OBJEKT DETECTION USING HISTOGRAM OF GRADIENTS

Salma HAMDI

Supervised by Dr. Naeem Nisar SHEIKH

Spring 2018
OBJECT DETECTION USING HISTOGRAM OF GRADIENTS

Capstone Report

Student Statement:
I have applied ethics to the design process and in the selection of the final proposed design. And I have held the safety of the public to be paramount and has addressed this in the presented design wherever may be applicable.

__________________________
Salma Hamdi

Approved by the Supervisor

__________________________
Dr. Naeem Nisar Sheikh
1. ACKNOWLEDGEMENTS

This capstone project was a great chance for both personal and professional development. It was an opportunity to work on challenging topics and to put under the spotlight my persistence and my hard-working self. However, none of this would have been achieved without the limitless help, advice, and patience of my supervisor Dr. Naeem Nisar Sheikh. Dr. Sheikh helped me in taking the right decisions and provided me with the adequate support and the necessary advice for the completion of my project. So, thanks to his knowledge, his availability, and his openness to discussion, I was able to learn various notions that I believe would help me even after the end of my AUI life.

Moreover, I would also like to take a moment to express my gratitude for whom I have become today thanks to Allah, to my beloved family and parents, to my supporting friends, my devoted professors, as well as any person who has tried to push me to become my better self, who has ever inspired me, and believed in me. Hence, for such trust that many people have put in me, I will make sure to strive for greatness, to fight for a better world, because that’s eventually what we are all meant to be doing.
Table of Contents

LIST OF FIGURES 6
LIST OF TABLES 7
ABSTRACT 8
INTRODUCTION 9
LITERATURE REVIEW 11
OUR METHODOLOGY 13
  Basic Definitions 13
  Dataset 15
  Tools 16
IMPLEMENTATION 16
  Histogram of Oriented Gradients 16
    Feature extraction and analysis 16
      Compute the Gradients 16
      Compute Cell Histogram 18
        Define a cell 18
        Constructing histogram 18
      Block overlapping 19
        Define a block 19
        Normalizing the histograms 19
  Classification using SVMs 20
    Linear Support Vector Machines 20
      Assumptions 20
      Overall idea 20
      SVM mathematical explanation 21
      Convex Optimization Problem 22
      The Margin (M) 23
      Dual Formulation 25
      Lagrange Multiplier 26
      Slater’s Condition and Sion’s Minimax Theorem 27
      The Wolfe dual problem 27
      Karush-Kuhn-Tucker conditions 28
      Stationarity 29
      Complementary slackness 29
      Primal feasibility and Dual feasibility 29
      Quadratic Programming 30
    Soft-Margin Support Vector Machines 30
      Finding the hyperplane for HOGs 32
        A small example 32
TRAINING AND TESTING PHASE

Process
Metrics to Assess Performance
Results

Testing Phase 1
No C (Hard margin):
C=1
C=0.01
C=0.5
C=10

Testing Phase 2
Flipped upside-down and flipped to the side images

LOCAL BINARY PATTERN

TESTING WITH LBP

STEEPLE ANALYSIS

Societal
Ethical
Technical
Environmental
Political
Legal
Economic

PROJECT IMPACT

CONCLUSIONS

FUTURE WORK

REFERENCES
2. LIST OF FIGURES

Figure 5.1. The Dalal & Triggs’ pipeline
Figure 5.2. The pipeline defined in our project
Figure 5.1.1. Gradient in terms of intensity
Figure 5.1.2. Gradient in a curve
Figure 5.1.3. Graphic Histogram
Figure 5.1.4. D.S. Histogram
Figure 5.2.1. Sample of the dataset
Figure 6.1.1.1. Mask for the x-direction and mask for the y-direction
Figure 6.1.1.2. Graphical representation and Implementation form
Figure 6.1.1.3. Block of stride 1
Figure 6.2.1.2. Linear separators in 2D and 3D
Figure 6.2.1.3-1. Different possible hyperplanes: case 1, case 2, case 3
Figure 6.2.1.3-2 Two additional hyperplanes crossing support vectors D and 2.
Figure 6.2.1.3-3. Margin
Figure 6.2.1.5. Margin computation
Figure 6.3.1.1. Linear Separators
Figure 8.1. An example matrix and its corresponding comparison matrix
2. LIST OF TABLES

Table 6.2.1.6. Primal to dual constraint and variable conversion
Table 7.2. Accuracy, precision, recall, specificity, and F-measure formulas.
Table 7.3.1.1-1. No C Confusion matrix
Table 7.3.1.1-2. No C Performance
Table 7.3.1.2-1 C=1 Confusion matrix
Table 7.3.1.2-2 C=1 Performance
Table 7.3.1.3-1 C=0.01 Confusion matrix
Table 7.3.1.3-2 C=0.01 Performance
Table 7.3.1.4-1. C=0.5 Confusion matrix
Table 7.3.1.4-2. C=0.5 Performance
Table 7.3.1.5-1. C=10 Confusion matrix
Table 7.3.1.5-2. C=10 Performance
Table 7.3.2.1-1. Flipped upside-down and to the side Confusion matrix
Table 7.3.2.1-2. Flipped upside-down and to the side Performance
Table 9.1. HOG-LBP Confusion matrix
Table 9.2. HOG-LBP Performance
3. **ABSTRACT**

This report details the work done on the capstone project under the theme of image processing and image recognition and precisely human detection using Histogram of Oriented Gradients. Human detection has already been accomplished and several papers discussing it have been published. In this project, we are performing human detection using two methods: Histogram of Gradients (HOG) only, and Histogram of Gradients along with Local Binary Pattern (HOG-LBP). The two methods use quite similar concepts in terms of computing the variation of each pixel in contrast with its surroundings and its neighbors, and using that to build histograms. The two methods of processing the image output a feature vector of that image. The feature vectors of all images are considered as the new processed data that needs to be classified. For classification, we are using support vector machines (SVM). The type of support vector machines used is linear SVMs. Yet, within the linear type, we are testing for two subtypes: hard-margin linear SVMs and soft-margin linear SVMs.

This project consists of two main steps: training and testing. The training phase consists of using a dataset of pedestrians, analyzing that dataset using HOG-only or HOG-LBP, then, using SVMs to construct a classifier. The second phase consists of testing the classifier on a new dataset. The testing for the HOG-only approach resulted in a 73.5% accuracy while the HOG-LBP approach gave a 100% accuracy, both for the same 200 image training and 200 image testing dataset.
4. **INTRODUCTION**

Cameras have become a must everywhere and to everyone, from personal use to professional and security uses. Hence, millions of scene-captures happen daily. Yet, until about half a century ago, the captured images and videos represented, only, a taken slot and space in memory. Along the previous decade, the computer vision field started to know major changes, advancements, and research investments.

Computer vision is one of the main sub disciplines of computer science, which is concerned with deciphering and interpreting images and videos. Such a task has become mandatory with the technological big bang. Instead of storing data in its raw format, it is interesting to try and label it, give it meaning, or try to extract some patterns from it. For instance, such processing is being done nowadays by several companies such as Google for its Google Photos, Facebook in its identification/tagging suggestions, and many other application that are being released to serve, for example, blind people and help detecting what is surrounding them using their phone’s camera. Furthermore, computer vision is also considered as an important pillar in the field of robotics. By allowing robot to process images, they can become more aware of their surroundings and thus, become more effective in navigating and/or manipulating their environments.

One of the main targets of computer vision is object detection and especially human detection and recognition. Detecting the presence of humans in images or in videos became an important step towards more intelligent machines, and this importance will be further emphasized and explained in the STEEPLE analysis.

Considering its importance, human detection is seen as a crucial technique when it is being adopted for critical uses. With humans being subjects of the technique, it is necessary that the mechanism should be precise enough to reduce the error margins or even eliminate them. That is why the human detection task is more difficult than it seems to be. First, humans can have various postures, so detecting them can be harder than most regular objects. Second, occlusion
issues can impede the process. Third, in order to fully be able to recognize humans in all their states, it is important to have a wide and diversified dataset.

Nonetheless, despite these various stumbling blocks, progress has been recorded in human detection, and various ways are being used in order to perform such task. One of the methods being used, and which we will be using, is histogram of gradients approach along with support vector machine classifier. In the following sections, we will have look at the mechanism behind this method and the mathematical reasoning behind support vector machines. Then, we will be testing the accuracy of our program in identifying humans vs. non-humans pictures. Moreover, we will be trying to use additional techniques such as local binary pattern to further enhance the accuracy of our program.
5. LITERATURE REVIEW

Object recognition and detection has been performed for a while now, and several techniques were used in doing so. The existing object detection techniques vary in terms of principle from template matching, shape based recognition, to color/texture based recognition.[24]

Template matching is a method that is based on having an image as a template and trying to find occurrences of that template in a image. In other words, this technique consists of saving an image as a standard form of an object, and looking for a similar instance in an image. This method is frequently used with text and non-colored images. This particular use is due, first, to the fact that letters and numbers don’t usually exist in various forms, so having a standard version to compare to can be easy. Second, an object can be similar to the reference we have, but the colors could be different, so it is better to have grey-scaled images.[24]

Shape based recognition consists of detecting objects using their shapes. The edges of an object can most of the time allow matching the object. Hence, by analysing the shape of a set of templates, we can find that particular object. This technique can be used for colored images as well as gray-scale images. The color in this case is not considered as a detection criterion. However, for color based recognition, it consists of using shape based recognition, and adding to it the color criterion. In this technique, we start accounting for colors. Colors become additional information that can further help identifying objects having particular color histograms.

Taking into consideration the fact that our project subjects are humans, and that humans can have different skin shades, and can wear different clothes with different textures and colors, so using the template matching and the color based recognition would be ineffective. Therefore, we are left with the shape-based method. Within the shape-based method, a widely used method is Histogram of Oriented Gradients. The Histogram of Oriented Gradients (HOG) technique is used to extract features of objects following a change in the intensity. Hence, following the distribution of those intensity gradients, the edges of objects are
highlighted to allow for features to be extracted and for shapes to be discerned clearer. After getting its shape, the object is then classified using a machine-learning technique, in our case we are using Support Vector Machines.

The first step to better understanding the approach was looking at papers analysing object/human detection using HOG technique. The project has been already tackled, and several papers, discussing it, have been published. Thus, we decided to refer to widely-used papers in the topic of human detection, such as *Histograms of Oriented Gradients for Human Detection* written by Navneet Dalal and Bill Triggs in 2005. The paper describes the method of histograms of gradients and how it can be used in human detection. Moreover, according to Dalal and Triggs, the paper “shows experimentally that grids of Histograms of Oriented Gradient (HOG) descriptors significantly outperform existing feature sets for human detection.”[22] The paper, indeed, shows results of different ways of implementation of human detection and among those results, the HOG method shows to outperform other methods.

For classification, several techniques are available to perform this task such as convolutional neural networks, but the one we will be using in our program is support vector machines. Our choice was based, first on our interest in the concept, and second on our influence by the paper we are referencing.

Support vector machines are one of the widely used supervised machine learning algorithms. They require a strong understanding of the math behind them, in order to be able to implement them. Therefore, to do so, I used the e-book *Support Vector Machines Succinctly* by Alexandre Kowalczyk. This book was helpful in understanding the reasoning behind support vector machines, as well as, it provided the appropriate math needed to understand such complex concepts. The mathematical background required consisted of multivariable and linear algebra. Based on the various notions of vectors, matrices, gradients and so on, the writer clearly explains how optimal hyperplane is found by manipulating several primal to dual conversions along with solving various optimization problems.
6. OUR METHODOLOGY

For our methodology, as mentioned in the literature review, we are using the Dalal and Triggs paper written in 2005 since it is a widely referenced paper in human detection. After analysing the paper, we managed to define the steps which our project and program will take. We concluded that during the implementation, the first main task performed is feature extraction and analysis, followed by classification using a machine-learning technique: Support Vector Machine.

The steps of the implementation are based on the Dalal and Triggs’ paper. Below is the Dalal and Triggs paper’s method pipeline:

![Figure 5.1. The Dalal & Triggs’ pipeline [22]](image)

Our pipeline is as follows:

![Figure 5.2. The pipeline used in our project](image)

However before getting deeper into implementation, let us, in order to build a common ground of understanding, first define the key terms used in our project as well as the dataset and tools used.

6.1. Basic Definitions
The first term we define is **gradient**. It is a “differential operator applied to a multi-dimensional vector-valued function to yield a vector whose components are the partial derivatives of the function with respect to its variables.” [2] In fact, the gradient can be considered as a special kind of derivatives that applies to functions with more than one variable. That being a definition of the gradient, it is used in pointing to the direction of the steepest ascent or descent. As a vector, the gradient can be seen as a magnitude and as a direction. The magnitude is the maximum possible steepness from that point, while the angle points in the direction of the steepest point.

![Figure 5.1.1. Gradient in terms of intensity [3]](image1)

The second term to highlight is **histogram**. Histogram is a statistical method that is used to display the distribution of certain elements over a number of bins, which represent particular values. Bins are “rectangles to show the frequency of data items in successive numerical intervals of equal size.”[4] In our project, the histogram to be used is less of a graphical entity and more of a data structure.

![Figure 5.1.2. Gradient in a curve [21]](image2)
The reason of the use of the gradient method is that gradients mainly highlight the areas in which there is a change (steep ascent/descent). These areas are usually the borders of objects, so the gradients end up being big around those borders, which helps in defining the object. The type of the object depends more on the edges than on color or other weightless details, and this fits with the use of gradients which discards such non-crucial information.[5]

6.2. Dataset

Our dataset is taken from the PETA dataset (containing the MIT, CUHK, and other datasets), along with the INRIA dataset, which both contain various size (mostly 64 x 128 pixel) black and white, and colored images of pedestrians mainly in a vertical/ portrait mode where the person isn’t upside down. The position of the person, on the other hand, isn’t constrained since the images taken are of pedestrians and bystanders that are supposedly non-static/in motion. Thus, “ideally there is no particular bias in their pose.”[7] Nonetheless, a constraint that was put on the dataset is that individuals in the picture have a height greater than 100.

These images come along with labels that define whether it is a person or not. The MIT dataset provides a file containing labels, while the INRIA dataset defines those labels by dividing the images into two folders: Positive (pos) and Negative (neg).
6.3. **Tools**

In order to implement this project, we are using GNU Octave as a substitute for Matlab, knowing that it doesn’t require a license and is available for free. Octave is a software used for performing complex computations and is known for providing solutions for (non-)linear problems. “Octave helps in solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with Matlab.” [6] Furthermore, it is a very convenient tool for this work because of its facility in working with matrices and vectors, which are at the heart of digital picture representation.

7. **IMPLEMENTATION**

7.1. **Histogram of Oriented Gradients**

7.1.1. **Feature extraction and analysis**

The first step consists of dividing each image into blocks (grid). Then, for each detector window, we will be building gradient vectors using the properties of each pixel i.e. color and intensity.

7.1.1.1. **Compute the Gradients**

In this step, for a 2D surface, the gradient is derived by computing the derivative in terms of x and the derivative in terms of y. As much as this may sound complex, for an image, computing the gradient consists, mainly, of measuring, for each pixel, the variation of its surrounding pixels. “When taking the derivative of an image, you’re actually taking what’s called a discrete derivative, and it’s more of an approximation of the derivative.”[8] This variation is computed by applying a correlation mask in the x direction and another mask in the y direction. Mask is a term used in correlation filtering.

Filtering is the process of applying modifications to an image. In order to apply those changes, a filter/mask/kernel is used on each pixel. A mask, also called kernel and filter is simply a matrix that contains the weights by which each pixel is affected by its surrounding pixels.
Hence, the mask is used to re-compute the value of each pixel in a picture. The new value is the sum of the adjacent pixels with the mask weights taken into consideration. This type of filtering is the correlation filtering. [11]

![Mask for the x-direction and mask for the y-direction](image)

Knowing that the images used can be colored, for this particular case, the derivatives in x and y directions are computed for the three channels for each pixel. Hence, for a 64 x 128 pixel image, we get three 64 by 128 matrices. Each matrix describes the set of gradients of a color channel (RGB). To get the overall gradients, we average the values of the channel-specific gradients (RGB).

For instance, once the gradient is derived, the magnitude of the resulting vector is used to get some details about the sharpness of the borders. The angle is \( \theta = \arctan \left( \frac{g_y}{g_x} \right) \), and the magnitude of the gradient is \( g = \sqrt{g_x^2 + g_y^2} \).

The angle/ direction of the gradient will normally vary between 0 and 360°, which is called a “signed gradient”. Nonetheless, according to Dalal and Triggs’ paper, for better performance of the program, the gradients are better used as unsigned. The human detection is mainly based on edge detection, which doesn’t require knowing the sign of the gradient since colors aren’t much important. Thus, it is not important to know whether it is an ascent or a descent much more than it is to know that it is a steep slope in which intensity changes so as to spot an edge. In other words, the gradients should be converted from a 0-360° base to a 0-180°. The solution to such conversion is to add \( \pi \) to the thetas that are negative, since theta in our program is retrieved in radians (for all \( \theta < 0 \), \( \theta = \theta + \pi \); \( \theta \text{ in rad} \)).
7.1.1.2. **Compute Cell Histogram**

**Define a cell**

After retrieving the gradients belonging to all pixels, we subdivide the image into cells of 8x8 pixels. Then, for each cell we construct the cell-histogram.

**Constructing histogram**

As previously defined, our histogram will be a data structure containing nine (9) cells. Each cell corresponds to a bin. The range of the histogram goes from 0 to 180° (\(\pi\)). This is because, as mentioned before, the paper we used as reference compares the performance of signed gradients (0-360°) and unsigned gradients (0-180°) and asserts that the unsigned gradients perform better.

The range of each individual bin is \(\frac{180°}{9} = 20°\). The reason of having nine bins is again based on the paper used, in which the performance of the 9 bin histogram is better than less or more bins. Less would lead to omitting some important angles and more would lead to overfitting and having too much information.

Hence, in order to define the contribution of each angle to the bins, we decided to use the linear interpolation with respect to the center. First, we identify the two bins to which the angle is most likely to contribute. Then, the contribution is computed using linear interpolation with respect to the center of those bins. This creates more consistency and more precision than choosing only one bin.

Therefore, eventually, a similar histogram is constructed, figure.6.5 is the graphical representation, and figure.6.6 is the implementation form.
Define a block
In this step, a block is defined as a set of 2x2 cells (2 upper and 2 lower adjacent cells), so it has in total 8x8x4 pixels. A block is used to define the cells whose histograms should be normalized together. The point behind doing so is to have more consistency in terms of change of intensity and some other factors. Hence, normalizing adjacent cells would help maintaining harmony.

Normalizing the histograms
The block swipes over the image with a stride of 1, both in the x-direction and y-direction, each at a time. Therefore, we end up in total with 7x15 blocks.

Since each block has 4 cells, it has 4 histograms, and in total 4x9 bins. The 36 available bins are considered all as one vector, which is normalized using an L2-norm.

L2-norm means

$$||v|| = \sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}$$

and

$$u = v/||v||.$$
Once the block normalization happens, the histograms of all blocks (7x15x4) are gathered into one big HOG, which has the size of 7x15x4x9 = 3780 bins.

7.2. Classification using SVMs

7.2.1. Linear Support Vector Machines

7.2.1.1. Assumptions
It is important to keep the following assumptions in mind while going through the sections below.
First, assuming and considering we already have a dataset, each element in the dataset is labeled as either -1 or 1. In other words, an element is either part of the class 1 or part of the class -1, which is same as either part of class 1 or not part of it. This is only valid within our project since the data we are using does not require a multiclass classification.

7.2.1.2. Overall idea
As a basic idea of support vector machines, they are algorithms that are used to classify data into classes. The classification happens by considering each element in the data as a single point, with the information/values of the point considered as its coordinates. Hence, in a dataset where the elements are scattered around, by simply knowing their locations, the SVM algorithm allows to find the optimal hyperplane that separates our data points such that each of the sides belongs to a class. Finding such hyperplane can be done by assuming that the dataset is mostly linearly separable. A hyperplane is a binary classifier, which is a (n-1)-D plane that is defined in \( R^n \). For instance, in a 2D space, a hyperplane would be simply a line, and in a 3D space, a hyperplane would be a 2D plane.
We feed the dataset that we have, along with the labels of each data point, as input to the support vector machine, then, we get as outcome a slope (w) and a y-intercept/bias (b). These two components allow us to define the hyperplane that we can use for new data to classify it.

### 7.2.1.3. SVM mathematical explanation

Following what was previously mentioned, a hyperplane can be defined by $w \cdot x + b$, with $w$ and $x$ as vectors. Let’s suppose a hyperplane is defined as $w \cdot x + b = 0$. For an element to be in class 1, $w \cdot x + b \geq 0$, and for it to be in class -1, $w \cdot x + b \leq 0$.

However, the equations above do not insure that when trying new data the error margin will be small.
All the above hyperplanes are solutions to the classification of the available data. However, if new data is added to be classified, there are chances it will be misclassified especially if we have case 2 or case 3.

Thus, one hyperplane is not enough, that is why we will be using two more support hyperplanes (H1: $w \cdot x + b = 1$ and H2: $w \cdot x + b = -1$) in addition to the main one (H0: $w \cdot x + b = 0$).

The purpose of adding these two hyperplanes is to maximize the distance between the first element from class 1 and the first element from class -1, and by first I mean the closest to the main hyperplane. The closest point to the main hyperplane is called a support vector, and we will be seeing more about it later on.

On the other hand, for the distance we are trying to maximize, it is called the margin. Margin is often called “no man's land”[12] since it does not include any element. Optimizing the margin gives us more precision and less error, when classifying new elements.

**Figure 6.2.1.3-2 Two additional hyperplanes crossing support vectors D and 2.**

**Figure 6.2.1.3-3. Margin**

### 7.2.1.4. Convex Optimization Problem

Convex optimization is a kind of optimization that concerns convex functions.
As a simple definition for 2 dimensional functions, a function is labeled convex if, taking any two random points from the graph and drawing line that goes through them, the graph of the function is either below the line or they both overlap. That is why, when performing a minimization for convex functions, it is easier to solve the optimization problem since the global minimum of the function would be the local minimum.

However, to define convex for all multi-dimensional functions, we can start first by defining convex regions, “a convex region is a region where, for every pair of points within the region, every point on the straight line segment that joins the pair of points is also within the region.”[25] In other words, for a region R, if from any point a to any point b, the link between the two sides still exists within the region, that region is called convex. That is why in order to call a graph convex, it needs to limit a convex region from the bottom.

After defining convex functions and convex regions, the convex optimization problem looks like:

\[
\min f(x); \quad \text{such that } g_i(x) \leq 0, \quad i = 1, ..., n
\]

With the condition that \( f \), and all \( g_i \) are convex functions. For the constraints, we can also have equality constraints or other inequality constraints.

Back to our initial problem, knowing that linear functions are convex since the segment between each two points of our line would overlap the line itself. We can say that having linear constraints in our problem makes the region, defined by those constraints, convex. Therefore, this makes the function, under which those constraints are applied, convex as well. So, we will be applying convex optimization on our problem and taking it into consideration during the entire explanation.

7.2.1.5. The Margin (M)

First, taking \( H_1 \), we know that \( w \) is normal to it, so the unit vector \( u \) is \( u = w / \| w \| \). Let \( v \) be a vector that has the magnitude to the margin M, so \( v = M \cdot u \).
Then, let us consider an element \( p \) on \( H_1 \) and an element \( q \) on \( H_2 \), such that \( q \) is the perpendicular projection of \( p \) on \( H_2 \). Starting from the origin, \( p \) (bold means a vector) and \( q \) are defined as \( p = q + v \).

Since we have \( p \) belongs to \( H_1 \), \( w. p + b = 1 \)

\[
\Rightarrow w. (q + v) + b = 1 \\
\Rightarrow w. q + w. M. u + b = 1 \\
\Rightarrow w. q + w. M. (w \parallel w) + b = 1 \\
\Rightarrow w. q + M. (w^2 \parallel w) + b = 1 \\
\Rightarrow w. w = \| w \|^2 \\
\Rightarrow w. q + M. (\| w \|^2 \parallel w) + b = 1 \\
\Rightarrow w. q + M. \parallel w + b = 1 \\
\Rightarrow M. \parallel w = 1 - (w. q + b)
\]

Since we have \( q \) belongs to \( H_2 \), \( w. q + b = -1 \), so,

\[
M. \| w \| = 1 - (-1) = 2 \\
\Rightarrow M = 2/\| w \|
\]

Hence, in order to optimize and maximize the margin \( M \), we have to minimize \( \| w \| \).

Thus, the maximization problem became a minimization one.

In addition to the optimization, new constraints are applied. Knowing that the margin is defined between \( H_1 \) and \( H_2 \), since we know that no element can be within it, all elements would be either above \( H_1 \): \( w. x + b = 1 \) or below \( H_2 \): \( w. x + b = -1 \). In other words, for an element \( x_1 \) that belongs to class 1,

\[
w. x_1 + b \geq 1; \ y_1 = y(x_1) = 1,
\]

And for an element \( x_2 \) that belongs to class -1,

\[
w. x_2 + b \leq -1; \ y_2 = y(x_2) = -1.
\]

Both constraints can be combined into one main constraint:

\[
y_i (w. x_i + b) \geq 1; \ i = 1, ..., n
\]

To minimize \( \| w \| \), we convert it to a quadratic problem. The conversion is mainly about minimizing \( \| w \|^2/2 \), instead of \( \| w \| \), such that the constraint is \( y_i (w. x_i + b) \geq 1 \);
In other words, $y_i (w \cdot x_i + b) - 1 \geq 0 \quad i = 1, \ldots, n$. The reason we are using $\| w \|^{2/2}$ instead of $\| w \|$ is because it is easier to solve this way, as it became a mixture of convex optimization problem and quadratic optimization problem, which is called in the book *Support Vector Machines Succinctly* as “convex quadratic optimization problem.” [15]

### 7.2.1.6. Dual Formulation

The goal right now is

$$\minimize \| w \|^2 / 2,$$

such that $y_i (w \cdot x_i + b) - 1 \geq 0, \quad i = 1, \ldots, n$

This formulation is called “primal formulation.” [13] In order to solve it in an easier way, we have to reformulate it into what is called a “dual formulation” [13]

Duality in optimization is mainly done when looking for a minimum of a function $f$. Hence, when applying the duality, instead of looking for that minimum, we begin looking for the max of the dual version of $f$.

**Table 6.2.1.6. Primal to dual constraint and variable conversion** [28]

<table>
<thead>
<tr>
<th>Primal</th>
<th>Dual</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$th constraint $\leq$</td>
<td>$i$th variable $\geq 0$</td>
</tr>
<tr>
<td>$i$th constraint $\geq$</td>
<td>$i$th variable $\leq 0$</td>
</tr>
<tr>
<td>$i$th constraint $=$</td>
<td>$i$th variable unrestricted</td>
</tr>
<tr>
<td>$i$th variable $\geq 0$</td>
<td>$i$th constraint $\geq$</td>
</tr>
<tr>
<td>$i$th variable $\leq 0$</td>
<td>$i$th constraint $\leq$</td>
</tr>
<tr>
<td>$i$th variable unrestricted</td>
<td>$i$th constraint $=$</td>
</tr>
</tbody>
</table>

However, the maximum found will not be necessarily the specific minimum we are looking for. [26] This is called duality gap. Duality gap is either zero, when the infimum of the primal problem is equal to the supremum of the dual problem or greater than zero when the
supremum is less than the infimum. Duality gap allows strong and weak duality to exist within this scope. Strong duality is when the duality gap is non-existent, and weak duality is when there is a duality gap. [27]

After understanding the duality, the purpose of it is that, sometimes, it can be an easier way to solve the initial primal problem, since the constraints of the primal problem become variables of the dual problem. This is beneficial since usually the constraints are less than the variables. [31]

Hence, we will eventually state the problem as follows

\[ \max(\lambda) \quad K(\lambda) = \sum \lambda_i - 1/2 (\sum \sum \lambda_i \lambda_j y_i y_j x_i x_j) \]

Such that \( \lambda_i \geq 0 \) and \( \sum \lambda_i y_i = 0; \quad i = 1, \ldots, n \).

In the sections below, this formula will be derived step by step. In order to derive it, we will have to introduce various concepts including the Lagrange multipliers.

### 7.2.1.7. Lagrange Multiplier

Lagrange multiplier is used for optimization of functions that are under some constraints which are usually equality constraints. [14]

Hence, mathematically speaking, Lagrange states that \( \nabla f(x) = \lambda \nabla g(x) \), such that \( \lambda \) is the Lagrange multiplier. So, in order to minimize \( f(x) \) while caring for \( g(x) \), we find solution to the equation \( \nabla f(x) - \lambda \nabla g(x) = 0 \), which can also be written as

\[ \nabla K(x, \lambda) = \nabla f(x) - \lambda \nabla g(x) = 0, \]

Such that \( K(x, \lambda) = f(x) - \lambda g(x) \).

Applying the Lagrange method to our problem, \( f(x) \) would be \( f(w) = \| w \|_2^2 \), and \( g(x) \) would be \( g(w, b) = \sum g_i(w, b) \), such that \( g_i(w, b) = y_i (w_i x_i + b) - 1; \quad i = 1, \ldots, n \).

Thus, \( K(w, b, \lambda) = f(w) - \sum \lambda_i g_i(w, b) \);

Each constraint \( g_i \) has a particular \( \lambda_i \)

\[ K(w, b, \lambda) = \| w \|_2^2 - \sum \lambda_i [y_i (w_i x_i + b) - 1] \]

Therefore, to minimize \( \| w \|_2^2 \), we are ought to solve \( K(w, b, \lambda) = 0 \).

Nonetheless, considering the complexity of solving such a problem, it could be solved if we have only few elements. [15]
That is why we will have to perform another duality conversion. The primal problem is minimizing \( w \) and \( b \) while maximizing \( \lambda \):

\[
\min(w, b) \quad \max(\lambda) \quad K(w, b, \lambda); \text{ Such that } \lambda_i \geq 0; \ i = 1, \ldots, n
\]

We are using two optimizations at once because we want the maximum \( \lambda \) that also satisfies the minimum \( w \) and \( b \). Hence, having minimum and maximum optimizations simultaneously will allow us to make sure the fit result is found.

### 7.2.1.8. Slater’s Condition and Sion’s Minimax Theorem

Slater’s condition suggests that having a particular solution for the optimization problem, which satisfies the conditions and exists within the range of the optimized function, means an existing strong duality. [16][17]

After defining the Slater’s Condition, and seeing how we have strong duality, we will be referring to Sion’s minimax Theorem.

Sion’s minimax theorem is based on John Von Neumann’s minimax theorem and on strong duality. The Sion’s minimax theorem states that when we have strong duality,

\[
\min_c \max_b \ f = \max_b \min_c \ f . [29]
\]

For instance, our problem will be stated differently.

Instead of \( \min(w, b) \) \( \max(\lambda) \quad K(w, b, \lambda); \text{ such that } \lambda_i \geq 0; \ i = 1, \ldots, n \), we swap the min and max and get \( \max(\lambda) \ \min(w, b) \quad K(w, b, \lambda); \text{ such that } \lambda_i \geq 0; \ i = 1, \ldots, n \).

Aside from Slater’s condition switch, another primal to dual conversion should be done. The new dual problem, we will be dealing with, will be

\[
\max(\lambda) \quad K(\lambda) = \sum \lambda_i \ - \ 1/2 \ (\sum \sum \lambda_i \ y_i \ y_j \ x_i \ x_j) ;
\]

Such that \( \lambda_i \geq 0 \) and \( \sum \lambda_i \ y_i = 0; \ i = 1, \ldots, n \).

This is called the Wolfe dual problem, which will be further explained below.

### 7.2.1.9. The Wolfe dual problem

The Wolfe dual problem tackles the optimization problem by taking it to another level and using differentiation to solve it.
We already saw that:
\[ K(w, b, \lambda) = \left\| w \right\|^2 / 2 - \sum \lambda_i [y_i (w \cdot x_j + b) - 1] = 1/2 w^T w - \sum \lambda_i [y_i (w \cdot x_j + b) - 1]. \]

In order to perform the optimization, we need to derive \( K(w, b, \lambda) \). Since \( K \) is in terms of \( w, b, \) and \( \lambda \), we need to derive it in terms of the variables \( w \) and \( b \) (partial derivatives and gradients).

\[
dK/dw = w - \sum \lambda_i y_i x_i \\
dK/db = -\sum \lambda_i y_i \\

\]

We solve for \( dK/dw = 0 \) and \( dK/db = 0 \).

\[
dK/dw = 0 \Rightarrow w - \sum \lambda_i y_i x_i = 0 \Rightarrow w = \sum \lambda_i y_i x_i \\
dK/db = 0 \Rightarrow -\sum \lambda_i y_i = 0 \Rightarrow \sum \lambda_i y_i = 0 \\

\]

From the previous results, we try replacing \( w \) and the term \( \sum \lambda_i y_i \) with the results found:

\[
\[ K(w, b, \lambda) = 1/2 w^T w - \sum \lambda_i [y_i (w \cdot x_j + b) - 1] \]
\[
= 1/2 (\sum \lambda_i y_i x_i) (\sum \lambda_j y_j x_j) - \sum \lambda_i [(\sum \lambda_j y_j x_j) x_i + b] - \sum \lambda_i \\
= 1/2 (\sum \lambda_i \lambda_j y_i y_j) - \sum \lambda_i (\sum \lambda_j y_j x_j) x_i - \sum \lambda_i y_i b + \sum \lambda_i \\
= 1/2 (\sum \lambda_i \lambda_j y_i y_j) - \sum \lambda_i \lambda_j y_j x_j x_i - \sum \lambda_i y_i + \sum \lambda_i \\
= -1/2 (\sum \lambda_i \lambda_j y_i y_j) x_i x_j + \sum \lambda_i \\
= \sum \lambda_i - 1/2 (\sum \lambda_i \lambda_j y_i y_j x_i x_j) \\
\]

So, since we no longer have \( w \) and \( b \), instead of writing \( K(w, b, \lambda) \), we write

\[ K(\lambda) = \sum \lambda_i - \frac{1}{2} (\sum \lambda_i \lambda_j y_i y_j x_i x_j). \]

However, the Lagrangian method we have been using so far only works for equality constraints, but our constraints are, in fact, inequality constraints \( (\lambda_i \geq 0; \ i = 1, ..., n) \). The question is how to still use this method but with inequality constraints. To answer this question, I would like to introduce the Karush-Kuhn-Tucker conditions (KKT).[15]

### 7.2.1.10. Karush-Kuhn-Tucker conditions

When facing inequality constraints, new conditions must be satisfied. Those new conditions are known as the KKT conditions.

Without going into very complex details, the main KKT conditions are:
Stationarity

\[dK/dw = w - \sum \lambda_i y_i x_i = 0\]
\[dK/db = \sum \lambda_i y_i = 0\]

These two conditions were already stated before. They mean that a particular point is stationary in terms of the fluctuation of K. “When there is no constraint, the stationarity condition is just the point where the gradient of the objective function is zero. When we have constraints, we use the gradient of the Lagrangian.” [15]

Complementary slackness

\[\lambda_i [y_i (w \cdot x_i + b) - 1] = 0\]

This means either \(\lambda_i = 0\) or \([y_i (w \cdot x_i + b) - 1] = 0\Rightarrow y_i (w \cdot x_i + b) = 1\). [18]

In other words, if \([y_i (w \cdot x_i + b) - 1] > 0\), then \(\lambda_i = 0\), else if \([y_i (w \cdot x_i + b) - 1] = 0\), then \(\lambda_i \geq 0\). In the last case when \(\lambda_i > 0\), the constraints that we have are defined by the support vectors that lay on \(H_1\) and \(H_2\).

Support Vectors

Given a particular hyperplane of separation, we look at the two points (one from each class) closest to the hyperplane, in a sense we are seeking a hyperplane that maximizes the distance with respect to its own closest point. [19]

Primal feasibility and Dual feasibility

Primal feasibility (\([y_i (w \cdot x_i + b) - 1] \geq 0\)) and dual feasibility (\(\lambda_i \geq 0\)) just restate the constraints that the primal problem and the dual problem have, respectively, already set.

Now that we made sure our problem can be solved using inequality constraints, we first need to solve for the Lagrangian multipliers \(\lambda\), and then try to find \(w\) and \(b\) to be able to get our hyperplane. To be able to solve the problem, we use quadratic programming.

7.2.1.11. Quadratic Programming

Quadratic programming is one of the solutions to optimization, specifically optimization that involves multivariable quadratic functions, under linear conditions. [32]
The quadratic programming problem can be stated as follows: [33]

\[
\text{Minimize} \quad \frac{1}{2} x^T Q x - c^T x \\
\text{Such that} \quad -\lambda \leq 0 \text{ and } \lambda . \ Y = 0 ;
\]

Since our problem can be solved by solving the equation

\[ K(\lambda) = \sum_{i} \lambda_i - \frac{1}{2} \left( \sum_{i,j} \lambda_i y_i x_j \right) \]

we will convert it into a minimization problem to solve it using quadratic programming.

So, instead of maximizing \( \lambda \), we will minimize it such that:

\[
\text{Min } \lambda \quad K(\lambda) = \frac{1}{2} \left( \sum_{i,j} \lambda_i y_i x_j \right) - \sum_{i} \lambda_i ; \\
\text{Such that} \quad -\lambda \leq 0 \text{ and } \sum_{i} \lambda_i y_i = 0 ; \quad i = 1, ..., n .
\]

Then, let \( \lambda \) be \( (\lambda_1, \lambda_2, ..., \lambda_n)^T \) (T means transpose), and let \( Y \) be \( (y_1, y_2, ..., y_m)^T \) and \( X \) be \( (x_1, x_2, ..., x_n)^T \).

This way we can write our new quadratic programming version of the problem:

\[
\text{Min } \lambda \quad \frac{1}{2} \sum_{i,j} \lambda_i y_i x_j - \sum_{i} \lambda_i ; \\
\text{Such that} \quad -\lambda \leq 0 \text{ and } \sum_{i} \lambda_i y_i = 0 ; \quad i = 1, ..., n .
\]

\[ \Rightarrow \text{Min } \lambda \quad \frac{1}{2} \lambda^T (Y Y^T (X \ast X^T)) \lambda - \lambda \\
\text{Such that} \quad -\lambda \leq 0 \text{ and } \lambda . \ Y = 0 ;
\]

\[ H=(Y Y^T X X^T) ; x= \lambda ; f=-1; A=-1; b=0; Aeq=Y; beq=0 [30] \]

The ‘.*’ operator is used for an element by element operation.

### 7.2.2. Soft-Margin Support Vector Machines

The Dalal and Triggs paper, we are referring to, uses soft-margin linear support vector machines. Soft-margin SVMs assume that the data is not perfectly linearly separable, so there are some data points that are separate from their group and cluster. After such assumption, soft-margin SVMs allow a small margin of misclassification, while still penalizing such misclassification.

To do so, soft-margin SVMs introduce the slack variables. This slack variable \( (\zeta: \text{zeta}) \) allows relaxing our previous constraints. So, instead of trying to get 100 percent accuracy, we allow
for some errors to happen, that is why the previous problem we had is called **hard-margin** problem while this one is called **soft-margin**. Hence our constraints become the following:

\[ y_i (w \cdot x_i + b) \geq 1 - \zeta_i \quad ; \quad i = 1, ..., n \quad [15] \]

However making our constraints this way infers that the bigger the slack variable, the more likely for the constraint to be valid. Nonetheless, it does not necessarily mean we are having the right classification.[15] For instance, to avoid such a situation, we add all the slack variables to the optimized function:

\[
\text{minimize} \quad \| w \|_2^2 + C \sum \zeta_i, \\
\text{such that} \quad y_i (w \cdot x_i + b) \geq 1 - \zeta_i \quad \text{and} \quad \zeta_i \geq 0; \quad i = 1, ..., n
\]

We added a \( C \sum \zeta_i \) to make sure that, by tuning the C value, we are controlling how loose or tight we want our margins to be. Hence, we are adding a cost to pay for overlooking some data points. This will give a solution to the optimization that will try to balance between the margin constraint and the noise constraint. [15]

After performing similar steps of duality and optimization the problem, we end up with:

\[
\max(\lambda) \quad K(\lambda) = \sum \lambda_i - 1/2 (\sum \lambda_i \lambda_j y_i y_j x_i \cdot x_j); \\
\text{Such that} \quad 0 \leq \lambda_i \leq C \quad \text{and} \quad \sum \lambda_i y_i = 0; \quad i = 1, ..., n \quad [15][34]
\]

The problem, to solve above, is the same problem we were solving for hard-margin SVM, the only difference is in the constraints. \( \lambda_i \) became limited, not only by 0, but also by C. C is called a “regularization term” since it helps in tweaking the Lagrange multipliers’ range.[20]

Moreover, we do not have a fit all C term for all problems. The C constant depends on the set of data we have and can be found with a trial and error way as a brute force method or with “grid search along with cross validation”[15], which are not within the scope of our project, since the Dalal and Triggs paper clearly states the use of a C=0.01 value.

### 7.3. Finding the hyperplane for HOGs

After understanding how support vector machines work and the mathematics behind them, the next step is to use them to classify our dataset into humans and non-humans classes.
Before performing such step, we first applied our understanding on a small example of blacks and whites. The example is defined in a 2D space to better help in visualizing the hyperplane, which will be simply a line.

### 7.3.1. A small example

In this example the elements taken were $W1(1,3)$ $W2(1,1)$ $B1(3,3)$ $B2(3,1)$. The figure below represents the cases in which we have a possible linear separator with the last case being not possible, and in this case Kernel SVMs (non-linear SVMs) are used. Nonetheless, in our projects we are mainly using linear SVMs, so kernel SVMs are beyond this scope.

![Figure 6.3.1.1. Linear Separators](image-url)
8. **TRAINING AND TESTING PHASE**

8.1. **Process**

After performing the black and white example, and knowing how to retrieve the \( w \) and the \( b \) values, we decided to apply the same SVM concept on our dataset. We picked 200 images (100 humans and 100 non-humans) and input them to our program in order to extract their feature vectors and use them to find the appropriate and optimal hyperplane. The hyperplane found is characterized by its \( w \) vector and its \( b \) value. Hence, we stored them in a .mat file, so as to load the model whenever we want, and to avoid repeating the “training” (hyperplane retrieval) phase every time. The hardest part of this task is that the dataset is hard to visualize since each element has 3780 dimension. But since we did the black and white example before, we got to see how it works.

Nonetheless, the basic way to check if we were getting, more or less, our wanted hyperplane was to test the retrieved hyperplane using a testing dataset. Thus, the next step is testing our SVM model on new data. Therefore, we chose a new subset of the dataset. The first testing done consisted of injecting 100 new positive images (human images) and 100 new negative images (non-human) into the program. After analyzing these images using the Histogram of Oriented Gradients method, the HOG is taken as input \( (x) \) in the hyperplane \( (y = wx + b) \), then, we get the \( y \) value. After getting the predicted \( y \) value, we get to compare it to the actual \( y \) value.

8.2. **Metrics to Assess Performance**

Using the actual label and the predicted one for the 200 images, we were able to build our confusion matrix.

A confusion matrix is a method for depicting the performance of a model, and it is usually in the format below.

<table>
<thead>
<tr>
<th>Predicted: Yes</th>
<th>Predicted: No</th>
</tr>
</thead>
</table>

33
Using the results recorded in this confusion matrix, we get to compute the **accuracy**, the **precision**, the **recall**, the **specificity** and the **F1-score** (F-measure) of our model.

- The accuracy is simply the proportion of the correct predictions over the total predictions. [35]
- The precision is the fraction of the correct positive predictions over all the positive predictions. [35]
- The recall is the ratio of the correct positive predictions over the sum of the entire actual positives. [35]
- The specificity measures the ratio of the true negative predictions over the entire negative predicted set. [36]
- F-measure is in function of both the recall and the precision, and in other words, a function of false positives and false negatives. [35]

<table>
<thead>
<tr>
<th>Actual: <strong>Yes</strong></th>
<th><strong>TP</strong> (True Positive)</th>
<th><strong>FN</strong> (False Negative)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual: <strong>No</strong></td>
<td><strong>FP</strong> (False Positive)</td>
<td><strong>TN</strong> (True Negative)</td>
</tr>
</tbody>
</table>

**Table 7.2. Accuracy, precision, recall, specificity, and F-measure formulas.**

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>( \frac{(TP+TN)}{(TP+FP+FN+TN)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>( \frac{TP}{(TP+FP)} )</td>
</tr>
<tr>
<td>Recall</td>
<td>( \frac{TP}{(TP+FN)} )</td>
</tr>
<tr>
<td>Specificity</td>
<td>( \frac{TN}{(TN+FP)} )</td>
</tr>
<tr>
<td>F-measure</td>
<td>( \frac{2*(Recall \times Precision)}{(Recall + Precision)} )</td>
</tr>
</tbody>
</table>

**8.3. Results**

The testing is performed several times for several values of C.
8.3.1. Testing Phase 1

8.3.1.1. No C (Hard margin):
Total CPU time taken by training (finding hyperplane) is 279.75 seconds
Total CPU time taken by testing is 295.34 seconds

Table 7.3.1.1-1. No C Confusion matrix

<table>
<thead>
<tr>
<th>Predicted: Yes</th>
<th>Predicted: No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual: Yes</td>
<td>54 TP (True Positive)</td>
</tr>
<tr>
<td>Actual: No</td>
<td>5 FP (False Positive)</td>
</tr>
</tbody>
</table>

Table 7.3.1.1-2. No C Performance

<table>
<thead>
<tr>
<th>Measure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.745</td>
</tr>
<tr>
<td>Precision</td>
<td>0.915254</td>
</tr>
<tr>
<td>Recall</td>
<td>0.54</td>
</tr>
<tr>
<td>Specificity</td>
<td>0.95</td>
</tr>
<tr>
<td>F-Measure</td>
<td>0.679246</td>
</tr>
</tbody>
</table>

8.3.1.2. C=1
Total CPU time taken by training (finding hyperplane) is 301.97 seconds
Total CPU time taken by testing is 330.22 seconds

Table 7.3.1.2-1 C=1 Confusion matrix
Table 7.3.1.2-2 C=1 Performance

<table>
<thead>
<tr>
<th>Measure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.745</td>
</tr>
<tr>
<td>Precision</td>
<td>0.915254</td>
</tr>
<tr>
<td>Recall</td>
<td>0.54</td>
</tr>
<tr>
<td>Specificity</td>
<td>0.95</td>
</tr>
<tr>
<td>F-Measure</td>
<td>0.679246</td>
</tr>
</tbody>
</table>

8.3.1.3. C=0.01

Total CPU time taken by training (finding hyperplane) is 314.343750 seconds
Total CPU time taken by testing is 299.593750 seconds

Table 7.3.1.3-1 C=0.01 Confusion matrix

<table>
<thead>
<tr>
<th>Predicted: Yes</th>
<th>Predicted: No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual: Yes</td>
<td></td>
</tr>
<tr>
<td>47 TP (True Positive)</td>
<td>53 FN (False Negative)</td>
</tr>
<tr>
<td>Actual: No</td>
<td></td>
</tr>
<tr>
<td>2 FP (False Positive)</td>
<td>98 TN (True Negative)</td>
</tr>
</tbody>
</table>

Table 7.3.1.3-2 C=0.01 Performance

<table>
<thead>
<tr>
<th>Measure</th>
<th>Value</th>
</tr>
</thead>
</table>
### 8.3.1.4. C=0.5

Total CPU time taken by training (finding hyperplane) is **263.31 seconds**

Total CPU time taken by testing is **264.22 seconds**

**Table 7.3.1.4-1. C=0.5 Confusion matrix**

<table>
<thead>
<tr>
<th>Actual: Yes</th>
<th>Predicted: Yes</th>
<th>Predicted: No</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP (True Positive)</td>
<td>54</td>
<td>46 FN (False Negative)</td>
</tr>
<tr>
<td>FP (False Positive)</td>
<td>5</td>
<td>95 TN (True Negative)</td>
</tr>
</tbody>
</table>

**Table 7.3.1.4-2. C=0.5 Performance**

<table>
<thead>
<tr>
<th>Measure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.745</td>
</tr>
<tr>
<td>Precision</td>
<td>0.915254</td>
</tr>
<tr>
<td>Recall</td>
<td>0.54</td>
</tr>
<tr>
<td>Specificity</td>
<td>0.95</td>
</tr>
<tr>
<td>F-Measure</td>
<td>0.679246</td>
</tr>
</tbody>
</table>
8.3.1.5. C=10

Total CPU time taken by training (finding hyperplane) is **259.06 seconds**

Total CPU time taken by testing is **257.56 seconds**

Table 7.3.1.5-1. C=10 Confusion matrix

<table>
<thead>
<tr>
<th></th>
<th>Predicted: Yes</th>
<th>Predicted: No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual: Yes</td>
<td>54 TP (True Positive)</td>
<td>46 FN (False Negative)</td>
</tr>
<tr>
<td>Actual: No</td>
<td>5 FP (False Positive)</td>
<td>95 TN (True Negative)</td>
</tr>
</tbody>
</table>

Table 7.3.1.5-2. C=10 Performance

<table>
<thead>
<tr>
<th>Measure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.745</td>
</tr>
<tr>
<td>Precision</td>
<td>0.915254</td>
</tr>
<tr>
<td>Recall</td>
<td>0.54</td>
</tr>
<tr>
<td>Specificity</td>
<td>0.95</td>
</tr>
<tr>
<td>F-Measure</td>
<td>0.679246</td>
</tr>
</tbody>
</table>

8.3.2. Testing Phase 2

In this testing second phase, we are making changes in the properties of our testing dataset. For instance, we are testing for images that are flipped upside-down or flipped to the side.
Since, previously, a non-existing C value gave the highest accuracy; we will try it on this new data set.

8.3.2.1. Flipped upside-down and flipped to the side images

Total CPU time taken by training (finding hyperplane) is \textbf{259.06 seconds}

Total CPU time taken by testing is \textbf{257.56 seconds}

Table 7.3.2.1-1. Flipped upside-down and to the side Confusion matrix

<table>
<thead>
<tr>
<th>Predicted: Yes</th>
<th>Predicted: No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual: \textbf{Yes}</td>
<td>46 TP (True Positive)</td>
</tr>
<tr>
<td>Actual: \textbf{No}</td>
<td>5 FP (False Positive)</td>
</tr>
</tbody>
</table>

Table 7.3.2.1-2. Flipped upside-down and to the side Performance

<table>
<thead>
<tr>
<th>Measure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.705</td>
</tr>
<tr>
<td>Precision</td>
<td>0.901961</td>
</tr>
<tr>
<td>Recall</td>
<td>0.46</td>
</tr>
<tr>
<td>Specificity</td>
<td>0.95</td>
</tr>
<tr>
<td>F-Measure</td>
<td>0.609272</td>
</tr>
</tbody>
</table>

This method was tested on a set of 200 images (100 humans, 100 non-humans) and the results averaged towards 73.5% accuracy, which makes sense, since the Dalal and Triggs paper predicts about 80% accuracy, which is pretty close.
Moreover, one of the important key points to note is the fact that the C value used by the Dalal and Triggs paper, which equals 0.01, resulted in the lowest accuracy (72.5%) among all the tested values of C. On the other hand, the hard margin (no C), and C=0.5, 1, 10 gave the highest accuracy (74.5%).

9. LOCAL BINARY PATTERN

However, although the accuracy was close to the expected accuracy from the paper, as an attempt to make it even higher, I was able to find several papers that advice and recommend using Local Binary Pattern with HOG claiming that the accuracy of the results is even higher than Dalal and Triggs’ results. That is why we decided to further implement this method by adding the Local Binary Pattern (LBP) component.

First, to define Local Binary Pattern, it is a method used in image recognition. It mainly revolves around analyzing grey images and textures. The concept behind the LBP approach is taking a particular pixel, called the center pixel, and comparing it to k pixels that fall on its surrounding circle of radius R. While performing the comparison, if the center pixel’s value is greater than a surrounding pixel, the result of that comparison is a 0, and if the inverse, it is a 1.

In our case, the number of surrounding pixels we chose is 8, and the radius is 1. [38]

![Figure 8.1. An example matrix and its corresponding comparison matrix](image)

Then, after performing this comparison and retrieving these values, the result is 00010010. We convert the binary number to a decimal one, by multiplying each cell i by $2^{i-1}$.

In other words,

$$LBP = 2^0 * 0 + 2^1 * 1 + 2^2 * 0 + 2^3 * 0 + 2^4 * 0 + 2^5 * 0 + 2^6 * 0 + 2^7 * 0 = 18$$

However, the general rule of LBP can be stated as follows,
$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) \ast 2^p$$ \[37]\); where,

- \(s(x)\) is a function that returns 1 if \(x \geq 0\) and 0 if \(x < 0\).
- \(P\) is the number of surrounding pixels to compare to.
- \(R\) is the radius.
- \(g_p\) is the value of \(p^{th}\) neighbour of the central pixel.
- \(g_c\) is the value of the central pixel.

After repeating this process of LBP computation for all pixels of an image, for each cell (8x8 pixel), we compute a histogram of 256 bin, in which each bin represents the number of pixels that have that particular value. Then, similar to the HOG, we perform block normalization. After doing so, we end up building our image LBP feature vector.

Following this, we merge both our previously computer HOG feature vector with the LBP feature vector to get a new bigger feature vector, which we use in our classification.

**10. TESTING WITH LBP**

We test our new feature vector using the same data previously used, as well as additional tests using reversed images and on the side images and without C value. The results were all the same and as follow.

Total CPU time taken by training (finding hyperplane) is **2213 seconds**

Total CPU time taken by testing is **2250 seconds**

<table>
<thead>
<tr>
<th>Table 9.1. HOG-LBP Confusion matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual: <strong>Yes</strong></td>
</tr>
<tr>
<td><strong>TP</strong> (True Positive)</td>
</tr>
<tr>
<td><strong>FP</strong> (False Positive)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 9.2. HOG-LBP Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measure</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Accuracy</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>Precision</td>
</tr>
<tr>
<td>Recall</td>
</tr>
<tr>
<td>Specificity</td>
</tr>
<tr>
<td>F-Measure</td>
</tr>
</tbody>
</table>

The training and tests were done using the same dataset as the HOG-only method, and the results were, as expected, better in terms of accuracy. An accuracy of 100% was achieved over three 200 images set (100 humans, 100 non-humans) experiments: regular images, reversed images, and flipped to the side images.

11. STEEPLE ANALYSIS

11.1. Societal

The program we are implementing tries to provide a tool for classification of images, to know and sense the existence of humans in images and scenes. This is beneficial for developing tools that need to perform certain tasks when encountering humans. For example, and according to Paul, Haque, & Chakraborty, human detection can be used for “abnormal event detection, human gait characterization, congestion analysis, person identification, gender classification and fall detection for elderly people” [1] and so on. These examples along with several others are a mere example of the benefits that such programs can offer society.

11.2. Ethical

Computer ethics have attracted a lot of attention in the last few years as incorporation of more technology in daily life has opened many avenues for unethical uses of this technology. Nonetheless, many ethical issues have been arising lately in the field of computer vision, especially with facial recognition that has known a huge popularity wave. The issues vary from identity theft to privacy invasion. Nonetheless, the program we are developing is not
concerned with facial recognition, but more with human detection. In other words, the program’s main task is spotting humans in a picture. The program does not go beyond this; hence, such ethical issues are unlikely to arise. Moreover, in terms of our tools, we will follow ethical practice of not using any pirated or compromised software -- rather we are aiming at using the free or open-source versions whenever possible.

11.3. **Technical**

The program develops an image processing and recognition technique that could be used in analyzing the presence of humans in images and frames and could be used in new technologies such as self-driven cars and machines. It can also be used, if trained on a different dataset other than humans, in identifying other objects. This object identification could be further used in applications to help blind people identify objects around them.

Moreover, on the software side, we were able to use Octave, which is an open source version of Matlab. Nonetheless, Octave is still no yet fully developed and various built-in functions, which already exist in Matlab, were missing from Octave.

11.4. **Environmental**

Our program can be used in order to reduce energy consumption for machines that have to function when facing a human. Having our program used, the machine can put itself into power saving mode and allocates the minimum amount of energy required to perform human detection. This decrease in energy consumption can also motivate owners to use more renewable energies.

11.5. **Political**

Some political issues may arise in case the program is used for malicious uses. For instance, if used in military robots, some issues regarding civilians’ safety may begin between either two countries that are already in war or even between non-warring countries. This can worsen existing conflicts and can leave for new political confrontations and create tension.

11.6. **Legal**

The program could spark the creation of new regulations that ensure its correct use and that ban any malicious party from using it to serve its twisted intentions.
11.7. **Economic**

On one hand, the appropriation of such program can reduce costs for companies requiring the use of human identifying machines, and reduce the utilities cost by decreasing the energy consumption.

On the other hand, since the program can be used to identify the presence of humans in an automated way, this may lead to the big problem that is being caused nowadays by automation, which is the layoff of employees. Knowing that one machine can outperform several employees, the human resources start to seem more expensive and less performing. Hence, in many industries many employers choose to reduce their costs by dismissing their employees and going for software and machines, which may contribute to the increase of the unemployment rate.

12. **PROJECT IMPACT**

As said before, many ethical issues have been arising lately in the field of computer vision, especially with facial recognition which is becoming more and more wide-spread. The issues vary from identity theft to privacy invasion. Nonetheless, the program we are developing is not concerned with facial recognition, but more with human detection. In other words, the program’s main task is spotting humans in a picture. The program does not go beyond this; hence, such ethical issues are unlikely to arise. However, other ethical issues may surface, such as the use of the program to attack humans. One of the famous ideas that have been talked about in the military field is military robots. Hence, using this project’s concept, a military robot can identify the presence of humans and start attacking them. This military robot may not even be able to recognize between civilians and armed people, thus putting in danger and targeting everyone.

Another ethical issue would be the lack of precision and the error margin of the program. No matter how small this margin is, it can put humans in danger. For instance, if a machine is supposed to stop once facing a human, the odds are that one day it may not stop. The
probability may be very low, for example like 1 in a million. Yet, the remaining question is “are we willing to allow a 1 in a Million casualty, for the sake of a Million - 1 success?” The nature of the subjects, with whom our program deals, is sensitive. Human life is irreplaceable; thus, with this program and other technological novelties working in contact with human, it remains unanswered whether we should go for the common good and global welfare at the expense of some casualties.

On the other hand, concerning the ethical implications of conducting research and using ideas of papers already published, our research makes sure to clearly give credit to the owners of those ideas. Moreover, another point to question is the authenticity and reliability of the research. Falsification of results and overfitting results, to support particular conclusions, can be dangerous, if overlooked. However, in our case, we made sure to perform our own tests and our own experiments to further verify the accuracy of the information referenced.

From the social side, knowing that we, for sure, have machines that are working automatically, it is important for the machines to be able to detect the presence of humans so as to stop, slow down, or react in a certain way. Having such machines can help people in better performing daily tasks, and allow them to avoid several incidents such as work accidents due to machines not stopping when facing workers. Moreover, by training the program on different datasets of different objects can help humans do the repetitive task of tagging images of common objects, and/or identifying an object for a researcher or even an amateur hobbyist or even help blind people identify objects in their surroundings.

Furthermore, since our program uses support vector machines as classifiers, and since support vector machines have started developing around the 80’s, after the first linear perceptron was made in the 60’s, it makes them quite a new topic in the field of mathematics and machine learning. For instance, there is a lack of simplified explanation of the concepts behind support vector machines. Hence, thanks to this project we were able to further investigate the mathematical background and write it in a more simplified and intuitive way. Such a task would allow for more understanding of support vector machines and could even allow for even more sophisticated models that could allow for a state of the art. Furthermore, this could
also push for more usage of support vector machines in developing various other projects that aren’t necessarily linked to ours.

13. CONCLUSIONS

In this project, we implemented a human detection program by trying and replicate already existing work. As much as replication seems meaningless, it is actually very important in further developing the techniques and aiming for higher accuracy. An example of this importance is the fact that the replication of the Dalal and Triggs work is the one that first led to the combination of the local binary pattern and histogram of oriented gradient, which led to a higher accuracy. Replication can open the door for further development, creativity, and critical thinking. It can allow for more understanding of concepts and for more efforts to be put in testing the ideas and methods, modifying them, and maybe even adding more to them. That is why in our case, we used two methods. First, using solely the histogram of gradients technique, we computed the histograms, retrieved our feature vectors, and then used them to find the optimal hyperplane that would classify our data best. This method was tested on a set of 200 images (100 humans, 100 non-humans) and the results averaged towards 73.5% accuracy. Nonetheless, in an attempt to enhance the accuracy, we added the local binary pattern method and combined it with the histogram of gradient approach, as it was mentioned in several papers that the combination of the two methods can increase the accuracy even in cases of occlusion. [39]

The training and tests were done using the same dataset as the HOG-only method, and the results were, as expected, better in terms of accuracy. An accuracy of 100% was achieved over three 200 images set (100 humans, 100 non-humans) experiments: regular images, reversed images, and flipped to the side images. This clearly shows how the replication of the Dalal and Triggs work did not stop us from further enhancing our program, but it was a door that allowed us to discover new concepts such as LBP. Moreover, one of the important key points to note is the fact that the C value used by the Dalal and Triggs paper, which equals 0.01, resulted in the lowest accuracy (72.5%) among all the tested values of C. On the other hand, the hard margin (no C), and C=0.5, 1, 10 gave the
highest accuracy (74.5%). This highlights how the replication doesn’t always have to be total, and how the C value depends on the data being used to define the hyperplane and there is no fit-all value for C, even when having two programs within the same context.

Besides the results of our program, a major thing to highlight is the massive role that the incorporation of linear algebra and multivariable calculus played in both the understanding of this project, and its completion. Various concepts seen in the courses of Linear Algebra and Multivariable Calculus were crucial to the implementation of both HOG, and SVMs. Moreover, other mathematical background, which was not necessarily part of any courses taken in my undergraduate studies, was needed. This is a proof of how the project is in fact multidisciplinary in terms of use of concepts and notions. It combined different types of math along with computer science notions to eventually result in an image recognition program.

14. FUTURE WORK

Concerning the future work, the main focus would be on maintaining a high accuracy while decreasing the runtime. One of the issues faced with the HOG-LBP approach is dimensionality. HOG method gave 3780 dimensions per feature vector, however, the LBP method alone gave 107520 dimensions per LBP feature vector, so combined with HOG, and we get an 111300 feature vector. Such high dimensionality creates an issue in terms of runtime. In order to decrease the runtime, one of the ways would be reducing the dimensions of the feature vectors. Having a reduced feature vector would reduce the time needed for computations and thus, it may allow for using this program with real time detection. This latter is, also, one of the interesting extensions of the project, and which could create a great update to the program. Actually, in an attempt to trespass this stumbling block, we already tried implementing Principal Component analysis in order to reduce the dimensionality of our feature vectors. We successfully managed to reduce it for the HOG feature vector, but we could not apply it on the LBP feature vector since its dimensions were too large to allow for the software’s (Octave) covariance computation. Hence, it would be good to find different techniques that can be used to reduce size vectors with big dimensions in an optimal way and which could still allow us to get high accuracy. In other words, we need to find techniques
that can eliminate dimension that do not add much information gain to our data and that can still leave our data linearly separable.
15. REFERENCES


