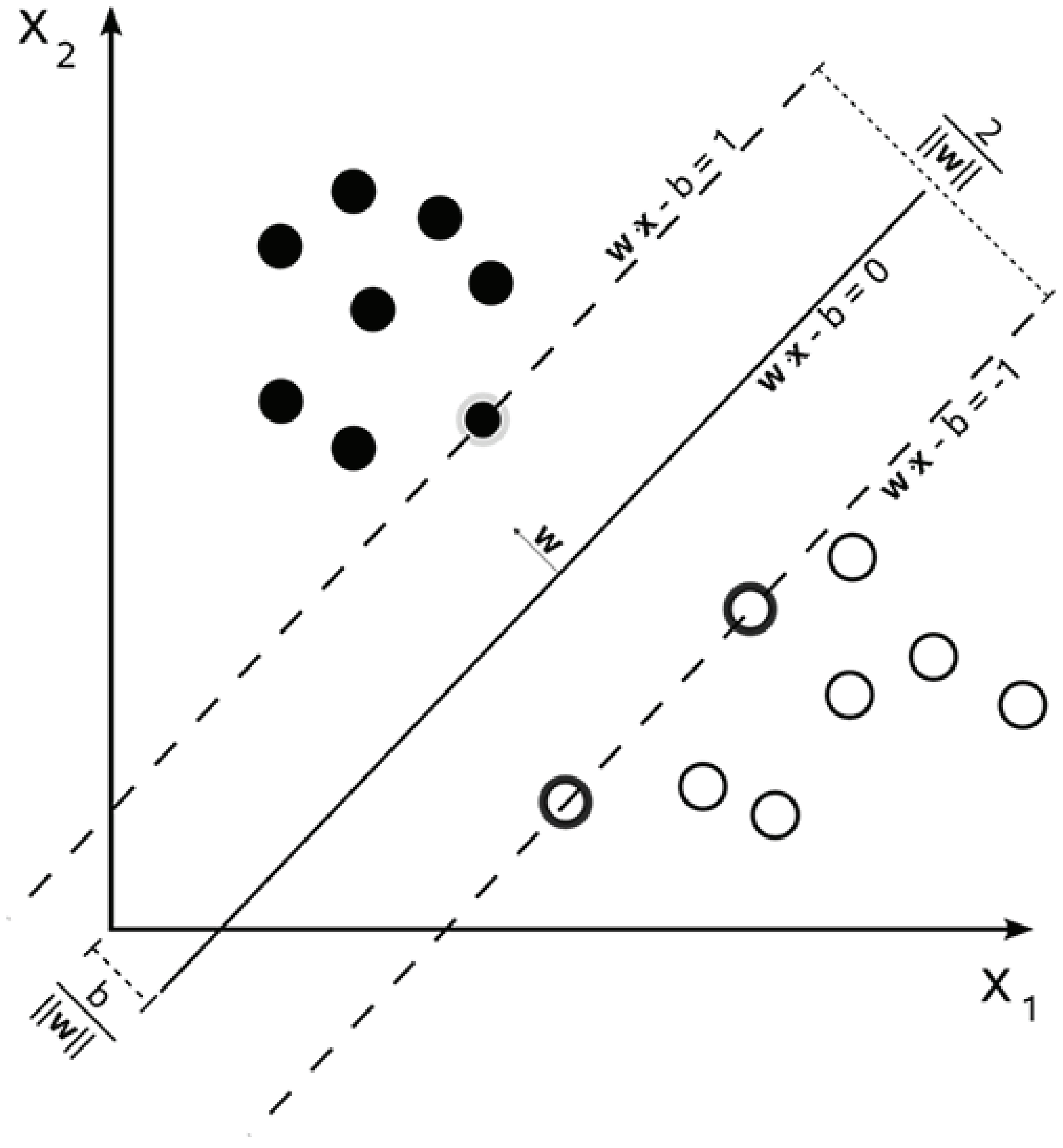

Figure 3.6: Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors. Image Courtesy: Wikipedia

3.6.1.2 Structural SVM

A structural SVM is a supervised machine learning method for learning to predict complex outputs. This is contrasted with a binary classifier which makes only simple yes/no predictions. A structural SVM, on the other hand, can learn to predict complex outputs such as entire parse trees or DNA sequence alignments. Training a classifier consists of showing pairs of correct sample and output label pairs. After training, the structural SVM model allows one to predict for new sample instances the corresponding output label; that is, given a natural language sentence, the classifier can produce the most likely parse tree. To do this, it learns a function F(x,y) which measures how well a particular data sample x matches a label y. When used for prediction, the best label for a new x is given by the y which maximizes F(x,y).

In our work, the detector learns the parameter vector by formulating the problem as a structural SVM problem. The training procedure produces an object detector which can be used to predict the locations of objects in new images.

3.7 Applying Non Maximum Suppression

No matter which HOG + Linear SVM method you choose, you will (with almost 100% certainty) detect multiple bounding boxes surrounding the object in the image.

While each detection may in fact be valid, we certainly don't want our classifier to report to back to us saying that it found six faces when there is clearly only one face. This is common "problem" when utilizing object detection methods.

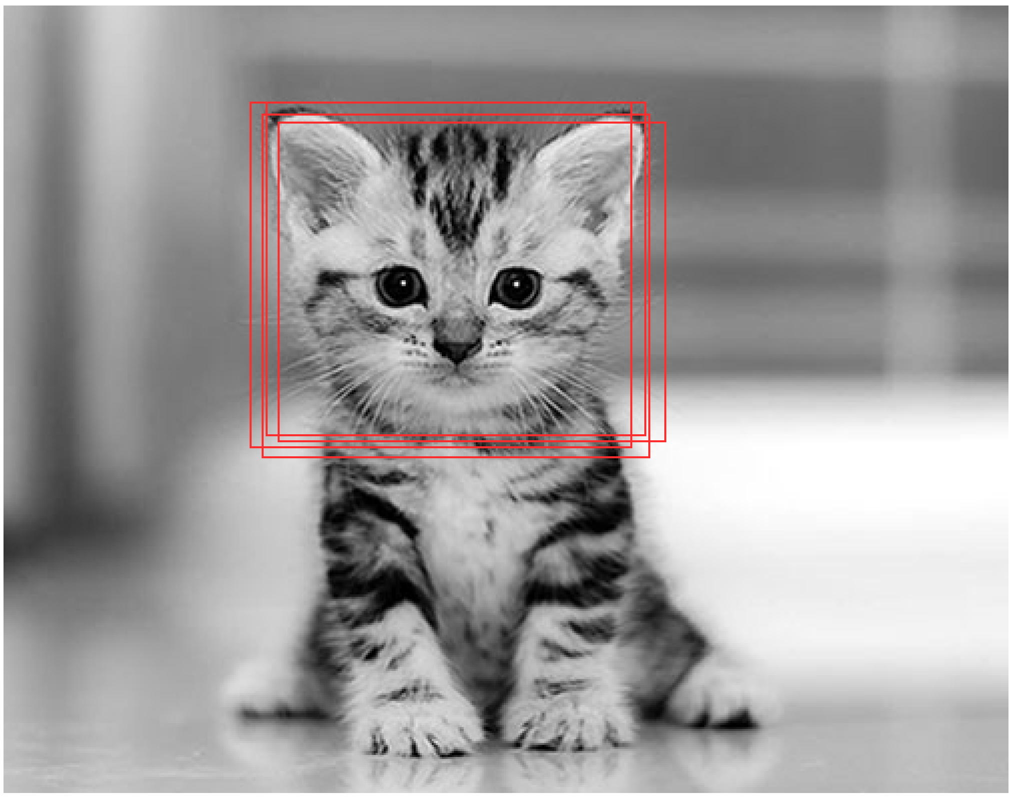

Figure 3.7: Multiple Detection on the same object before applying Non-maximum Suppression.

To fix this situation we'll need to apply Non-Maximum Suppression (NMS), also called Non-Maxima Suppression. [12]

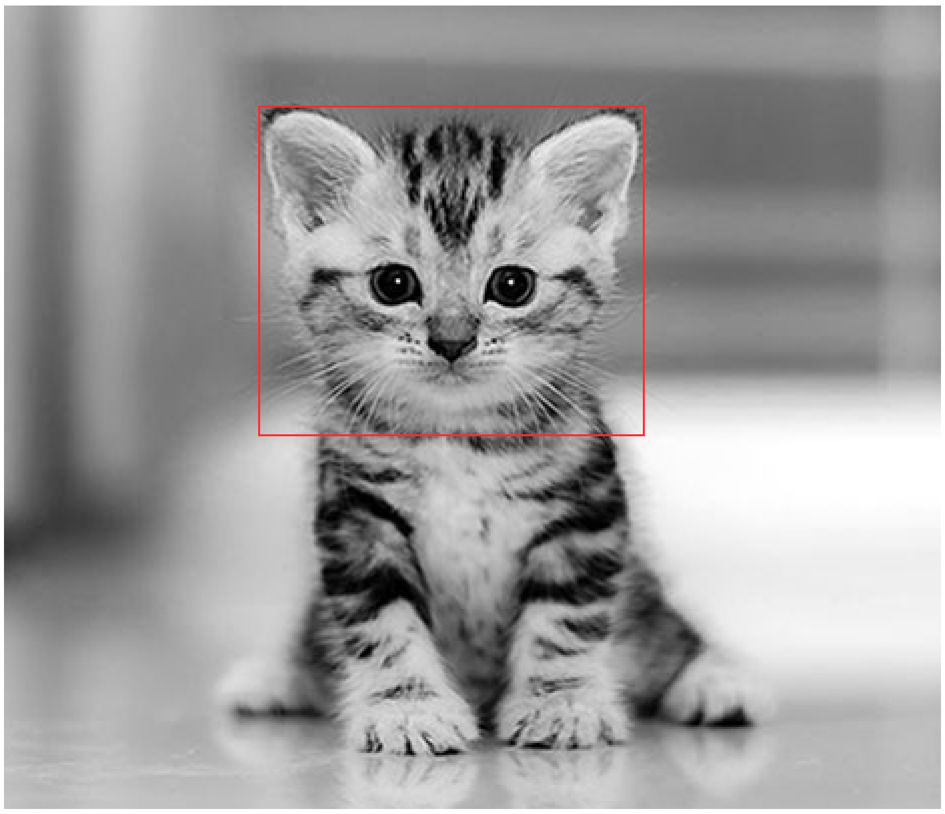

Figure 3.8: After applying Non-maximum Suppression.

Non-maximum suppression is used as an intermediate step in many computer vision algorithms. Non-maximum suppression is often used along with edge detection algorithms. The image is scanned along the image gradient direction, and if pixels are not part of the local maxima they are set to zero. This has the effect of suppressing all image information that is not part of local maxima.

The most common approach for NMS consists of greedy iterative procedure which we refer to as Greedy NMS. The procedure starts by selecting the best scoring window and assuming that it indeed covers an object. Then, the windows that are too close to the selected window are suppressed. Out of the remaining windows, the next top scoring one is selected, and the procedure is repeated until no more windows remain. This procedure involves defining a measure of similarity between windows and setting the threshold for suppression. These definitions vary substantially from one work to another, but typically they are manually designed. Greedy NMS, although relatively fast, has a number of downsides. First, by suppressing everything within the neighborhood with a lower confidence, if two or more objects are closed to each other, all but one of them will be suppressed. Secondly, Greedy NMS always keeps the detection with the highest confidence even though in some cases another detection in the surrounding might provide a better feed for two objects. Third, it returns all the bounding-boxes which are not suppressed, even though many could be ignored due to a relatively low confidence or the fact that they are sparse in a sub-region within the image.

Rather than greedy NMS method, there are multiple ways to remedy this problem. Triggs at al. suggests to use the mean-shift algorithm to detect multiple modes in the bounding box space by utilizing the (x,y) coordinates of the bounding box as well as the logarithm of the current scale of the image. In our work we used Felzenswalb's non-maximum suppression method.

Chapter 4

Experimental Results

After implementing HOG to detect cat's face we got results which will be presented in this chapter. Here we will show the results of testing on random dataset. After that we will also present our findings of testing on a special dataset to find the efficiency of the detector in isolating cat faces in images with varying angles and distance of the object from viewing plane.

4.1 Dataset

We randomly selected cat images and used them to make our training and testing datasets. The cat images are collected from "Kaggle Dog Vs. Cat" dataset. Both training and testing datasets were manually labeled. We labeled the positive samples with tight bounding truth boxes in our training dataset, we also labeled some negative samples by drawing ignore boxes where it was needed. We labeled the positive samples with truth boxes in the testing dataset so that the detector can compare it's detections with the positive samples.

4.2 Results

We used 60% of the images to train our detector and 40% of the images to test it. Our detector showed recall rate of 83% and precision rate of 97%. The final result is reasonably well for a HOG based object detector considering the fact that our testing dataset had cat images of varying poses, angles and distance of the object from viewing plane. We used a separate dataset to observe how well the detector performs and to find it's limitation; this will be discussed more in the next section.

4.3 Testing Images with Varying Angles

We used an additional testing dataset to find out how well our detector performs if the testing image is not a full frontal cat face. To do this we took a few full frontal cat faces and rotated those anti-clockwise from 0 degrees to 35 degrees with a 5 degree increment. We observed that the detector can isolate all the cat faces with 10 degree rotation, it can isolate some cat faces in between 11 to 20 degrees depending on the distance from the viewing plane. If the cat face is closer to the viewing plane then the detector can correctly isolate it even if the rotation is higher than 10 degrees.

4.4 Visualization of Experimental Results

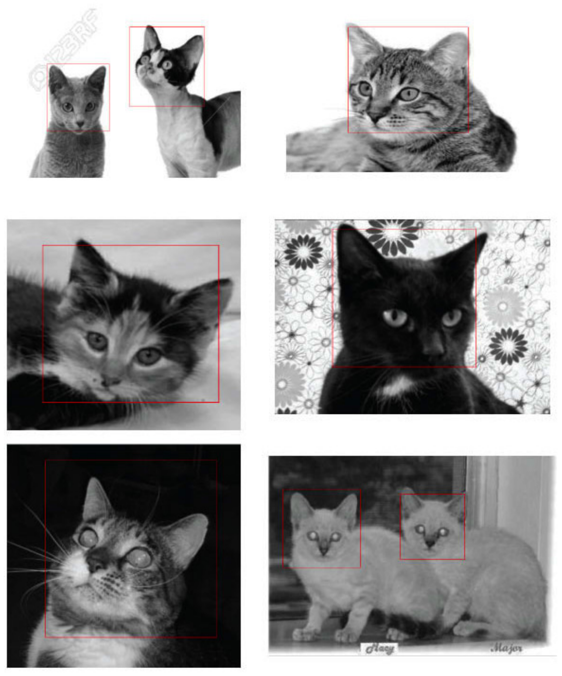

Figure 4.1: Some of the detected cat faces

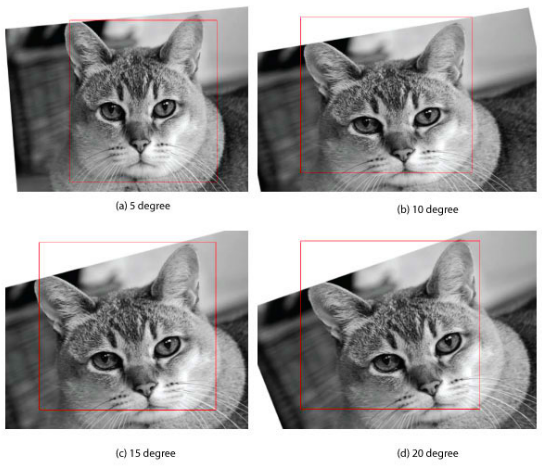

Figure 4.2: Detection on cat face with varying angle. (CCW Rotation)

4.5 Finding Optimal Value of C for SVM

The C parameter tells the SVM optimization how much we want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. The optimal value of C can vary depending on the data. We had to find the optimal value of C for our data, this involved relying on trial and error. We ran the SVM with different values of C and found that in our case the optimal value of C is 39.2. To do it we ran the SVM in a loop and recorded some values of C for which the recall rate of the detector is at least 80%. The table 4.1 shows those values of C.

Values of C	Precision	Recall
30.05	95%	81%
30.21	97%	80%
30.43	96%	80%
30.88	96%	81%
31.20	95%	81%
39.02	97%	82%
39.05	96%	81%
39.08	98%	82%
39.20	98%	83%

Table 4.1: Recall and Precision for Different Values of C

Chapter 5
Conclusion

Many works have been done in object detection in the field of computer vision before. But object detection with varying pose was always a challenging task.

Here we showed that, by extracting feature for object detection using HOG with the help of MMOD and training with SVM gives us a detector which is very orientation robust.

In other methods it would take hours days to train and requires to fiddle with false negative rates and all kinds of spurious parameters. HOG training is considerably simpler. Also in a very short time with a large amount of training data can be used.

Moreover HOG trainer train on every sub-window. So hard negative mining is not needed. It also means that often we do not have to use large training data.

Limitation:

- More than 20 degree rotated object cannot be detected.
- 3% false positive detection.

Future Work:

- Reduction in false detection.
- Improvement of the detection.
- Detection of 3D rotated object.

Bibliography

[1] “Computer vision,” in *ScienceDaily - sciencedaily.com*, 2017.

[2] “An introduction to classification:feature selection,” in *improvedoutcomes.com*, 2017.

[3] C. Cortes and V. Vapnik, “Support vector networks,” in *Machine Learning*, vol. 20, 1995.

[4] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” 2001.

[5] M. Ratasch, S. Romdhani, and T. Vetter, “Efficient face detection by a cascaded support vector machine using haar like features,” in *Pattern Recognition Symposium*, 2004.

[6] F. Pedro, B. Ross, M. David, and R. Deva, “Object detection with discriminatively trained part-based models,” 2009.

[7] N. Dalal and B. Triggs, “Hog for human detection,” 2005.

[8] “Introduction to support vector machines - opencv 2.4.13.3 documentation,” in *docs.opencv.org*, 2017.

[9] D. E. King, “Max-margin object detection,” 2015.

[10] P. Wang and Q. Ji, “Multi-view face detection under complex scene based on combined svms,” in *Proc. of ICPR*, 2004.

[11] "Tensorflow," in *tensorflow.org*, 2017.

[12] "Image pyramids," in *OpenCV 2.4.13.4 Documentation*, 2017.

[13] A. Rosebrock, "Sliding windows for object detection with python and opencv," in *pyimagesearch.com*, 2017.

[14] J. Hosang, R. Beneson, and B. Schiele, "Learning non-maximum suppression," 2017.

Appendix A
List of Notations

σ	This is Sigma
∇	This is Nabla
α	This is Alpha
$\sum$	This is Summation

Appendix B

List of Acronyms

HOG	Histogram of Oriented Gradients
CNN	Convolutional Neural Network
SVM	Support Vector Machine
MMOD	Max Margin Object Detection
1D	One Dimensional