OBJECT DETECTION SYSTEM USING HAAR-CLASSIFIER

WAN NAJWA BINTI WAN ISMAIL

This thesis is submitted as partial fulfillment of the requirements for the award of the Bachelor of Electrical Engineering (Hons.) (Electronics)

Faculty of Electrical & Electronics Engineering

Universiti Malaysia Pahang

MAY, 2009

“All the trademark and copyrights use herein are property of their respective owner. References of information from other sources are quoted accordingly; otherwise the information presented in this report is solely work of the author.”

Signature	: ____________________________
Author	: WAN NAJWA BINTI WAN ISMAIL
Date	: 11 MAY 2009

To

My beloved parents

and my siblings

“ who offered me unconditional love and support

throughout the course of this thesis”

Acknowledgement

First and foremost, I wish to express my gratitude to the project's supervisor, Mr. Mohd Zamri bin Ibrahim for his mentorship throughout the course of my graduate studies. This work would not be possible without Mr Zamri's ideas, dedications, supports, analogies and advices over the year.

I would like to extend my gratitude to those who gave the possibilities for me to complete this project especially to my beloved parents for giving such a great support and encouragement for me financially or technically through the duration of my studies.

Special thanks also to all my graduate friends, especially to those group members under Mr Zamri's supervision who has sharing the literature and invaluable assistance. I would also like to convey thanks to the administrator who gave me opportunity to find research material easily.

Abstract

The invention of new algorithms had encouraged to the reinforcement of image processing's application. An algorithm for the design object detection systems is presented. Haar-classifier is utilized as the algorithms for this object detection system. The exertion of Haar-Classifier had boosted to the upgrade system which is faster and more accurate. In this system, Haar-Classifier is conjunct with the Adaboost machine learning algorithms wherefore the performance of the system is upgraded. Development of this project is categorized into two phase which are training phase and execution phase. Training phase use OpenCV utilities such as haartraining.exe to train the object by calculating the object's weak constraints. This is for the purpose of finding the different features of the object of interest. The list of these weak constraints is converted to the xml file to be included in the coding which had been developed using Visual Studio 2005. The execution process will result on the detection process of object of interest. System will detect rounded image in any image which had been included in the system itself. Object detection system using Haar-classifier algorithm can perform best performance of high detection rate and high level of accuracy rate.

Abstrak

Penciptaan algoritma baru telah menggalakkan kepada perkembangan aplikasi sistempemprosesan imej. Algoritma untuk mereka sistem pengesanan objek telah diperkenalkan. Pengklasifikasi Haar telah digunakan sebagai algoritma untuk sistem pengesanan objek ini. Penggunaan pengklasifikasi Haar telah meningkatkan prestasi sistem supaya lebih cepat dan tepat. Untuk tujuan meningkatkan kadar pengesanan, pengklasifikasi Haar telah digabungkan dengan kaedah *Adaboost* yang menjadi punca kepada peningkatan kadar pengesanan objek. Pembentukan sistem ini terbahagi kepada dua bahagian iaitu fasa latihan dan fasa perlaksanaan. Fasa latihan menggunakan utiliti *OpenCV* seperti "haartraining.exe" untuk melatih objek yang hendak dikesan dengan cara mengira ciri kelemahan sesuatu objek itu. Hal ini bertujuan untuk mencari ciri-ciri berlainan yang ada pada sesuatu objek itu. Senarai cirri-ciri kelemahan ini ditukar kepada fail xml untuk dimasukkan ke dalam kod yang telah dibuat menggunakan perisian *Visual Studio 2005*. Fasa perlaksaan akan menghasilkan proses pengesanan objeck yang dikehendaki. System akan mengesan image berbentuk bulat dalam sebarang gambar yang telah dimasukkan ke dalam system. Sistem Pengesanan Objek menggunakan pengklasifikasi Haar algoritma mampu melaksanakan hasil yang bagus dengan kadar pengesanan dan kadar ketepatan yang tinggi.

TABLE OF CONTENTS

List of Figures

List of Table

List of Abbreviations

OpenCV – Open Source Computer Vision

Adaboost – Adaptive Boost

HCI -- Human-Computer Interaction

SFM -- Structure From Motion

IDE -- Integrated Development Environment

GUI – Graphical User Interface

VB.NET – Visual Basic. Net

Png -- Portable Network Graphic format

Xml – Extensible Markup Language

List of Appendices

CHAPTER 1

INTRODUCTION

Development of computer system and technology has encouraged on the development of intelligence on new technologies. Object detection system is developed as a contribution due to help humans in daily life. This system approach can be applied to robotic system and surveillance system. Designing object detection's system is a process to determine the location and the region of the object in a digital image. This system will only detect object of interest and ignore any others objects. The algorithm use in this project is Haar-Training which is used to calculate the threshold of the object of interest in order to obtain a new classifier. This process will be done using OpenCV utilities. Visual Studio 2005 software has been used to test the ability of the new classifier.

1.1 Problem Statement

This project uses the better performance of algorithms such as Haar-Training. for the purpose to get reasonable accuracy rate. Haar-Training could be the better solution as it has a combination with the Adaboost method. The cascade of Adaboosted (Adaptive boost) classifiers can achieve both accuracy and speed. The algorithm can achieve high detection accuracy and approximately 15 times faster than any previous approaches.

1.2 Objective

The purpose of doing this project is not basically just to detect objects in an image but also for some other purpose which are:

i) To utilize the object's region in a digital images.

ii) To leverage positive object's classifier using Haar-Training.

iii) To develop an object detection system using OpenCV and Visual Studio 2005 software.

1.2.1 Work Scope

There are few work scopes that related to this project which are first to develop a system that can be used to localized object features in a digital colored image. This is for the purpose to identify the location of desired object in an image which also had consisted of other images. Second is to develop a system by using Visual Studio 2005 software for the better performance. Visual Studio is the best software of interfacing OpenCV with GUI interface. Last but not least is to classify the weak and strong features from the desired object using Haar-Classifier to compare with the image's threshold in the xml file (database) for the detection process.

1.4. Project Overview

In general, object detection system's purpose is to detect desired object in any still image. In this project, the desired object input is circle image such as ball, circle and etc. The overall flow of this system can be described as following. A scanned image will be compared to the object models in the system's dataset and the system will detect whether the object exist is or not in an image. This system needs to go through certain important method which are first the collection of image, second is the training process using Haar-Training method, next is coding implementation and last is to match the object with the dataset which in this case was the cascaded image threshold which is contained in the xml file or the purpose to determine whether the desired object is exist or not.

1.5. Thesis Outline

This thesis is organized as follows:

Chapter 1 will describes the introduction of this system, the purpose of doing this project, the problem statement, the work scope and brief explanation of project's system flow.

In **Chapter 2**, the review about the information find on all the material or data used in the development of the system will be shown.

Explanation of all the methods use in development of this system and the step by step solution on developing training part and execution part will be described in **Chapter 3.**

Chapter 4 includes all the results followed with the explanation about the results after all the development process has done.

Chapter 5 is the last chapter and it will show the summary after all and come up with some recommendations for some improvements.

CHAPTER 2

LITERATURE REVIEW

This chapter will review on the information gathered in developing the object detection system. The information accumulated is all the basic notification used in order to develop the system including the basic definition of system approach, the algorithms and the basic process of the system.

2.1 Digital Image Processing

Digital image processing is the use of computer algorithms to perform image processing on digital images **[6].**

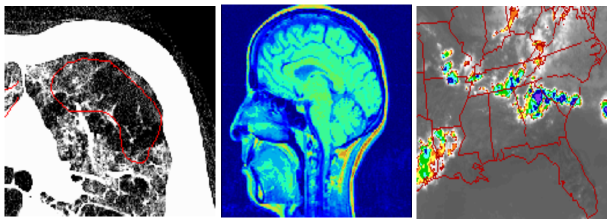
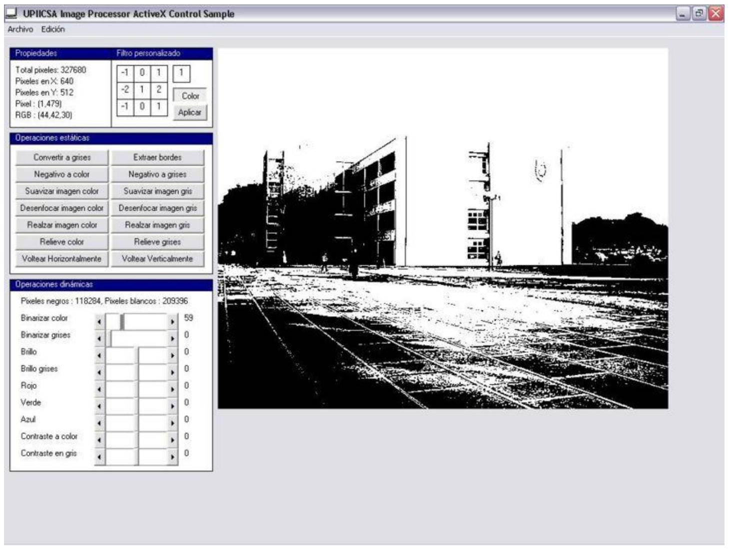


Fig 2.1: Application of Digital Image Processing

2.1.1 Image processing

Image processing is a physical process used to convert an image signal into a physical image. The image signal can be either digital or analog. The actual output itself can be an actual physical image or the characteristics of an image. The most common type of image processing is photography.



Total pixeles: 327680
Pixeles en X: 640
Pixeles en Y: 512
Pixel : (1,479)
RGB : (44,42,30)

-1	0	1	1
-2	1	2	
-1	0	1	

Pixeles negros : 118284, Pixeles blancos : 209396

Binarizar color	59
Binarizar grises	0
Brillo	0
Brillo grises	0
Rojo	0
Verde	0
Azul	0
Contraste a color	0
Contraste en gris	0

Fig 2.2: Monochrome black/white image

2.1.2 Algorithms

Algorithm is a finite sequence of instructions, an explicit, step-by-step procedure for solving a problem, often used for calculation and data processing. It is formally a type of effective method in which a list of well-defined instructions for completing a task.

In its most general sense, an algorithm is any set of detailed instructions which results in a predictable end-state from a known beginning. Algorithms are only as good as the instructions given, however, and the result will be incorrect if the algorithm is not properly defined [7].

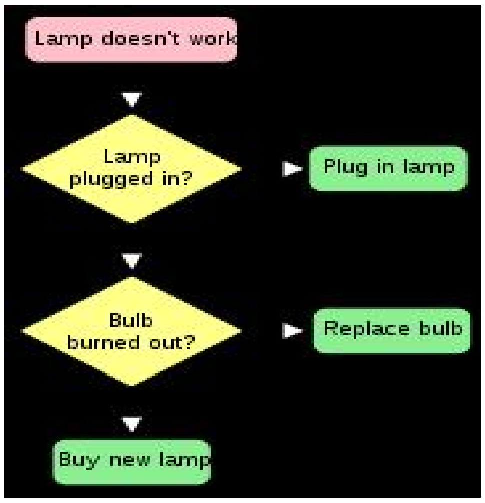


Fig 2.3: Algorithms can be represented in flowchart

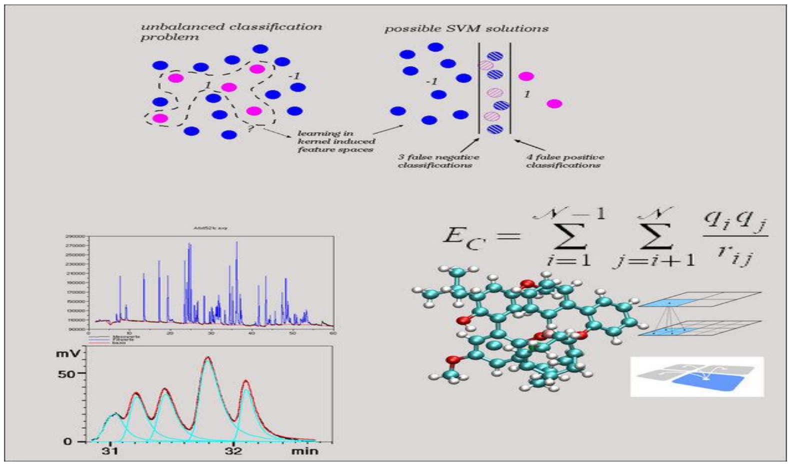

unbalanced classification problem
possible SVM solutions
learning in kernel induced feature spaces
3 false negative classifications
4 false positive classifications
$E_C = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{q_i q_j}{r_{ij}}$
mV
50
0
31
32
min

Fig 2.4: Example of Numerical Algorithms

2.2 Edge Detection

An improved algorithm based on frame difference and edge detection is presented for moving object detection. First of all, it detects the edges of each two continuous frames by Canny detector and gets the difference between the two edge images. And then, it divides the edge difference image into several small blocks and decides if they are moving areas by comparing the number of non-zero pixels to a threshold. At last, it does the block-connected component labeling to get the smallest rectangle that contains the moving object [8]

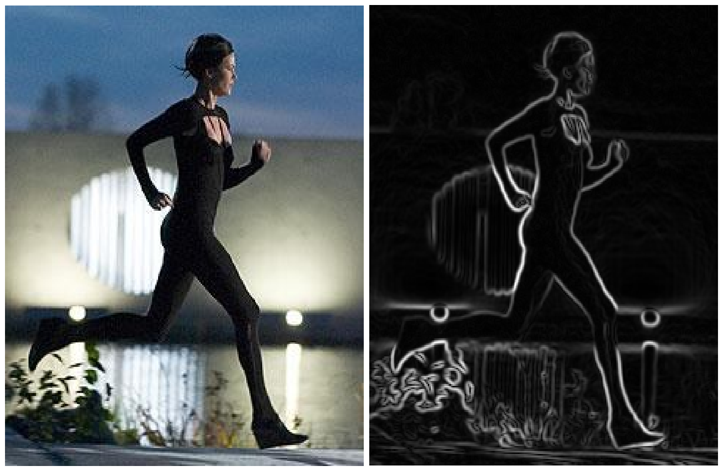


Fig 2.5: Edge Detection Process

2.3 Color conversion

Threshold is an image segmentation to convert grayscale to binary image. During the threshold process, individual pixels in an image are marked as “object” pixels if their value is greater than some threshold value (assuming an object to be brighter than the background) and as “background” pixels otherwise. This convention is known as *threshold above*. Variants include *threshold below*, which is opposite of threshold above; *threshold inside*, where a pixel is labeled "object" if its value is between two thresholds; and *threshold outside*, which is the opposite of threshold inside. Typically, an object pixel is given a value of “1” while a background pixel is given a value of “0.” Finally, a binary image is created by coloring each pixel white or black, depending on a pixel's label.

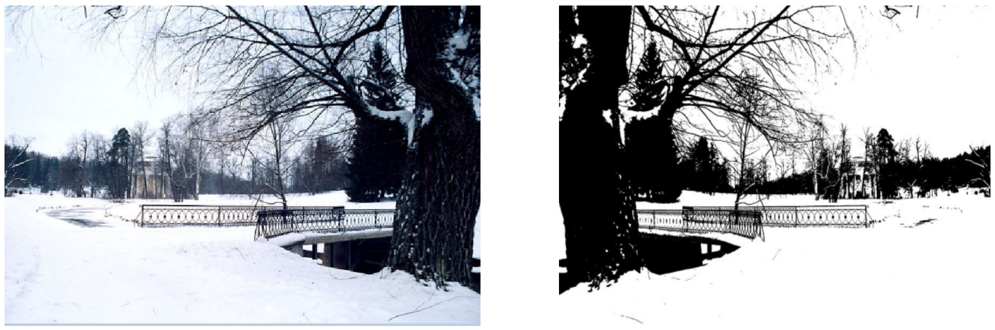


Fig 2.6: Result image of grayscale to binary color conversion process.

2.4 Haar-Classifier

Object detection system is given an image patch of known size or a feature and is to decide whether this features stemmed from an object, or a non object. For the purpose to get a reasonable accuracy of object detection performance, the Haar- classifier is applied to this system

Haar-Classifier encodes the existence of oriented contrasts between regions in the image. A set of these features can be used to encode the contrasts exhibited by an object. The detection technique is based on the idea of the wavelet template that defines the shape of an object in terms of a subset of the wavelet coefficients of the image.

Haar-like features are so called because they share an intuitive similarity with the Haar wavelets. Historically, for the task of object recognition, working with only image intensities (i.e. the RGB pixel values at each and every pixel of image) made the task computationally expensive. This feature set considers rectangular regions of the image and sums up the pixels in this region. The value this obtained is used to categorize images. For example, let us say we have an image database with human faces and buildings. It is possible that if the eye and the hair region of the faces are considered, the sum of the pixels in this region would be quite high for the human faces and arbitrarily high or low for the buildings [1]

The value for the latter would depend on the structure of the building, its environment while the values for the former will be more less roughly the same. We could thus categorize all images whose Haar-like feature in this rectangular region to be in a certain range of values as one category and those falling out of this range in another. This might roughly divide the set of images into ones having a lot of faces and a few buildings and the other having a lot of buildings and a few faces. This procedure could be iteratively carried out to further divide the image clusters [2].

The Algorithm use in this project is Haar-like features to find the weak constraints. There is little information that should be understood about Haar-like which are:

- Each Haar-like feature consists of two or three jointed "black" and "white rectangles:

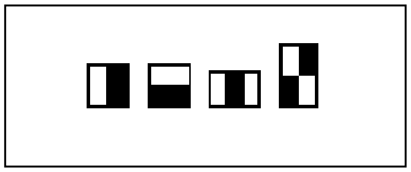


Fig 2.7 : A set of basic Haar-like features.

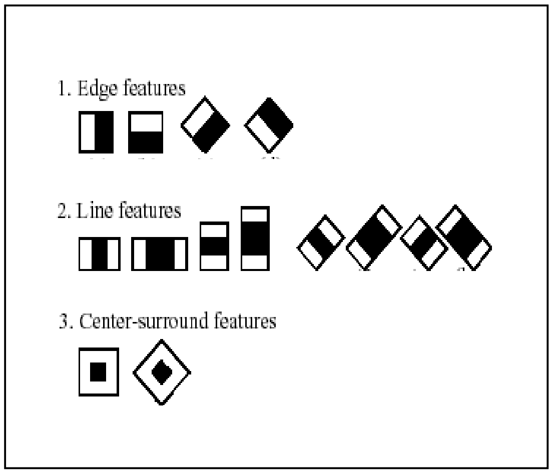


Fig 2.8: A set of extended Haar-like features.

- The value of a Haar-like feature is the difference between the sum of the pixel gray level values within the black and white rectangular regions:

f(x)= Sumblack rectangle (pixel gray level) – Sumwhite rectangle (pixel gray level)

Equ 2.1: Calculation on Haar-like feature's value.

2.5 AdaBoost

AdaBoost, short for Adaptive boosting, is a machine learning algorithm, formulated by Yoav Freund and Robert Schapire. It is a meta-algorithm, and can be used in conjunction with many other learning algorithms to improve their performance. AdaBoost is adaptive in the sense that subsequent classifiers built are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. Otherwise, it is less susceptible to the over fitting problem than most learning algorithms. AdaBoost calls a weak classifier repeatedly in a series of rounds t = 1….T. **[3]**

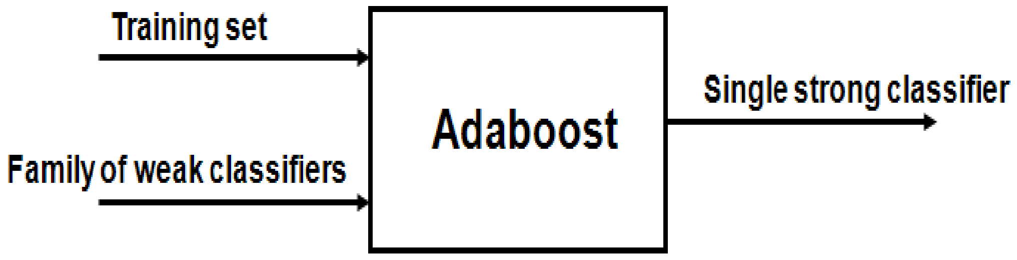


Fig 2.9: Adaboost figure simplification

2.6 OpenCV

The OpenCV library gives us a greatly interesting demo for object detection. Furthermore, it provides us programs (or functions) which they used to train classifiers for their face detection system (called Haartraining) so that we can create our own object classifiers using these functions. Example applications of the OpenCV library are Human-Computer Interaction (HCI); Object Identification, Segmentation and Recognition; Face Recognition; Gesture Recognition; Motion Tracking, Ego Motion, Motion Understanding; Structure From Motion (SFM), Stereo and Multi-Camera Calibration and Depth Computation and Mobile Robotic. Object Detection System is also the application of OpenCV. **[4]**

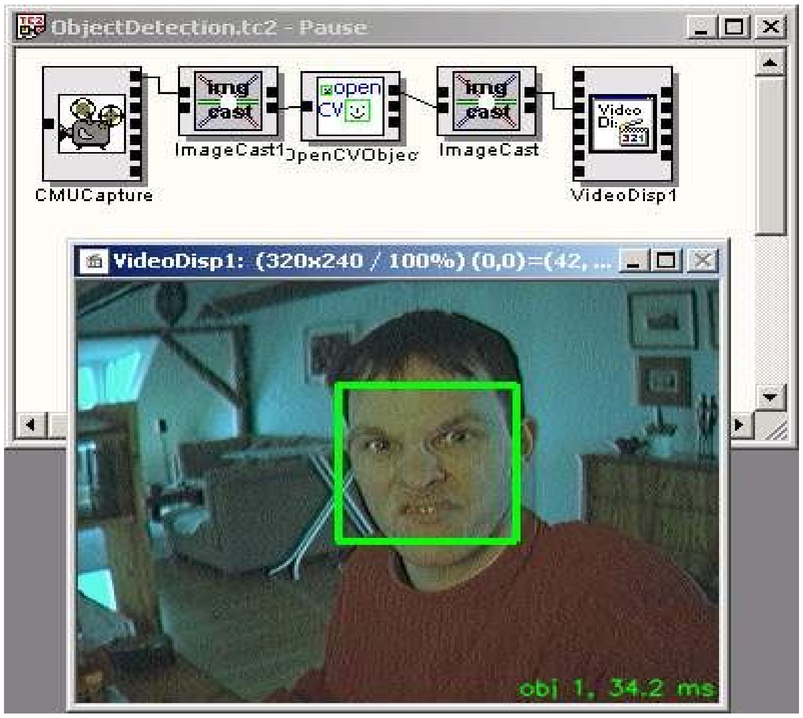

ObjectDetection.tc2 - Pause
CMUCapture
ImageCast1
OpenCVObjec
ImageCast
VideoDisp1
VideoDisp1: (320x240 / 100%) (0,0)=(42, ...
obj 1, 34.2 ms

Fig 2.10: The application of OpenCV (Face Detection)

2.7 Microsoft Visual Studio 2005

Microsoft Visual Studio is an Integrated Development Environment (IDE) from Microsoft. It can be used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight.

Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It allows plug-ins to be added that enhance the functionality at almost every level - including adding support for source control systems (like Subversion and Visual SourceSafe) to adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer).

Visual Studio supports languages by means of language services, which allow any programming language to be supported (to varying degrees) by the code editor and debugger, provided a language-specific service has been authored. Built-in languages include C/C++ (via Visual C++), VB.NET (via Visual Basic .NET), and C# (via Visual C#).

CHAPTER 3

METHODOLOGY

There are few methods contribute in developing this system. This chapter will review all the method use and how the Haar – Classifier is implemented to detect the object of interest. The trained object used in this project is rounded image.

3.1 System Framework

Object detection system required two different phase to complete the system development. First is the training phase which is work with OpenCV software. Training phase is to create a datasets from the collected images. Image must be collected and categorized in two categories which are positive image samples and negative image samples before image training process take part. Negative samples correspond to non-object images. Positive samples correspond to object of interest. There are few other steps contribute to the image training phase which are object marking , sample creating , haartraining, performance test and creating xml file. After training process is done, the execution phase is proceeded. Execution phase is to test the training using simple coding created using Visual Studio 2005.

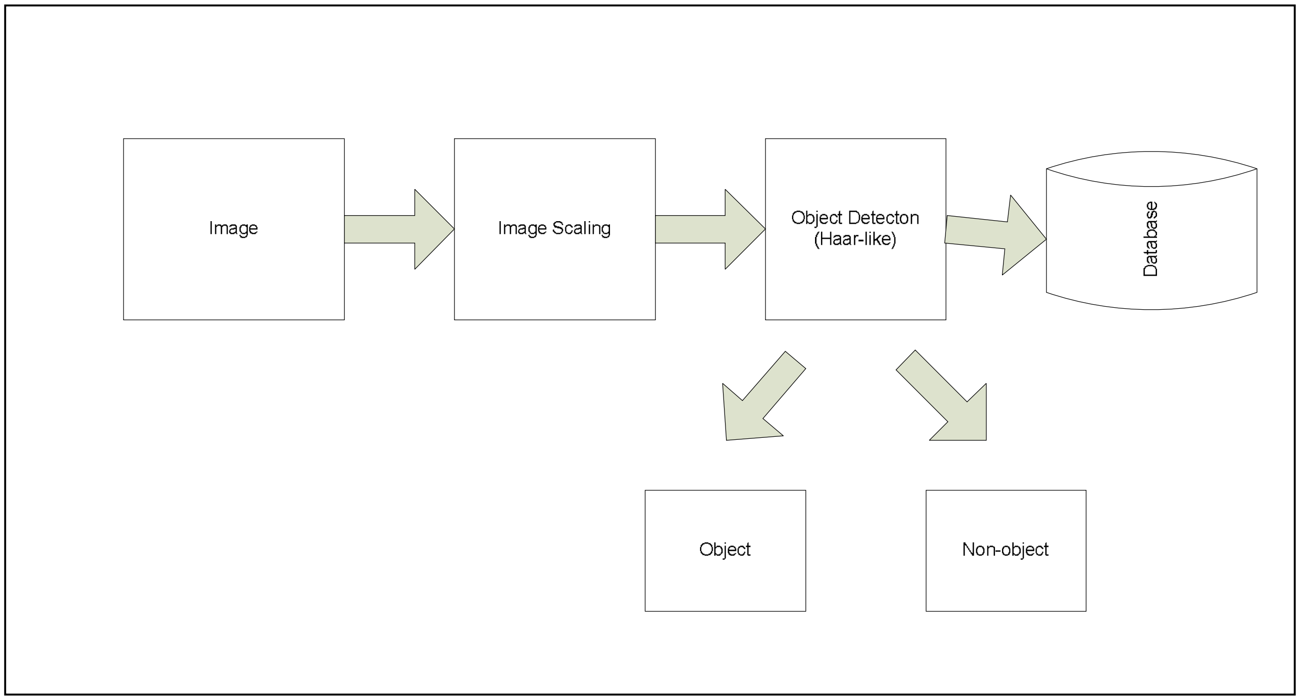


Fig 3.1: System’s Block Diagram

3.2 Flowchart of the system

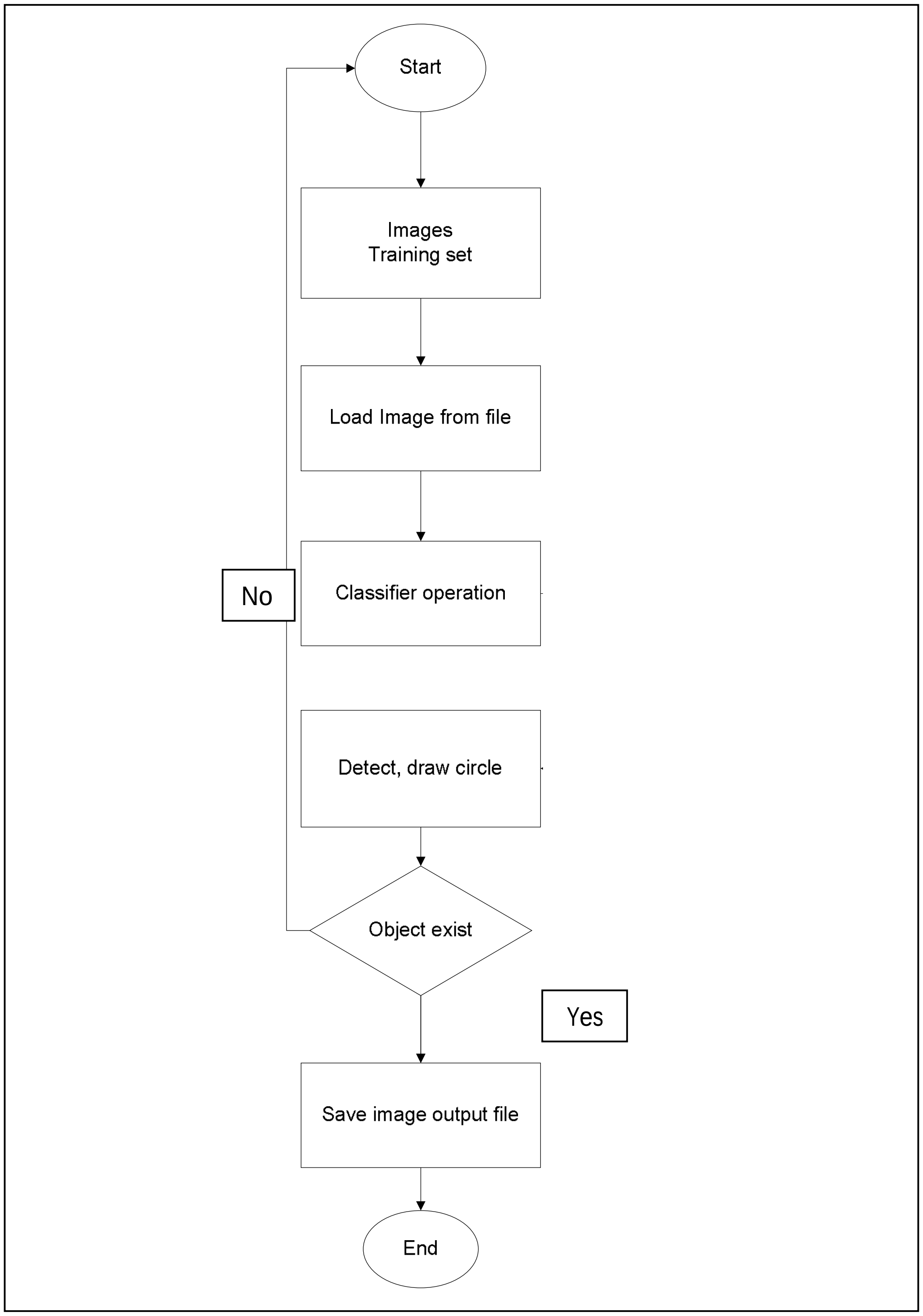


Fig 3.2 : The flowchart of the system

3.3 Block diagram of training phase

Training phase is separated to eight different processes to produce a classif ierwhich contains a list of weak constraints that can be used to differentiate the object of interest with other objects.

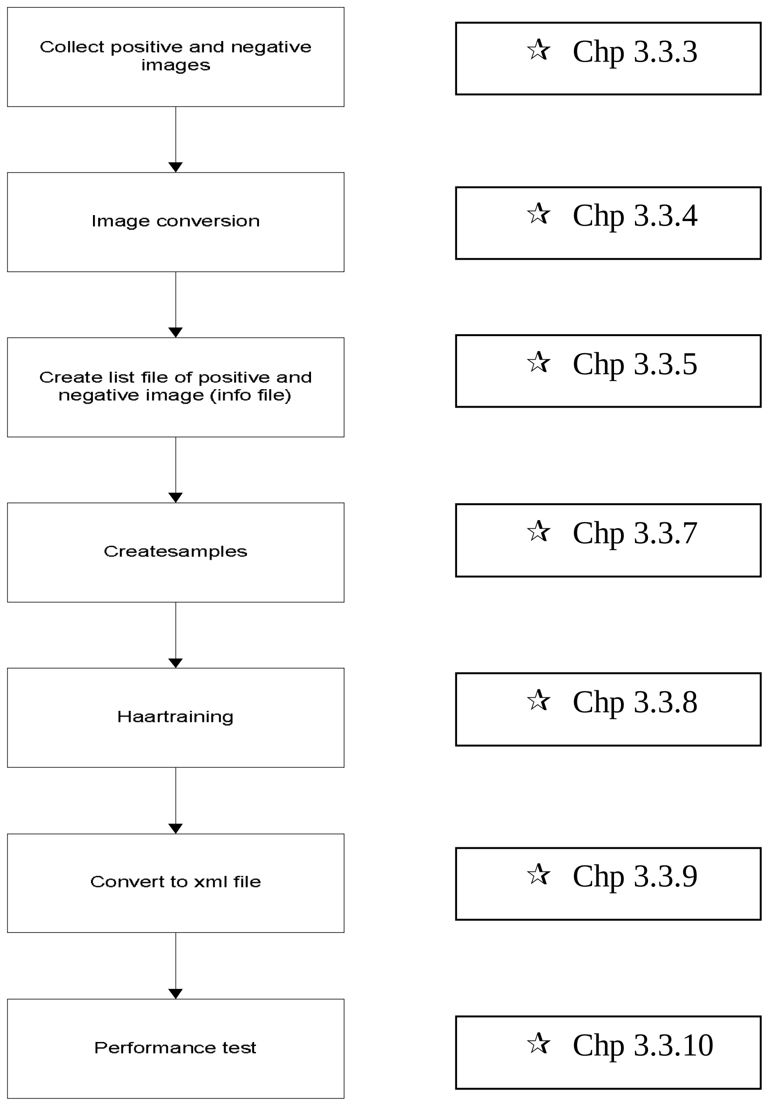


Fig 3.3 : The block diagram of the Haar-training proces

3.3.1 OpenCV installation

Training phase will mostly depend on OpenCV functional library which are createsamples.exe, haartraining.exe, convert_cascade.exe and performance.exe. These entire executable (available from the OpenCV installation directory) file had their own functions and all these functions is used during the training phase. The function of these executable files can be described as bellow:

Tab 3.1: OpenCV utilities

No.	Library	Function	Directory
1.	Createsamples.exe	To create sample from positives image collected	(C:\Program Files\OpenCV\bin\createsamples.exe)
2.	Haartraining.exe	To train sample to calculate the threshold of all samples	(C:\ Program Files\OpenCV\bin\haartraining.exe)
3.	Convert_cascade.exe	To convert file to xml file.	(C:\Program Files\OpenCV\samples\c)
4.	Performance.exe	To test the performance of trained samples.	(C:\Program Files\OpenCV\bin\performance.exe)

OpenCV must be installed first to perform all these functions. Installation of OpenCV is a very straight forward. After installation process is done, all the utilities files is ready to use and training process could be start.

3.3.2 Folder creation

For training phase, all the works must be done in one folder. To create the folder all the following steps must be considered and need to be done.

i) Create a folder in local disk C of any desirable name such as "haarcascade" which had been used in this project.

ii) Go to "C:\Program Files\OpenCV\bin\____".

iii) Copy the executable files below and place in the folder "haarcascade" (folder

1. "createsamples.exe"
2. "haartraining.exe"
3. "convert_cascade.exe"
4. "performance.exe"

3.3.3 Image collecting

Collect positive images that contain only objects of interest for example, circle image. The total of positive image collected must be up to 1000 of different images in same shape. These images can be collected by searching at Google or Yahoo image search or any other websites. Examples of positive images are shown in figure below.

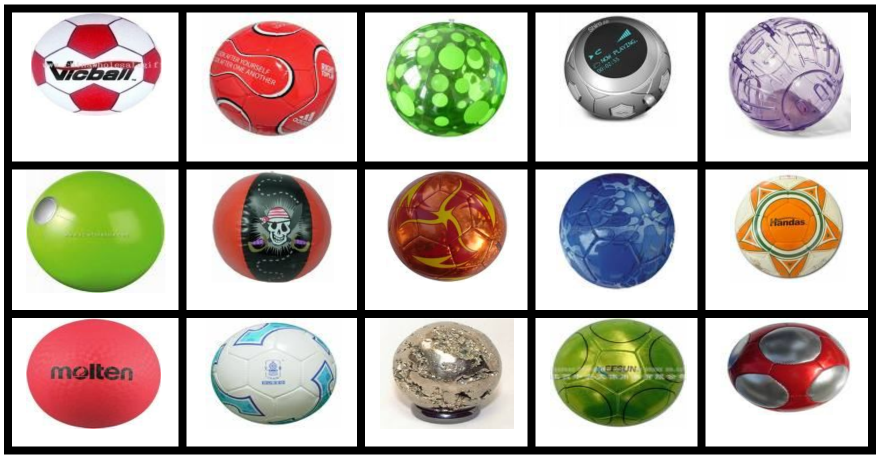


Figure 3.4: Sample training (positive samples)

Same goes to negative image where total image collected must be up to 1000 of any image which not contains any positive images. Negative image is also known as background image. Figure 5 show the example of negative images.

Fig 3.5 : Sample training (negative samples)

All positives image that has been collected must be placed in one folder. Rename the folder to any names and do same steps to negative image. The two folders of positive and negative images collected must be placed in the folder “haarcascade”.

3.3.4 Image Conversion

All images in jpeg or gif must be converted to bitmap (bmp) or Portable Network Graphic format (png). This is because "object marker.exe" which is the execution file needed to crop desired object to detect in an image could not support the file format of jpeg or gif. If marking object is done manually which means without using object marker.exe, just skip this step. Image should not convert to bitmap or png. To convert image we can either use the basic program from our computer such as "Paint" or by using image converter software such as "Pixillion".

These are the steps that can be done to convert image file format

1) **Step to convert image file format using basic computer program such as " Paint".**

   i) Go to "Start" menu on the taskbar. Click "All Programs"→ "Accessories" → "Paint".

   ii) Go to "Open" on menu taskbar and browse any file that need to convert.

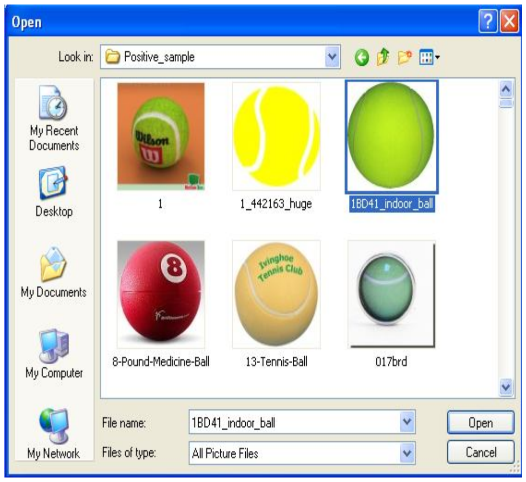


Fig 3.6: Browse file from folder.

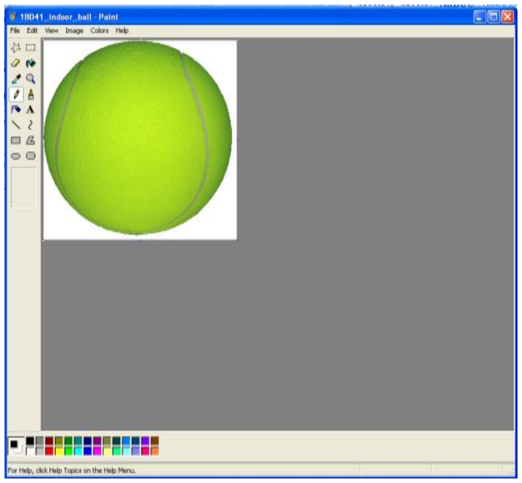

1BD41_indoor_ball - Paint
File Edit View Image Colors Help
For Help, click Help Topics on the Help Menu.

Fig 3.7: Image browsed display in Paint workspace

iii) Go to “File” → “Save as”.

iv) Change the file type to bmp or png. Then click Save. Image now had been converted to bitmap. Continue these steps with other 1000 images.

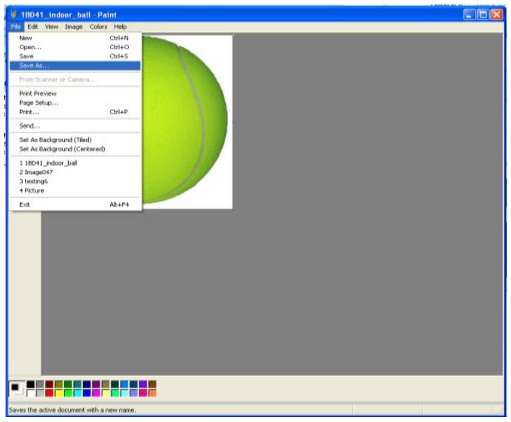
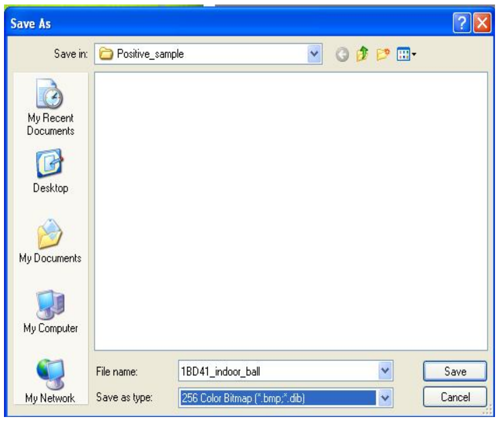

1BD41_indoor_ball - Paint
File Edit View Image Colors Help
New Ctrl+N
Open... Ctrl+O
Save Ctrl+S
Save As...
From Scanner or Camera...
Print Preview
Page Setup...
Print... Ctrl+P
Send...
Set As Background (Tiled)
Set As Background (Centered)
1 1BD41_indoor_ball
2 Image047
3 testing6
4 Picture
Exit Alt+F4
Saves the active document with a new name.

Fig 3.8: Click Save as in menu bar



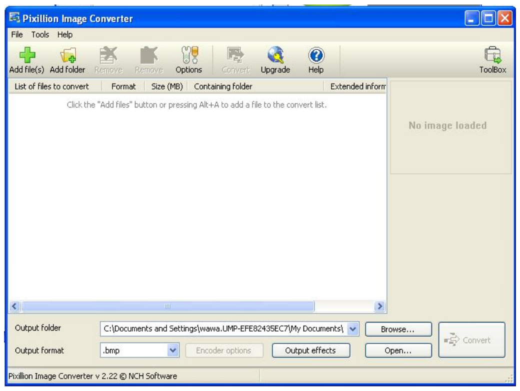
Save in: Positive_sample
File name: 1BD41_indoor_ball
Save as type: 256 Color Bitmap (*.bmp;*.dib)

Fig 3.9: Save file as bitmap @png file format

2) Steps to convert image file format using Pixillion software.

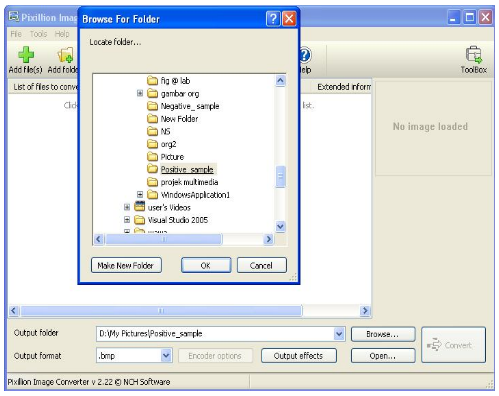
i) Open Pixillion image converter software.

ii) Click “Add Folder” icon in menu taskbar. Browse image folder to convert. Click “OK”.


Pixillion Image Converter
File Tools Help
Add file(s) Add folder Remove Remove Options Convert Upgrade Help ToolBox
List of files to convert Format Size (MB) Containing folder Extended inform
Click the "Add files" button or pressing Alt+A to add a file to the convert list.
No image loaded
Browse...
Encoder options Output effects Open...
Convert
Pixillion Image Converter v 2.22 © NCH Software

Output folder: C:\Documents and Settings\wawa.UMP-EFE82435EC7\My Documents\
Output format: .bmp

Fig 3.10: Paxillion workspace.

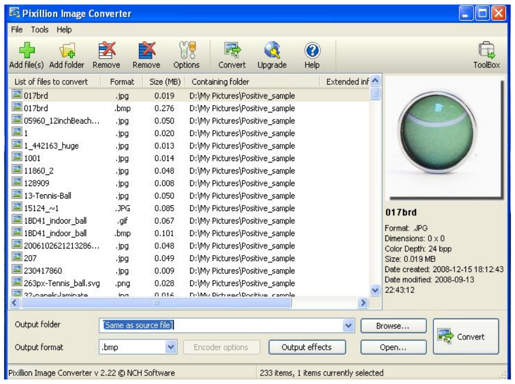


Output folder: D:\My Pictures\Positive_sample
Output format: .bmp

Fig 3.11: Browse image’s folder

iii) Change the “Output Folder” and “Output Format” to desired output. For advanced output effects, click “Output Effects”. Remember to change the file format to bitmap or png.

iv) Preaa “Ctrl+A” to select all files to convert. Click “Convert” button for completing conversion process. Now all files is in bitmap format.


Pixillion Image Converter
File Tools Help
Add file(s) Add folder Remove Remove Options Convert Upgrade Help ToolBox
Browse... Encoder options Output effects Open... Convert

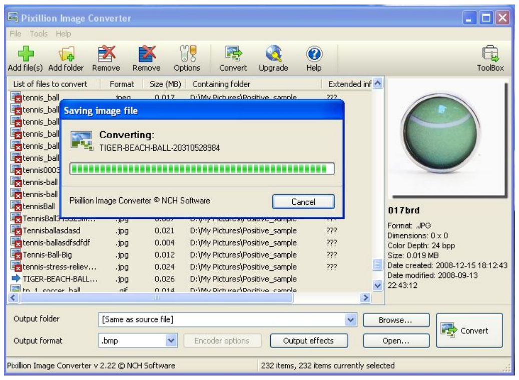
List of files to convert	Format	Size (MB)	Containing folder	Extended inf
017brd	.jpg	0.019	D:\My Pictures\Positive_sample	
017brd	.bmp	0.276	D:\My Pictures\Positive_sample	
05960_12inchBeach...	.jpg	0.050	D:\My Pictures\Positive_sample	
1	.jpg	0.020	D:\My Pictures\Positive_sample	
1_442163_huge	.jpg	0.013	D:\My Pictures\Positive_sample	
1001	.jpg	0.014	D:\My Pictures\Positive_sample	
11860_2	.jpg	0.048	D:\My Pictures\Positive_sample	
128909	.jpg	0.008	D:\My Pictures\Positive_sample	
13-Tennis-Ball	.jpg	0.050	D:\My Pictures\Positive_sample	
15124_~1	.JPG	0.085	D:\My Pictures\Positive_sample	
1BD41_indoor_ball	.gif	0.067	D:\My Pictures\Positive_sample	
1BD41_indoor_ball	.bmp	0.101	D:\My Pictures\Positive_sample	
2006102621213286...	.jpg	0.048	D:\My Pictures\Positive_sample	
207	.jpg	0.049	D:\My Pictures\Positive_sample	
230417860	.jpg	0.009	D:\My Pictures\Positive_sample	
263px-Tennis_ball.svg	.png	0.028	D:\My Pictures\Positive_sample	

017brd
Format: .JPG
Dimensions: 0 x 0
Color Depth: 24 bpp
Size: 0.019 MB
Date created: 2008-12-15 18:12:43
Date modified: 2008-09-13 22:43:12

Output folder: [Same as source file]
Output format: .bmp

Pixillion Image Converter v 2.22 © NCH Software — 233 items, 1 items currently selected

Fig 3.12: Change the “output folder”, “output format”.


Pixillion Image Converter
File Tools Help
Add file(s) Add folder Remove Remove Options Convert Upgrade Help ToolBox
Saving image file
Cancel
Browse... Encoder options Output effects Open... Convert

Converting:
TIGER-BEACH-BALL-20310528984

Pixillion Image Converter © NCH Software

List of files to convert	Format	Size (MB)	Containing folder	Extended inf
Tennisballasdasd	.jpg	0.021	D:\My Pictures\Positive_sample	???
tennis-ballasdfsdfdf	.jpg	0.004	D:\My Pictures\Positive_sample	???
Tennis-Ball-Big	.jpg	0.012	D:\My Pictures\Positive_sample	???
tennis-stress-reliev...	.jpg	0.024	D:\My Pictures\Positive_sample	???
TIGER-BEACH-BALL...	.jpg	0.026	D:\My Pictures\Positive_sample	

017brd
Format: .JPG
Dimensions: 0 x 0
Color Depth: 24 bpp
Size: 0.019 MB
Date created: 2008-12-15 18:12:43
Date modified: 2008-09-13 22:43:12

Output folder: [Same as source file]
Output format: .bmp

Pixillion Image Converter v 2.22 © NCH Software — 232 items, 232 items currently selected

Fig 3.13: Converting file format to bitmap.

3.3.5 Create info files

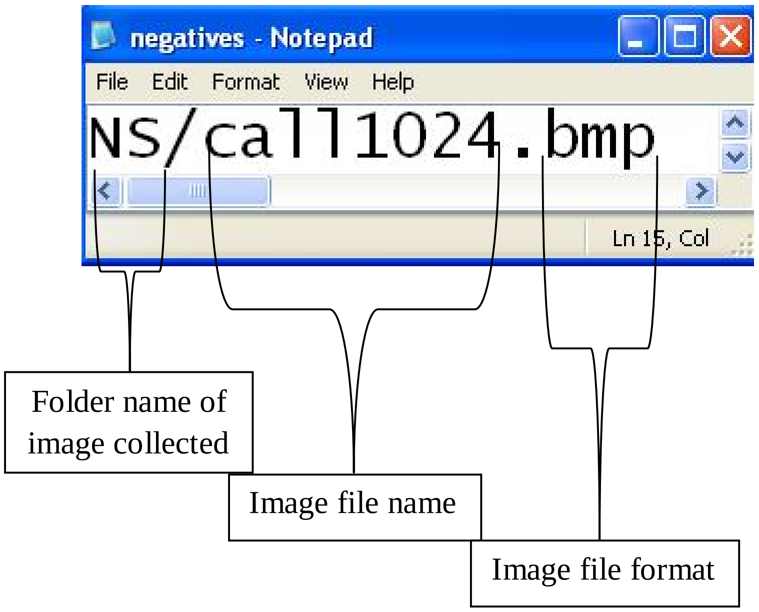
Info files are lists of all image files that have been collected to be trained. This list file must be in “.txt” file which means it is a text file which can be created using “Notepad” or “Word pad”. This method will be useful in createsamples session. These are the process for creating the info file of positive and negative image files:

i) List the entire negatives images file name in “Notepad”..

ii) Rename the file as “negatives” for instance use.

iii) For positives images, the info file will be created under section **3.4.5** which is object marking section. Don’t forget to place both file listing in the “haarcascade” folder.

The format of listing info file is;



```
NS/call1024.bmp
```

Fig 3. 14: Image file listing format

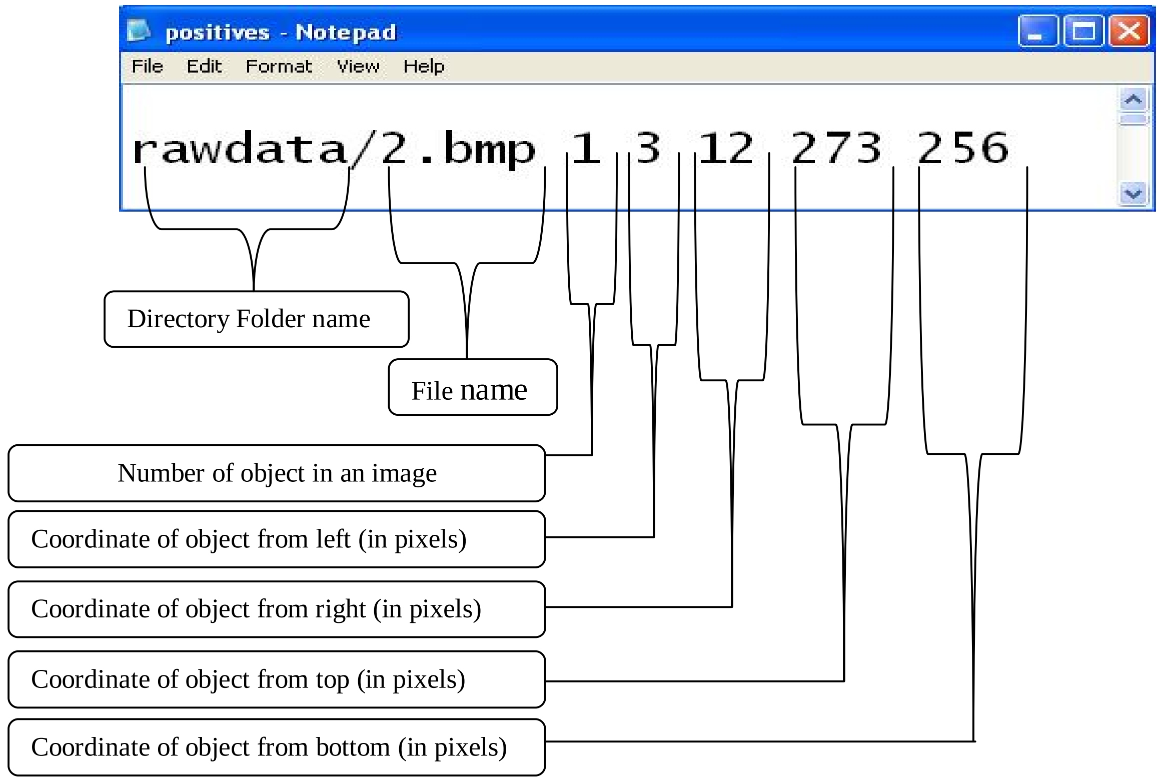
3.3.6 Object Mark (crop image)

Object marking method is important to know the location of object in an image whereas the location of object could be either at the bottom, top, center or in other –x and –y grid. So for the purpose of determining the coordinate of the object, the method of object marking should be done. This method is important to crop or extract the desired object to differentiate it from other object.

There are two solutions that can be used as the way to accomplish these steps. First is by crop it manually by using basic program such as Microsoft Office Picture Manager. Second way is by using object marker.exe which can be downloaded at OpenCV. Using object marker.exe should be much easier than by crop it manually.

The result by doing this method should be in the format of below condition and must be in .txt/.dat file.

The format of listing object marking info file is;


positives - Notepad
File Edit Format View Help
Directory Folder name
File name
Number of object in an image
Coordinate of object from left (in pixels)
Coordinate of object from right (in pixels)
Coordinate of object from top (in pixels)
Coordinate of object from bottom (in pixels)

```
rawdata/2.bmp 1 3 12 273 256
```

Fig 3.15 : Format of marked object list file

1) Step to mark object manually

This step should be more difficult because the entire file must be marked one by one but it was useful if “object marker.exe” couldn’t be find. These are the steps of converting file using “Paint”

Step 1:

i) Open “Microsoft Office Picture Manager”.

ii) Click “Picture" on the Menu bar and Click “Crop”.

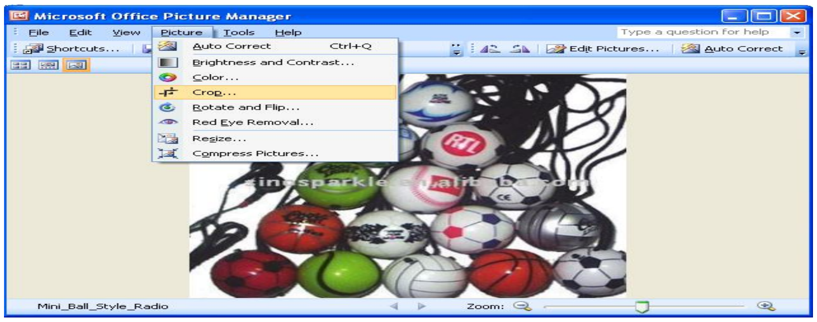
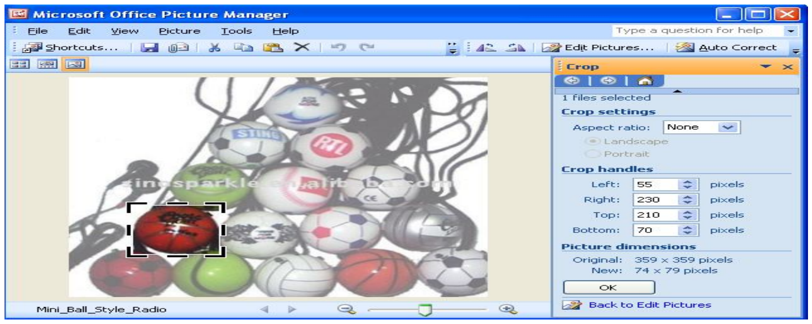


Fig 3.16: Microsoft Office Picture Manager workspace.

Step 2:

iii) Change the number of pixels under the crop handles to select desired object.

iv) Change the pixels on right, left, top and bottom part to select obje



1 files selected

Crop settings

Aspect ratio: None

Landscape

Portrait

Crop handles

Left: 55 pixels

Right: 230 pixels

Top: 210 pixels

Bottom: 70 pixels

Picture dimensions

Original: 359 x 359 pixels

New: 74 x 79 pixels

OK

Back to Edit Pictures

Fig 3.17: Crop Image By Changing Pixels in Crop Handles.

Step 3:

i) Continue the step for other 1000 of positive images collected.

ii) Create new file of text file and place the readings from crop handles of 1000 image to the text file that had been created.

iii) List the reading of marked object as the format show in figure

2) Step to mark object using object marker.exe.

This method should be easier and faster. The info file could be created automatically as long as the object of interest had been cropped from an image**.**

Step 1:

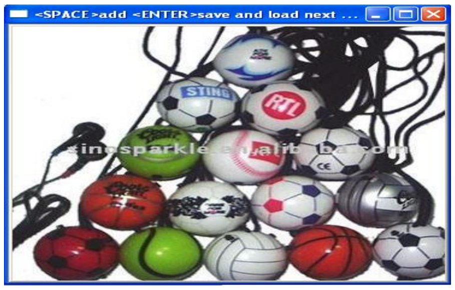
i) Download haarkit tools at ………..

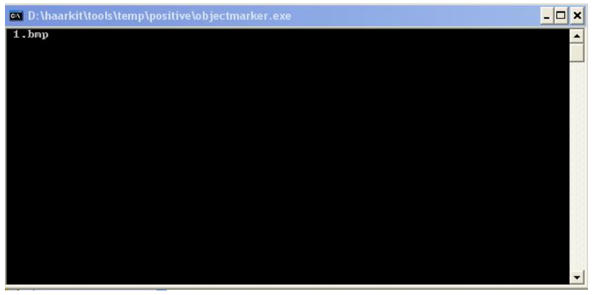
ii) Save all positives image in “rawdata” folder. “rawdata” folder can be found under directory haarkit\tools\temp\positive\rawdata. Folder “rawdata” can be renamed to any other desirable name.

iii) Remember all positive image files to be mark must be in bitmap format.

Step 2:

i) Clicks object marker.exe. Figure below will appear.



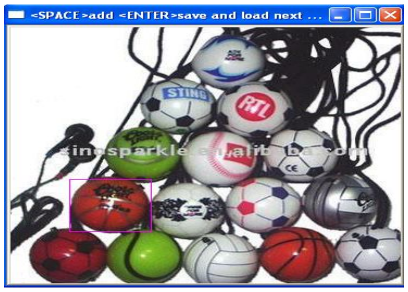


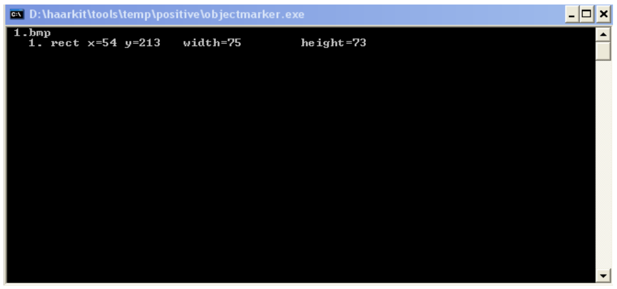
```
1.bmp
```

Fig 3.18: Windows appear after open object marker.exe.

Step 3:

i) Draw rectangular boundary to crop the desired object. If done, press space bar to enter the marking reading in command window and press enter to work with other image.





```
1.bmp
  1. rect x=54 y=213   width=75        height=73
```

Fig 3.19: Windows appear after drawing rectangular boundary on object.

Step 4:

i) All the result of marking will be saved in the file named info.txt which can be finding in the path" haarkit\tools\temp\positive".

ii) Rename the file info.txt to any desirable name. In this project, the file name has been renamed as "positives.txt" and places it in the "haarcascade" folder (worked folder).

3.3.7 Sample creating method using createsample.exe.

OpenCV has built in training system to construct a classifier, for training purpose the system works quite well. OpenCV generates the samples images that will be used for training purpose. The program used for creating samples using OpenCV is createsamples.exe and the command used is "–createsamples".

This is the step correspond to the process of creating the samples.

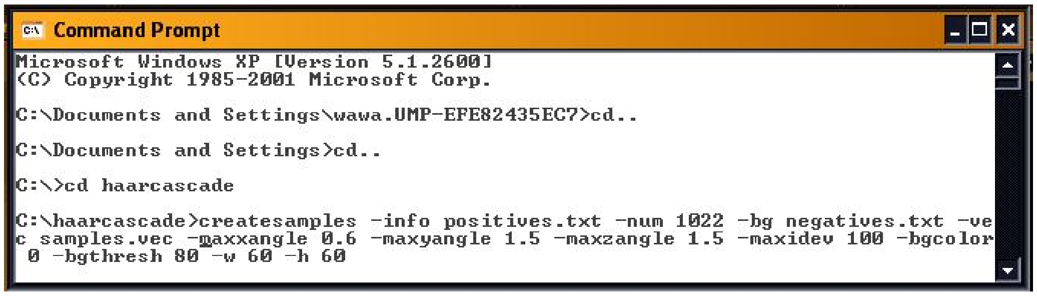
Step :

i) Open Command Prompt and type the following coding.

```
createsamples –info positives.txt -num 1022 -bg negatives.txt -vec
samples.vec -    maxxangle 0.6 -maxyangle 1.5 -maxzangle 1.5 -maxidev 100 -
bgcolor 0 -bgthresh 80 -w 50 -h 50
```

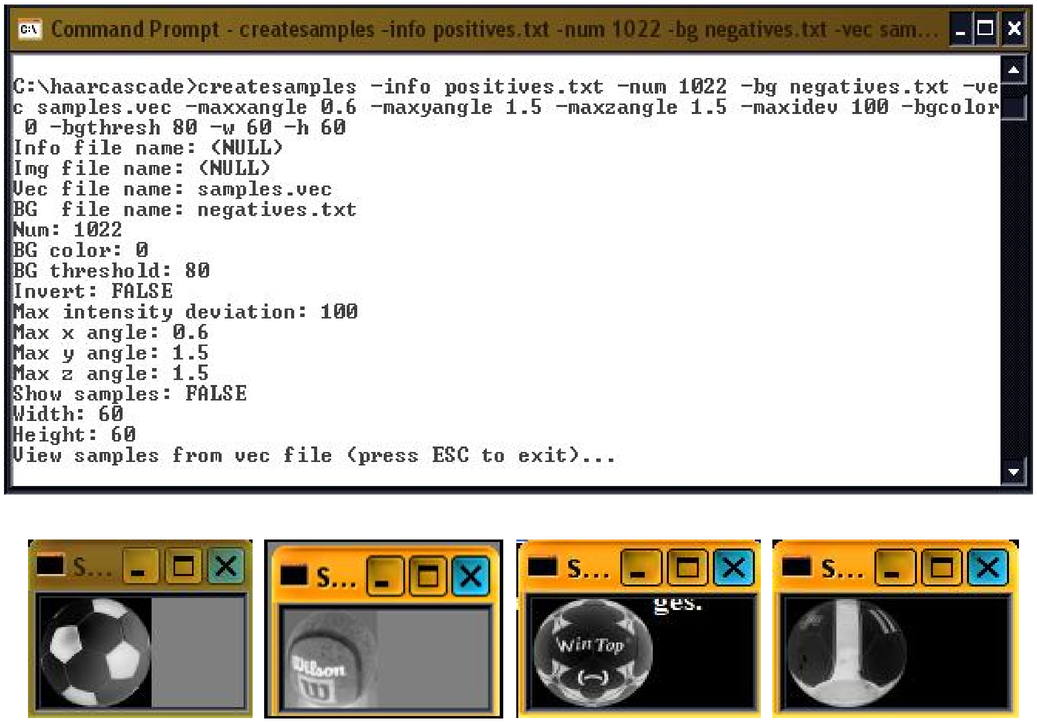
☆ All the values can be changed according to limitation and range.

```
Usage: ./createsamples
[-info <description_file_name @ info file of marked positive object>]
[-img <image_file_name>]
[-vec <vec_file_name>]
[-bg <background_file_name>]
[-num <number_of_samples = 1000>]
[-bgcolor <background_color = 0>]
[-inv] [-randinv] [-bgthresh <background_color_threshold = 80>]
[-maxidev <max_intensity_deviation = 40>]
[-maxxangle <max_x_rotation_angle = 1.100000>]
[-maxyangle <max_y_rotation_angle = 1.100000>]
[-maxzangle <max_z_rotation_angle = 0.500000>]
[-show [<scale = 4.000000>]]
[-w <sample_width = 24>]
[-h <sample_height = 24>]
```



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\wawa.UMP-EFE82435EC7>cd..

C:\Documents and Settings>cd..

C:\>cd haarcascade

C:\haarcascade>createsamples -info positives.txt -num 1022 -bg negatives.txt -ve
c samples.vec -maxxangle 0.6 -maxyangle 1.5 -maxzangle 1.5 -maxidev 100 -bgcolor
 0 -bgthresh 80 -w 60 -h 60
```

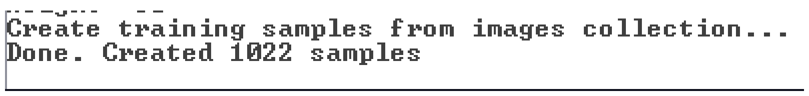
Fig 3.20: Implementing the coding on command prompt.



```
C:\haarcascade>createsamples -info positives.txt -num 1022 -bg negatives.txt -ve
c samples.vec -maxxangle 0.6 -maxyangle 1.5 -maxzangle 1.5 -maxidev 100 -bgcolor
 0 -bgthresh 80 -w 60 -h 60
Info file name: (NULL)
Img file name: (NULL)
Vec file name: samples.vec
BG  file name: negatives.txt
Num: 1022
BG color: 0
BG threshold: 80
Invert: FALSE
Max intensity deviation: 100
Max x angle: 0.6
Max y angle: 1.5
Max z angle: 1.5
Show samples: FALSE
Width: 60
Height: 60
View samples from vec file (press ESC to exit)...
```

Fig 3.21: Creating the samples of collecting images.

ii) Creating sample method is done after the command prompt display below result:



```
Create training samples from images collection...
Done. Created 1022 samples
```

Fig 3.22: Creating samples done.

3.3.8 Create training sample using haartraining.exe.

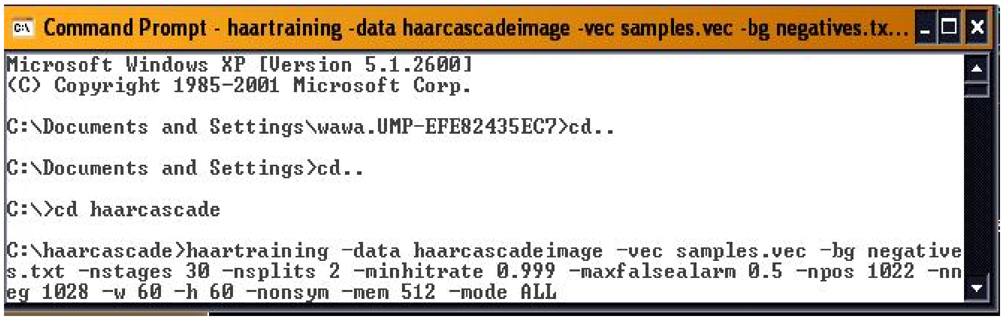
Training samples using OpenCV is based on AdaBoost approach. The main goals of the AdaBoost learning algorithm are to select a few set of features which represents features as well as possible round image and train the strong classifier which the linear combination of these best features. Eventually, training process should take at least three days and can also take to a week to complete training process. It's all depends on the total samples used and the memory usage for training process. As for recommendation, memory usage should not totally used. Just use half from the memory storage of your hard drive. Total memory usage could harm the computer's operational system. The following steps can describe briefly about haartraining process.

Step 1:

i) Open Command Prompt and type this coding.

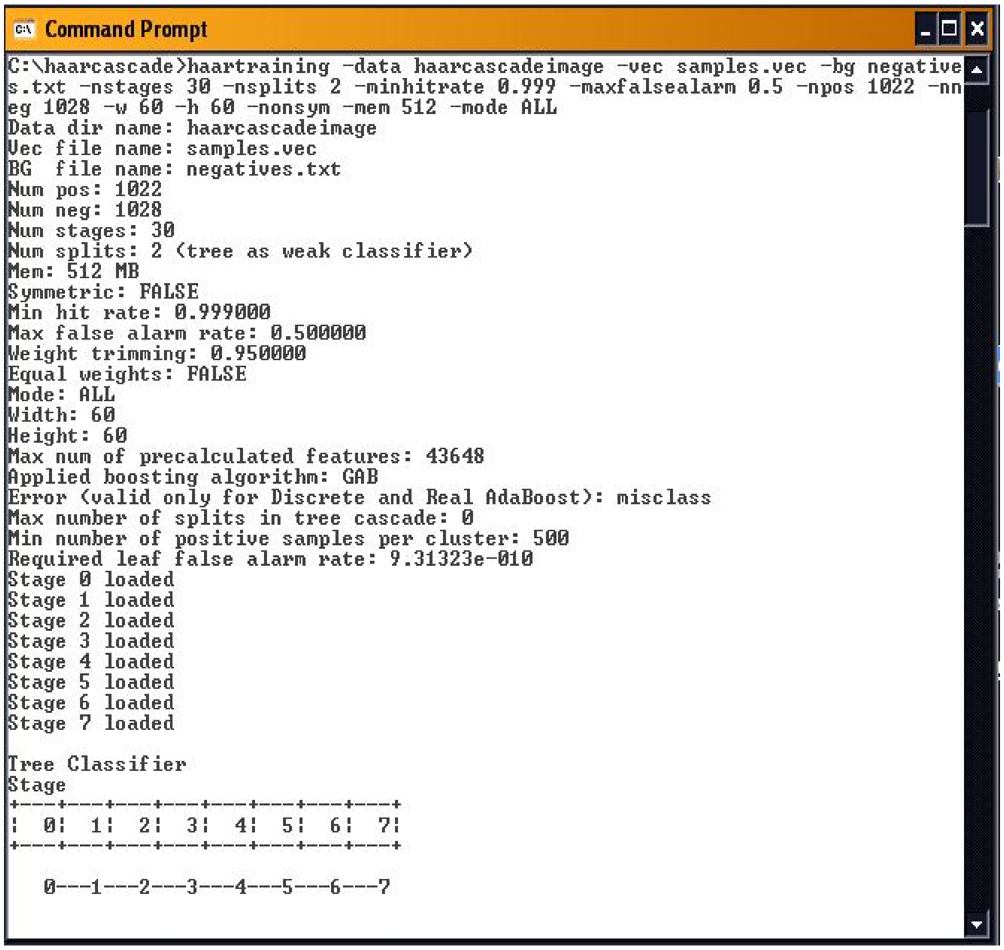
```
haartraining -data haarcascadeimage -vec samples.vec -bg negatives.txt -nstages 30 -
nsplits 2 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 1022 -nneg 1028 -w 60 -h 60 -
nonsym -mem 512 -mode ALL
```

```
Usage: ./haartraining
-data <dir_name>
-vec <vec_file_name>
-bg <background_file_name>
[-npos <number_of_positive_samples = 2000>]
[-nneg <number_of_negative_samples = 2000>]
[-nstages <number_of_stages = 14>]
[-nsplits <number_of_splits = 1>]
[-mem <memory_in_MB = 200>]
[-sym (default)] [-nonsym]
[-minhitrate <min_hit_rate = 0.995000>]
[-maxfalsealarm <max_false_alarm_rate = 0.500000>]
[-weighttrimming <weight_trimming = 0.950000>]
[-eqw]
[-mode <BASIC (default) | CORE | ALL>]
[-w <sample_width = 24>]
[-h <sample_height = 24>]
[-bt <DAB | RAB | LB | GAB (default)>]
[-err <misclass (default) | gini | entropy>]
```


Command Prompt - haartraining -data haarcascadeimage -vec samples.vec -bg negatives.tx...

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\wawa.UMP-EFE82435EC7>cd..

C:\Documents and Settings>cd..

C:\>cd haarcascade

C:\haarcascade>haartraining -data haarcascadeimage -vec samples.vec -bg negative
s.txt -nstages 30 -nsplits 2 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 1022 -nn
eg 1028 -w 60 -h 60 -nonsym -mem 512 -mode ALL
```

Fig 3.23: Implementing the coding on command prompt



```
C:\haarcascade>haartraining -data haarcascadeimage -vec samples.vec -bg negative
s.txt -nstages 30 -nsplits 2 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 1022 -nn
eg 1028 -w 60 -h 60 -nonsym -mem 512 -mode ALL
Data dir name: haarcascadeimage
Vec file name: samples.vec
BG  file name: negatives.txt
Num pos: 1022
Num neg: 1028
Num stages: 30
Num splits: 2 (tree as weak classifier)
Mem: 512 MB
Symmetric: FALSE
Min hit rate: 0.999000
Max false alarm rate: 0.500000
Weight trimming: 0.950000
Equal weights: FALSE
Mode: ALL
Width: 60
Height: 60
Max num of precalculated features: 43648
Applied boosting algorithm: GAB
Error (valid only for Discrete and Real AdaBoost): misclass
Max number of splits in tree cascade: 0
Min number of positive samples per cluster: 500
Required leaf false alarm rate: 9.31323e-010
Stage 0 loaded
Stage 1 loaded
Stage 2 loaded
Stage 3 loaded
Stage 4 loaded
Stage 5 loaded
Stage 6 loaded
Stage 7 loaded

Tree Classifier
Stage
+---+---+---+---+---+---+---+---+
|  0|  1|  2|  3|  4|  5|  6|  7|
+---+---+---+---+---+---+---+---+

   0---1---2---3---4---5---6---7
```

Fig 3.24: Training all samples process of 7 stages

3.3.9 Convert the datasets to xml file

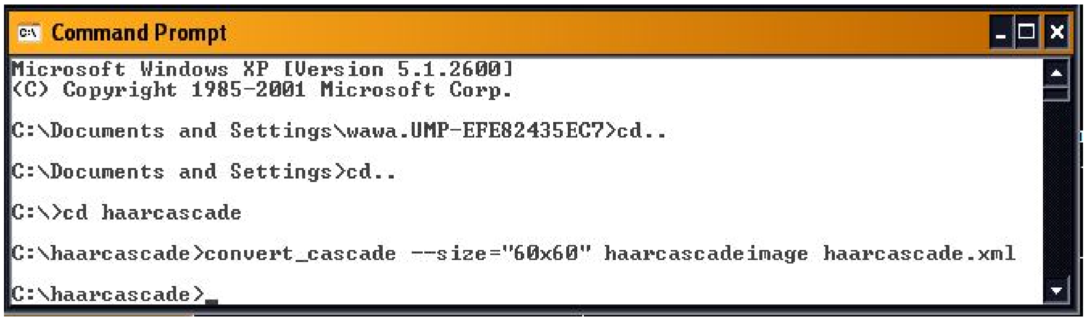
For detection this cascade of classifier should be converted in xml format. The haartraing generates a xml file when the process is completely finished.

The input format as following coding is implemented in command prompt.

```
{ convert_cascade --size="<sample_width>x<sampe_height>"
  <haartraining_ouput_dir> <ouput_file> }
```

Example:

```
convert_cascade --size="20x20" haarcascadeimage haarcascade.xml
```



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\wawa.UMP-EFE82435EC7>cd..

C:\Documents and Settings>cd..

C:\>cd haarcascade

C:\haarcascade>convert_cascade --size="60x60" haarcascadeimage haarcascade.xml

C:\haarcascade>_
```

Fig 3.25: Implementing the coding on command prompt.

Note that the result of converting datasets to xml file is the result of all process of training phase. The xml file created is a new classifier of desired object in this project which is round image. To detect round object in any image, just implement this .xml file in object detection coding represent in execution phase.

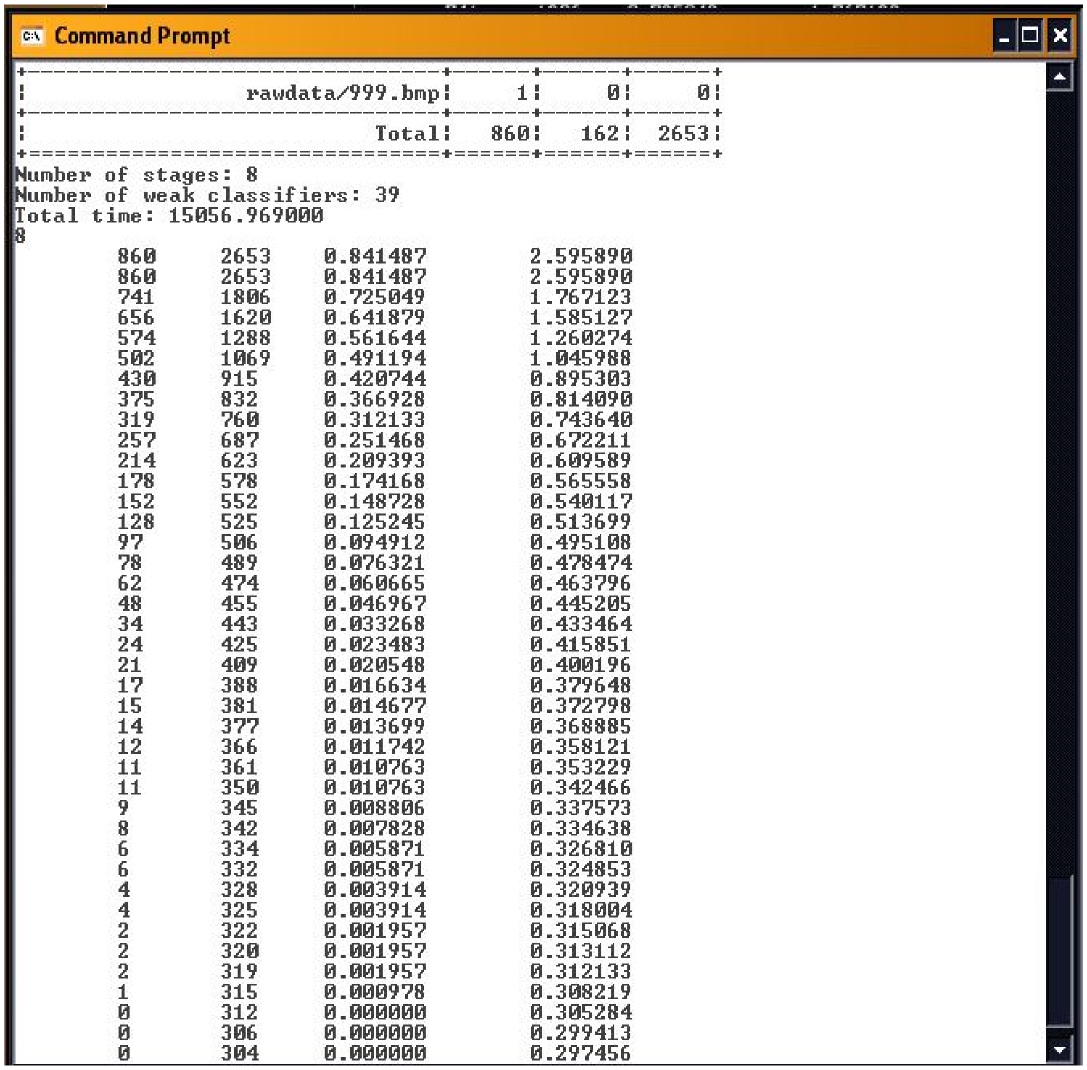
3.3.10 Performance Test

Before the training phase is done completely, the performance test is needed to examine the ability of classifier created on detecting desired object. This process is important because if the percentage of performance is low, the training phase should be repeating again with increment in total image collected for training. If high, the classifier performance is good and effective, so detection process will going accurately.

The performance of the classifier can be tested using this method.

```
performance -data haarcascade -w 20 -h 20 -info tests.dat -ni
                              or
performance -data haarcascade.xml -info tests.dat –ni
```

```
Usage: ./performance
-data <classifier_directory_name>
-info <collection_file_name>
[-maxSizeDiff <max_size_difference = 1.500000>]
[-maxPosDiff <max_position_difference = 0.300000>]
[-sf <scale_factor = 1.200000>]
[-ni]
[-nos <number_of_stages = -1>]
[-rs <roc_size = 40>]
[-w <sample_width = 24>]
[-h <sample_height = 24>]
```



```
+--------------------------------+-------+-------+-------+
|                 rawdata/999.bmp|      1|      0|      0|
+--------------------------------+-------+-------+-------+
|                           Total|    860|    162|   2653|
+================================+=======+=======+=======+
Number of stages: 8
Number of weak classifiers: 39
Total time: 15056.969000
8
        860     2653    0.841487        2.595890
        860     2653    0.841487        2.595890
        741     1806    0.725049        1.767123
        656     1620    0.641879        1.585127
        574     1288    0.561644        1.260274
        502     1069    0.491194        1.045988
        430     915     0.420744        0.895303
        375     832     0.366928        0.814090
        319     760     0.312133        0.743640
        257     687     0.251468        0.672211
        214     623     0.209393        0.609589
        178     578     0.174168        0.565558
        152     552     0.148728        0.540117
        128     525     0.125245        0.513699
        97      506     0.094912        0.495108
        78      489     0.076321        0.478474
        62      474     0.060665        0.463796
        48      455     0.046967        0.445205
        34      443     0.033268        0.433464
        24      425     0.023483        0.415851
        21      409     0.020548        0.400196
        17      388     0.016634        0.379648
        15      381     0.014677        0.372798
        14      377     0.013699        0.368885
        12      366     0.011742        0.358121
        11      361     0.010763        0.353229
        11      350     0.010763        0.342466
        9       345     0.008806        0.337573
        8       342     0.007828        0.334638
        6       334     0.005871        0.326810
        6       332     0.005871        0.324853
        4       328     0.003914        0.320939
        4       325     0.003914        0.318004
        2       322     0.001957        0.315068
        2       320     0.001957        0.313112
        2       319     0.001957        0.312133
        1       315     0.000978        0.308219
        0       312     0.000000        0.305284
        0       306     0.000000        0.299413
        0       304     0.000000        0.297456
```

Fig 3.26: Performance test process.

3.4 Execution phase.

Execution phase is a phase to test the classifier created during training phase. Simple coding using OpenCV library and Visual Studio software including the user interface should be created to test the classifier. There are also some few steps in order to completing the coding development process.

3.4.1 Create user interface.

Use MFC application to generate a form that will become a user interface that will connect the user with the system, the example of window form that can be created is as below:

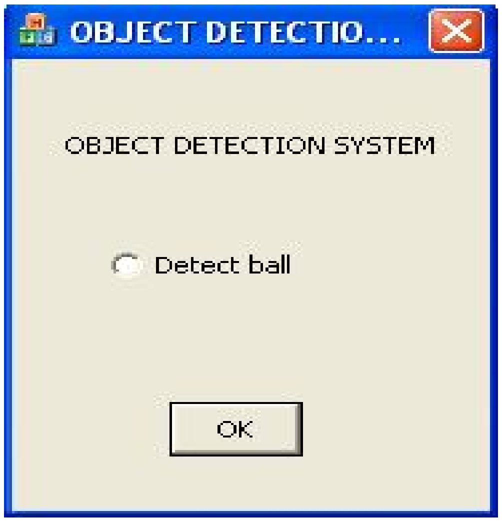
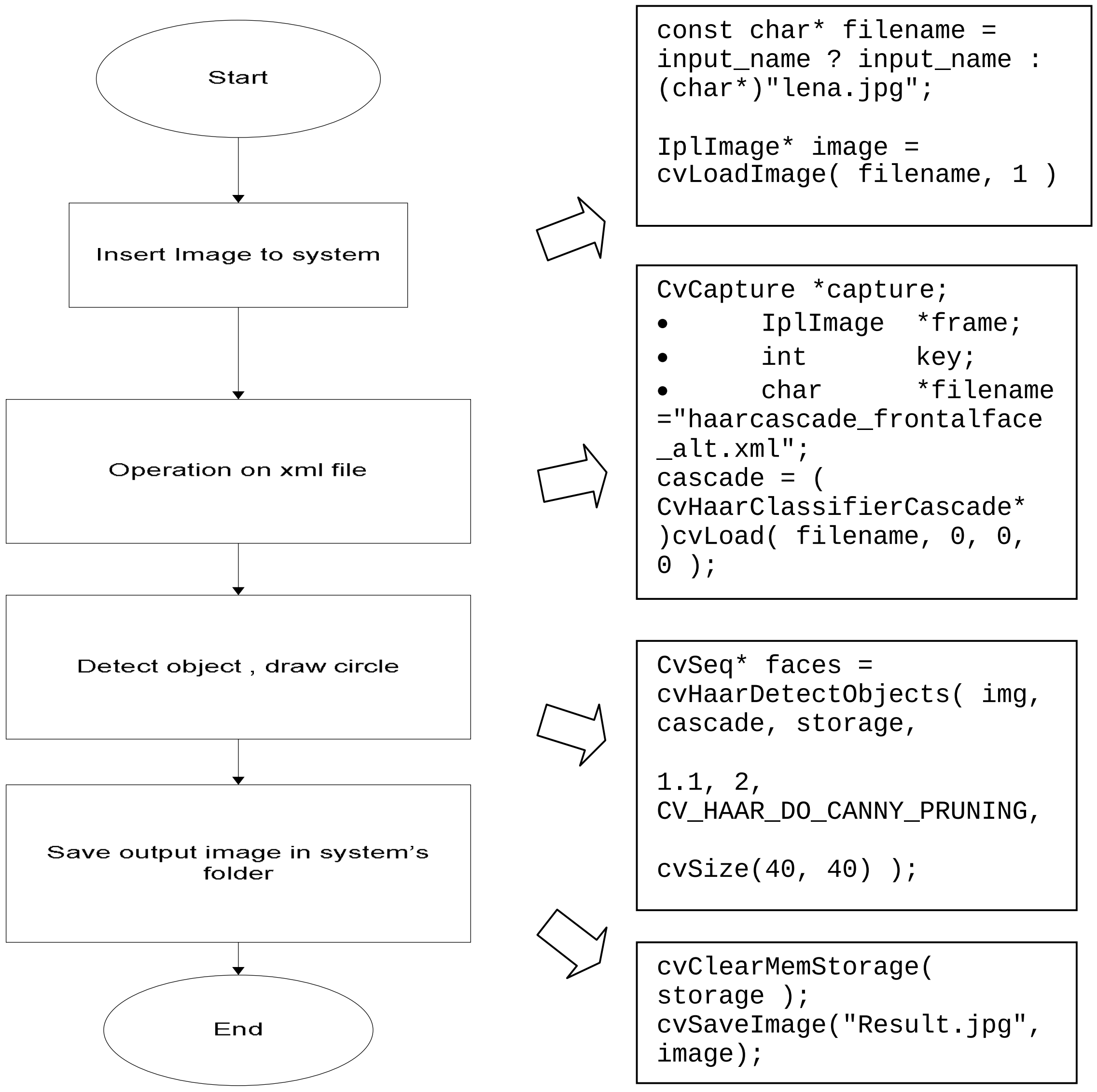


Fig 3.27 : User Interface

3.4.2 Coding development

Coding is developed step by step according to flowchart. Flowchart of the execution phase is as follows:


Start
Insert Image to system
Operation on xml file
Detect object , draw circle
Save output image in system's folder
End

```
const char* filename =
input_name ? input_name :
(char*)"lena.jpg";

IplImage* image =
cvLoadImage( filename, 1 )
```

```
CvCapture *capture;
•      IplImage  *frame;
•      int       key;
•      char      *filename
="haarcascade_frontalface
_alt.xml";
cascade = (
CvHaarClassifierCascade*
)cvLoad( filename, 0, 0,
0 );
```

```
CvSeq* faces =
cvHaarDetectObjects( img,
cascade, storage,

1.1, 2,
CV_HAAR_DO_CANNY_PRUNING,

cvSize(40, 40) );
```

```
cvClearMemStorage(
storage );
cvSaveImage("Result.jpg",
image);
```

Fig 3.28: Coding Flowchart

3.4.3 Coding implementation

System overview figure that detection process start after a button of detect ball is clicked. Therefore, adding some coding for detection to the button of “detect ball” is necessary. The steps of applying the coding to the button can be referred to the OpenCV using MFC website.

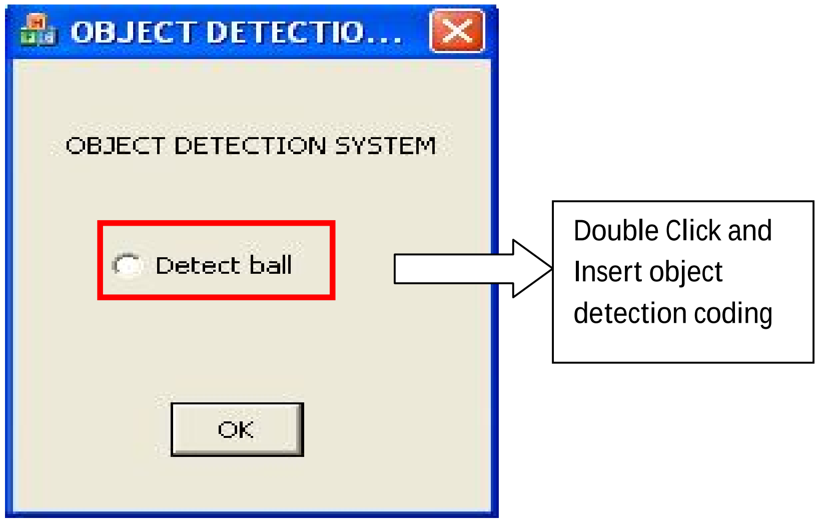


Figure 3.29: Add coding to the button

After implementing the coding to the button, the system can be executed and the result of the detection process can be observed.

CHAPTER 4

RESULTS AND DISCUSSIONS

The methodology had come out with some result in order to complete the object detection systems development. These are the results of step by step methods done in the methodology development during the training and execution phase within its problems.

i) Training phase

Training phase will produce a classifier of object of interest which is in the format of xml file. For the purpose of reproduce the xml file, all images must be converted to bitmap format, the list of positive and negative image and the vector file must be created. The most important part is the weak constraint of object must be calculated before file is converted to the xml file.

ii) Execution phase

Execution phase is the process to test the ability of the classifier. In the execution phase, the coding for object detection is developed using Visual Studio 2005 and OpenCV library.

4.1 Training phase results

In training phase, there are eight methods taking part and the result of all methods can be explained as below results. The problem of some steps during the training phase is described.

4.1.1 Result of folder creation

As all the work must be in one folder during the training phase, the OpenCV utilities that is needed during the training process must also placed in the folder. Figure below will show the "haarcascade" folder which contains all the OpenCV utilities for training process.

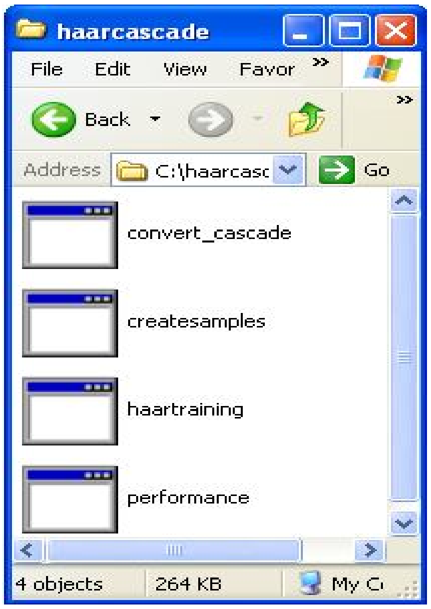


Fig 4.1: Initial "haarcascade" folder's contents.

If these utilities are not been placed in this folder, the OpenCV process could not being completed and some error of unrecognized command will appear. Placing all this utilities in the folder will solve this problem.

4.1.2 Result of Image Collecting

From the figure below, total positive image collected is 1022 and negative image collected is 1028. Most of positive image that had been collected is a single image which contains only the object of interest and does not contain any other image. This is for the purpose to make the object marking method easier. Note that the positive image folder which contains all positive images is named “rawdata” is placed in the folder “haarcascade”. Negative images folder is named as “NS” and also has been located in the folder “haarcascade”.

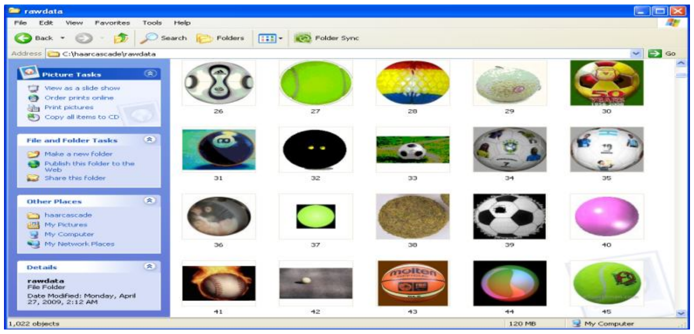


Fig 4.2: 1022 of Positive Image Collection.

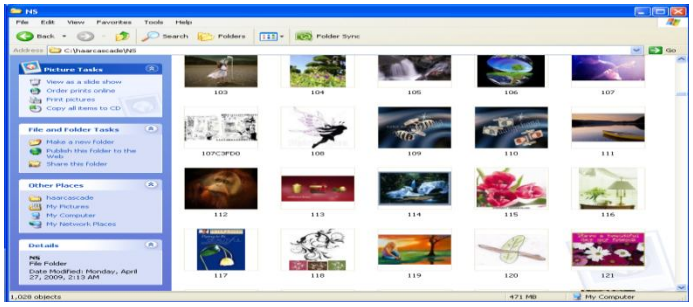
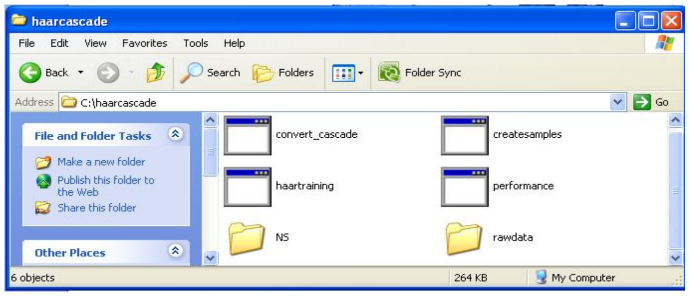


Fig 4.3 : 1028 of Negative Image Collection


haarcascade

Address C:\haarcascade

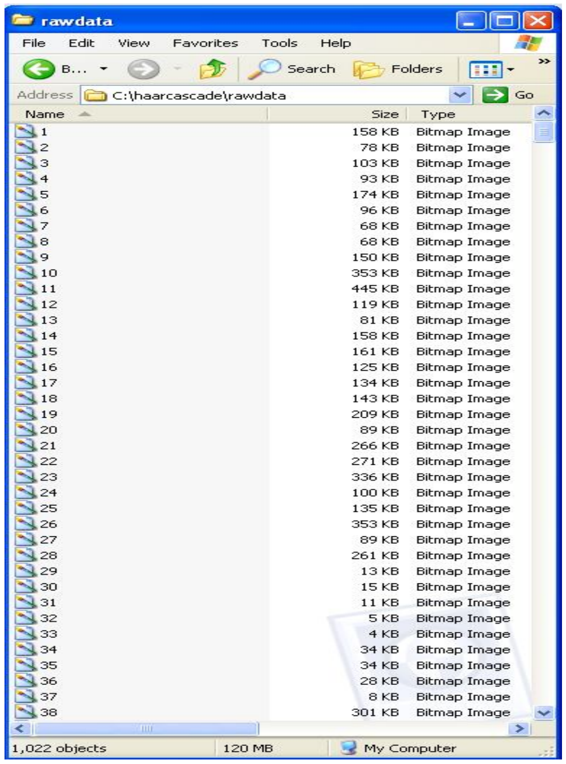
convert_cascade
createsamples
haartraining
performance
NS
rawdata

6 objects | 264 KB | My Computer

Fig 4.4: Positive image folder “rawdata” and Negative Image Folder “NS” is placed in the folder “haarcascade”

4.1.3 Result on Image Conversion

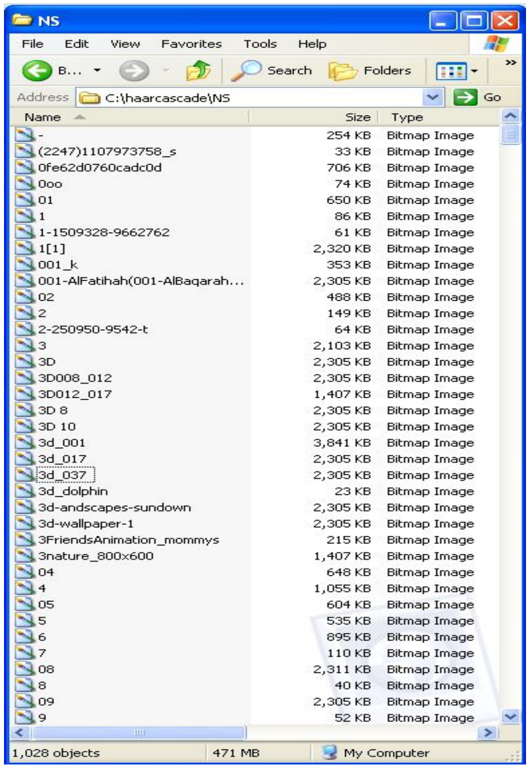
After the image converting process, all image file format now is in the format of bitmap.


rawdata

Address C:\haarcascade\rawdata

Name	Size	Type
1	158 KB	Bitmap Image
2	78 KB	Bitmap Image
3	103 KB	Bitmap Image
4	93 KB	Bitmap Image
5	174 KB	Bitmap Image
6	96 KB	Bitmap Image
7	68 KB	Bitmap Image
8	68 KB	Bitmap Image
9	150 KB	Bitmap Image
10	353 KB	Bitmap Image
11	445 KB	Bitmap Image
12	119 KB	Bitmap Image
13	81 KB	Bitmap Image
14	158 KB	Bitmap Image
15	161 KB	Bitmap Image
16	125 KB	Bitmap Image
17	134 KB	Bitmap Image
18	143 KB	Bitmap Image
19	209 KB	Bitmap Image
20	89 KB	Bitmap Image
21	266 KB	Bitmap Image
22	271 KB	Bitmap Image
23	336 KB	Bitmap Image
24	100 KB	Bitmap Image
25	135 KB	Bitmap Image
26	353 KB	Bitmap Image
27	89 KB	Bitmap Image
28	261 KB	Bitmap Image
29	13 KB	Bitmap Image
30	15 KB	Bitmap Image
31	11 KB	Bitmap Image
32	5 KB	Bitmap Image
33	4 KB	Bitmap Image
34	34 KB	Bitmap Image
35	34 KB	Bitmap Image
36	28 KB	Bitmap Image
37	8 KB	Bitmap Image
38	301 KB	Bitmap Image

1,022 objects | 120 MB | My Computer


NS

Address C:\haarcascade\NS

Name	Size	Type
-	254 KB	Bitmap Image
(2247)1107973758_s	33 KB	Bitmap Image
0fe62d0760cadc0d	706 KB	Bitmap Image
0oo	74 KB	Bitmap Image
01	650 KB	Bitmap Image
1	86 KB	Bitmap Image
1-1509328-9662762	61 KB	Bitmap Image
1[1]	2,320 KB	Bitmap Image
001_k	353 KB	Bitmap Image
001-AlFatihah(001-AlBaqarah...	2,305 KB	Bitmap Image
02	488 KB	Bitmap Image
2	149 KB	Bitmap Image
2-250950-9542-t	64 KB	Bitmap Image
3	2,103 KB	Bitmap Image
3D	2,305 KB	Bitmap Image
3D008_012	2,305 KB	Bitmap Image
3D012_017	1,407 KB	Bitmap Image
3D 8	2,305 KB	Bitmap Image
3D 10	2,305 KB	Bitmap Image
3d_001	3,841 KB	Bitmap Image
3d_017	2,305 KB	Bitmap Image
3d_037	2,305 KB	Bitmap Image
3d_dolphin	23 KB	Bitmap Image
3d-andscapes-sundown	2,305 KB	Bitmap Image
3d-wallpaper-1	2,305 KB	Bitmap Image
3FriendsAnimation_mommys	215 KB	Bitmap Image
3nature_800x600	1,407 KB	Bitmap Image
04	648 KB	Bitmap Image
4	1,055 KB	Bitmap Image
05	604 KB	Bitmap Image
5	535 KB	Bitmap Image
6	895 KB	Bitmap Image
7	110 KB	Bitmap Image
08	2,311 KB	Bitmap Image
8	40 KB	Bitmap Image
09	2,305 KB	Bitmap Image
9	52 KB	Bitmap Image

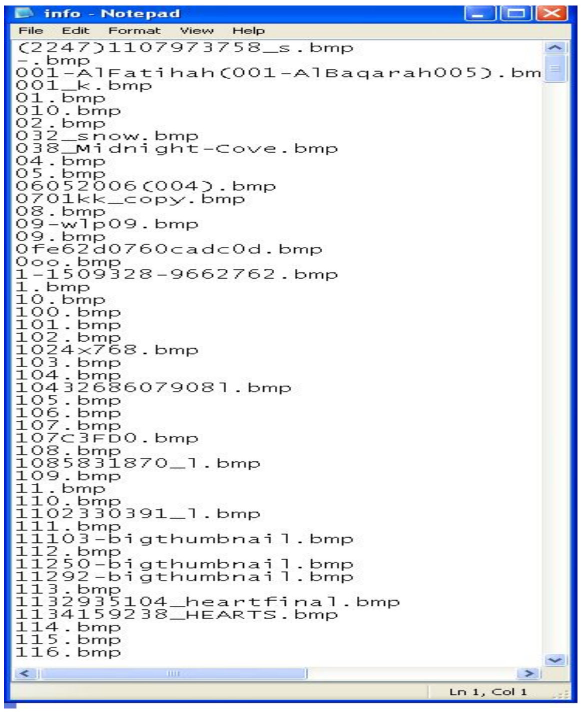
1,028 objects | 471 MB | My Computer

Fig 4.5: Positive Images in bmp format

Fig 4.6: Negative Images in bmp format.

4.1.4 Result on creating info files

Info file is a list of negative image's name and its directory. Info file also must be placed in the "haarcascade" folder to continue with next stages. The list file name of the collections file of negatives image is described as follows:



```
(2247)1107973758_s.bmp
-.bmp
001-AlFatihah(001-AlBaqarah005).bm
001_k.bmp
01.bmp
010.bmp
02.bmp
032_snow.bmp
038_Midnight-Cove.bmp
04.bmp
05.bmp
06052006(004).bmp
0701kk_copy.bmp
08.bmp
09-wlp09.bmp
09.bmp
0fe62d0760cadc0d.bmp
0oo.bmp
1-1509328-9662762.bmp
1.bmp
10.bmp
100.bmp
101.bmp
102.bmp
1024x768.bmp
103.bmp
104.bmp
10432686079081.bmp
105.bmp
106.bmp
107.bmp
107C3FD0.bmp
108.bmp
1085831870_1.bmp
109.bmp
11.bmp
110.bmp
1102330391_1.bmp
111.bmp
11103-bigthumbnail.bmp
112.bmp
11250-bigthumbnail.bmp
11292-bigthumbnail.bmp
113.bmp
1132935104_heartfinal.bmp
1134159238_HEARTS.bmp
114.bmp
115.bmp
116.bmp
```

Fig 4.7: Info file of negative image's collection.

The contents of the "haarcascade" folder are now increase and can be seen as following figure:

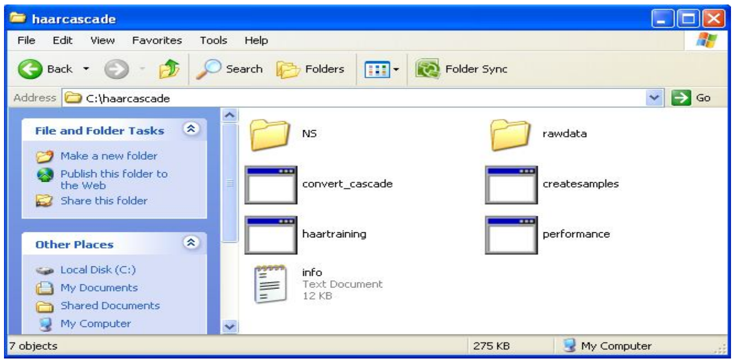
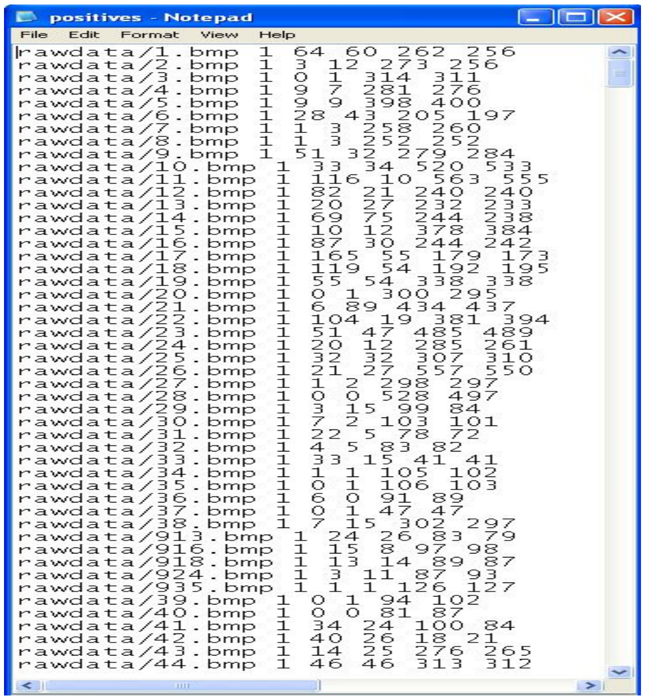


Figure 4.8: Contents of “haarcascade” folder after placing the info file text file.

4.1.5 Result of marking object

Purpose of object marking process is to crop object of interest from an image to create a list of data file which contains the cropped object’s coordinates. This method is done only to the positives images. Figure below will explain the output produce from object marking session.



```
rawdata/1.bmp 1 64 60 262 256
rawdata/2.bmp 1 3 12 273 256
rawdata/3.bmp 1 0 1 314 311
rawdata/4.bmp 1 9 7 281 276
rawdata/5.bmp 1 9 9 398 400
rawdata/6.bmp 1 28 43 205 197
rawdata/7.bmp 1 1 3 258 260
rawdata/8.bmp 1 1 3 252 252
rawdata/9.bmp 1 51 32 279 284
rawdata/10.bmp 1 33 34 520 533
rawdata/11.bmp 1 116 10 563 555
rawdata/12.bmp 1 82 21 240 240
rawdata/13.bmp 1 20 27 232 233
rawdata/14.bmp 1 69 75 244 238
rawdata/15.bmp 1 10 12 378 384
rawdata/16.bmp 1 87 30 244 242
rawdata/17.bmp 1 165 55 179 173
rawdata/18.bmp 1 119 54 192 195
rawdata/19.bmp 1 55 54 338 338
rawdata/20.bmp 1 0 1 300 295
rawdata/21.bmp 1 6 89 434 437
rawdata/22.bmp 1 104 19 381 394
rawdata/23.bmp 1 51 47 485 489
rawdata/24.bmp 1 20 12 285 261
rawdata/25.bmp 1 32 32 307 310
rawdata/26.bmp 1 21 27 557 550
rawdata/27.bmp 1 1 2 298 297
rawdata/28.bmp 1 0 0 528 497
rawdata/29.bmp 1 3 15 99 84
rawdata/30.bmp 1 7 2 103 101
rawdata/31.bmp 1 22 5 78 72
rawdata/32.bmp 1 4 5 83 82
rawdata/33.bmp 1 33 15 41 41
rawdata/34.bmp 1 1 1 105 102
rawdata/35.bmp 1 0 1 106 103
rawdata/36.bmp 1 6 0 91 89
rawdata/37.bmp 1 0 1 47 47
rawdata/38.bmp 1 7 15 302 297
rawdata/913.bmp 1 24 26 83 79
rawdata/916.bmp 1 15 8 97 98
rawdata/918.bmp 1 13 14 89 87
rawdata/924.bmp 1 3 11 87 93
rawdata/935.bmp 1 1 1 126 127
rawdata/39.bmp 1 0 1 94 102
rawdata/40.bmp 1 0 0 81 87
rawdata/41.bmp 1 34 24 100 84
rawdata/42.bmp 1 40 26 18 21
rawdata/43.bmp 1 14 25 276 265
rawdata/44.bmp 1 46 46 313 312
```

Fig 4.9: List of Positive Image’s info file contains its coordinate, file name and directory.

This info file of positive image also needs to be placed in the “haarcascade” folder.

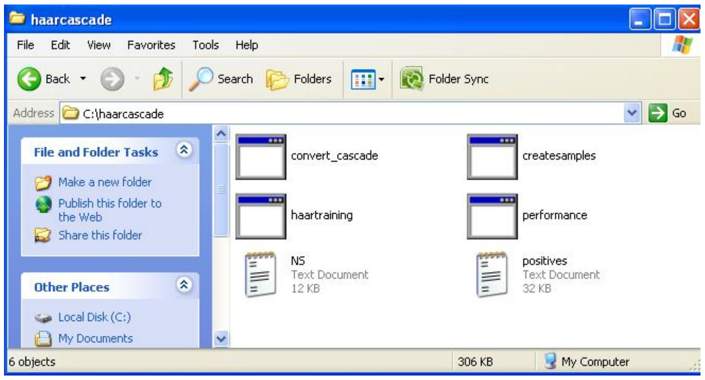


Fig 4.10: Info file of Positive Image named as “positives” is being placed in the “haarcascade” folder.

4.1.6 Result of samples creation

After create samples process is done, a .vec file which means a vector file is created automatically by OpenCV utilities –createsamples. This vector file is a sample file which contains the collection of sample that had been converted to undefined structure. The contents of vector file should not be opened as it shows the unknown structures which can’t be understand easily. The vector file will be used during training the classifier. As reference, the vector file can be opened and compare the result with following figure.

The contents of vector file will be something that is similar to following figure:

Fig 4.11: The Contents of Vector File

Vector file is an output file generated from sample creating process. In the folder "haarcascade" now there is a vector file created such in a figure below:

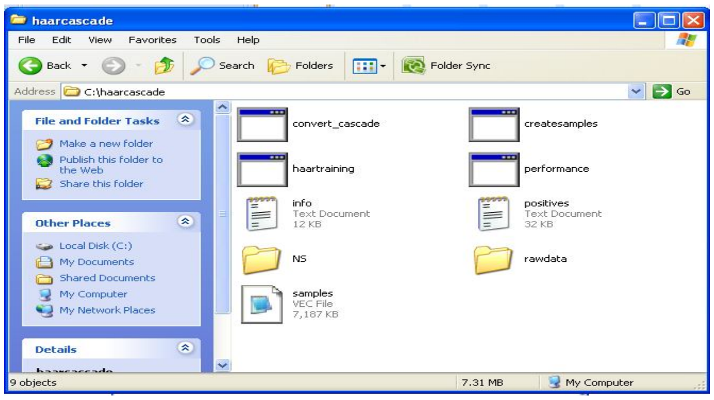
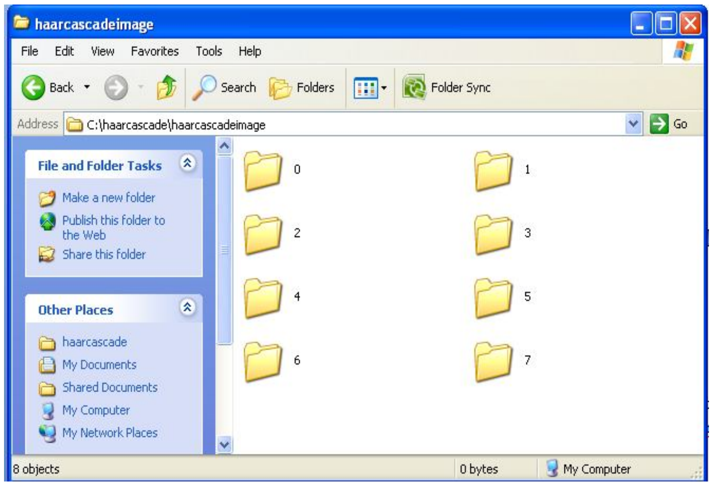


Fig 4.12: Vector file created in "haarcascade" folder after create samples process

4.1.7 Result of haartraining process

Haartraining process is a method for calculating the weak constraints of the object using it's formula. All the calculation will be done automatically by haartraining utilities. During the hartraining process, a folder of weak constraints calculation will be created in the "haarcascade" folder which had been named as "haarcascadeimage". In the folder of "haarcascadeimage", the folder of stages will be created automatically during the haartraining process. The number of stages created is determined during the coding implementation and its number must be larger than fourteen stages. Haartraining process will be calculating the thresholds according to the total of stages choose but sometimes, the process will be terminated. Even if increasing the number of stages, the training may finish in an intermediate stage when it exceeded desired minimum hit rate or false alarm because more cascading will decrease these rate for sure (0.99 until current * 0.99 next = 0.9801 until next). Or, the training may finish because all samples were rejected. In the case, increasing number of training samples is a must.

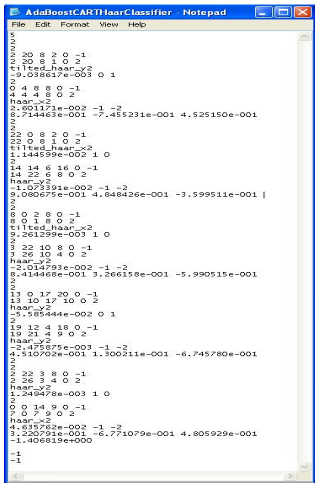
The result after threshold calculation is described by following figure:



C:\haarcascade\haarcascadeimage

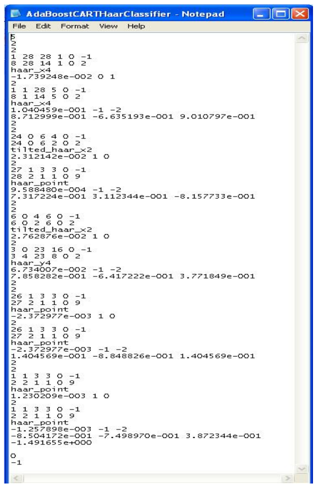
8 objects

Fig 4.13: The Stages of Weak Constraints Calculation


AdaBoostCARTHaarClassifier - Notepad
File Edit Format View Help

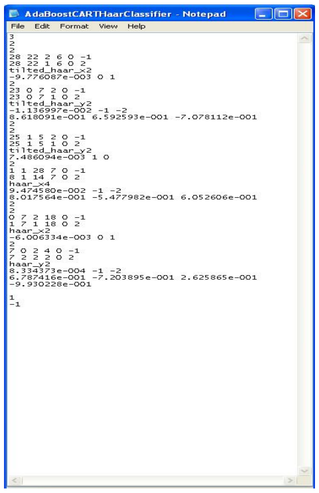
```
5
2
2
2 20 8 2 0 -1
2 20 8 1 0 2
tilted_haar_y2
-9.038617e-003 0 1
2
0 4 8 8 0 -1
4 4 4 8 0 2
haar_x2
2.601171e-002 -1 -2
8.714463e-001 -7.455231e-001 4.525150e-001
2
2
22 0 8 2 0 -1
22 0 8 1 0 2
tilted_haar_x2
1.144599e-002 1 0
2
14 14 6 16 0 -1
14 22 6 8 0 2
haar_y2
-1.073391e-002 -1 -2
9.080675e-001 4.848426e-001 -3.599511e-001 |
2
2
8 0 2 8 0 -1
8 0 1 8 0 2
tilted_haar_x2
9.261299e-003 1 0
2
3 22 10 8 0 -1
3 26 10 4 0 2
haar_y2
-2.014793e-002 -1 -2
8.414468e-001 3.266158e-001 -5.990515e-001
2
2
13 0 17 20 0 -1
13 10 17 10 0 2
haar_y2
-5.585444e-002 0 1
2
19 12 4 18 0 -1
19 21 4 9 0 2
haar_y2
-2.475875e-003 -1 -2
4.510702e-001 1.300211e-001 -6.745780e-001
2
2
2 22 3 8 0 -1
2 26 3 4 0 2
haar_y2
1.249478e-003 1 0
2
0 0 14 9 0 -1
7 0 7 9 0 2
haar_x2
4.635762e-002 -1 -2
3.220791e-001 -6.771079e-001 4.805929e-001
-1.406819e+000

-1
-1
```

a) **Stage 0**


AdaBoostCARTHaarClassifier - Notepad
File Edit Format View Help

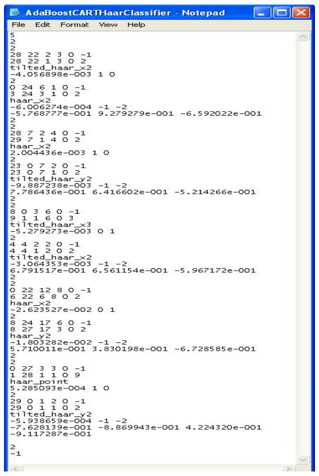
```
5
2
2
1 28 28 1 0 -1
8 28 14 1 0 2
haar_x4
-1.739248e-002 0 1
2
1 1 28 5 0 -1
8 1 14 5 0 2
haar_x4
1.040459e-001 -1 -2
8.712999e-001 -6.635193e-001 9.010797e-001
2
2
24 0 6 4 0 -1
24 0 6 2 0 2
tilted_haar_x2
2.312142e-002 1 0
2
27 1 3 3 0 -1
28 2 1 1 0 9
haar_point
9.588480e-004 -1 -2
7.317224e-001 3.112344e-001 -8.157733e-001
2
2
6 0 4 6 0 -1
6 0 2 6 0 2
tilted_haar_x2
2.762876e-002 1 0
2
3 0 23 16 0 -1
3 4 23 8 0 2
haar_y4
6.734007e-002 -1 -2
7.858282e-001 -6.417222e-001 3.771849e-001
2
2
26 1 3 3 0 -1
27 2 1 1 0 9
haar_point
-2.372977e-003 1 0
2
26 1 3 3 0 -1
27 2 1 1 0 9
haar_point
-2.372977e-003 -1 -2
1.404569e-001 -8.848826e-001 1.404569e-001
2
2
1 1 3 3 0 -1
2 2 1 1 0 9
haar_point
1.230209e-003 1 0
2
1 1 3 3 0 -1
2 2 1 1 0 9
haar_point
-1.257898e-003 -1 -2
-8.504172e-001 -7.498970e-001 3.872344e-001
-1.491655e+000

0
-1
```

b) **Stage 1**


AdaBoostCARTHaarClassifier - Notepad
File Edit Format View Help

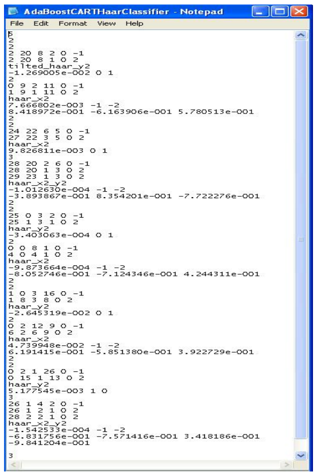
```
3
2
2
28 22 2 6 0 -1
28 22 1 6 0 2
tilted_haar_x2
-9.776087e-003 0 1
2
23 0 7 2 0 -1
23 0 7 1 0 2
tilted_haar_y2
-1.136997e-002 -1 -2
8.618091e-001 6.592593e-001 -7.078112e-001
2
2
25 1 5 2 0 -1
25 1 5 1 0 2
tilted_haar_y2
7.486094e-003 1 0
2
1 1 28 7 0 -1
8 1 14 7 0 2
haar_x4
9.474580e-002 -1 -2
8.017564e-001 -5.477982e-001 6.052606e-001
2
2
0 7 2 18 0 -1
1 7 1 18 0 2
haar_x2
-6.006334e-003 0 1
2
7 0 2 4 0 -1
7 2 2 2 0 2
haar_y2
8.334373e-004 -1 -2
6.787416e-001 -7.203895e-001 2.625865e-001
-9.930228e-001

1
-1
```

c) **Stage 2**


AdaBoostCARTHaarClassifier - Notepad
File Edit Format View Help

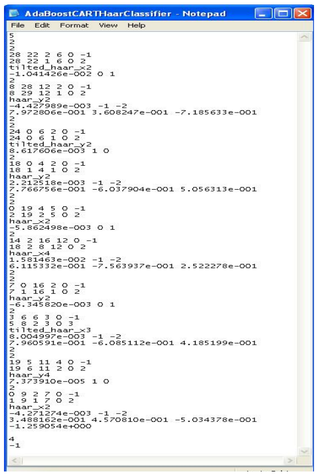
```
5
2
2
28 22 2 3 0 -1
28 22 1 3 0 2
tilted_haar_x2
-4.056898e-003 1 0
2
0 24 6 1 0 -1
3 24 3 1 0 2
haar_x2
-6.006274e-004 -1 -2
-5.768777e-001 9.279279e-001 -6.592022e-001
2
2
28 7 2 4 0 -1
29 7 1 4 0 2
haar_x2
2.004436e-003 1 0
2
23 0 7 2 0 -1
23 0 7 1 0 2
tilted_haar_y2
-9.887238e-003 -1 -2
7.786436e-001 6.416602e-001 -5.214266e-001
2
2
8 0 3 6 0 -1
9 1 1 6 0 3
tilted_haar_x3
-5.279273e-003 0 1
2
4 4 2 2 0 -1
4 4 1 2 0 2
tilted_haar_x2
-3.064353e-003 -1 -2
6.791517e-001 6.561154e-001 -5.967172e-001
2
2
0 22 12 8 0 -1
6 22 6 8 0 2
haar_x2
-2.623527e-002 0 1
2
8 24 17 6 0 -1
8 27 17 3 0 2
haar_y2
-1.803282e-002 -1 -2
5.710011e-001 3.830198e-001 -6.728585e-001
2
2
0 27 3 3 0 -1
1 28 1 1 0 9
haar_point
5.285093e-004 1 0
2
29 0 1 2 0 -1
29 0 1 1 0 2
tilted_haar_y2
-5.938659e-004 -1 -2
-7.628139e-001 -8.869943e-001 4.224320e-001
-9.117287e-001

2
-1
```

d) **Stage 3**


AdaBoostCARTHaarClassifier - Notepad
File Edit Format View Help

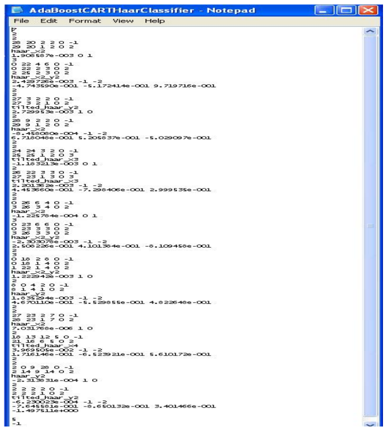
```
5
2
2
2 20 8 2 0 -1
2 20 8 1 0 2
tilted_haar_y2
-1.269005e-002 0 1
2
0 9 2 11 0 -1
1 9 1 11 0 2
haar_x2
7.666802e-003 -1 -2
8.418972e-001 -6.163906e-001 5.780513e-001
2
2
24 22 6 5 0 -1
27 22 3 5 0 2
haar_x2
9.826811e-003 0 1
3
28 20 2 6 0 -1
28 20 1 3 0 2
29 23 1 3 0 2
haar_x2_y2
-1.012630e-004 -1 -2
-3.893867e-001 8.354201e-001 -7.722276e-001
2
2
25 0 3 2 0 -1
25 1 3 1 0 2
haar_y2
-3.403063e-004 0 1
2
0 0 8 1 0 -1
4 0 4 1 0 2
haar_x2
-9.873664e-004 -1 -2
-8.052746e-001 -7.124346e-001 4.244311e-001
2
2
1 0 3 16 0 -1
1 8 3 8 0 2
haar_y2
-2.645319e-002 0 1
2
0 2 12 9 0 -1
6 2 6 9 0 2
haar_x2
4.739948e-002 -1 -2
6.191415e-001 -5.851380e-001 3.922729e-001
2
2
0 2 1 26 0 -1
0 15 1 13 0 2
haar_y2
5.177545e-003 1 0
3
26 1 4 2 0 -1
26 1 2 1 0 2
28 2 2 1 0 2
haar_x2_y2
-1.542533e-004 -1 -2
-6.831756e-001 -7.571416e-001 3.418186e-001
-9.841204e-001

3
```

e) **Stage 4**


AdaBoostCARTHaarClassifier - Notepad
File Edit Format View Help

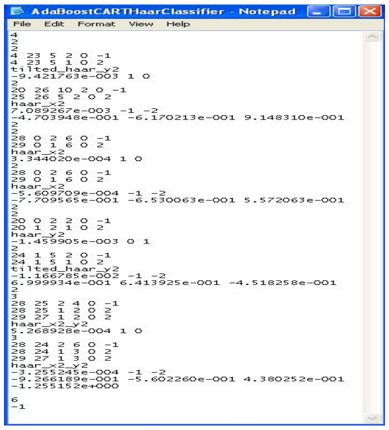
```
5
2
2
28 22 2 6 0 -1
28 22 1 6 0 2
tilted_haar_x2
-1.041426e-002 0 1
2
8 28 12 2 0 -1
8 29 12 1 0 2
haar_y2
-4.427989e-003 -1 -2
7.972806e-001 3.608247e-001 -7.185633e-001
2
2
24 0 6 2 0 -1
24 0 6 1 0 2
tilted_haar_y2
8.617606e-003 1 0
2
18 0 4 2 0 -1
18 1 4 1 0 2
haar_y2
2.212518e-003 -1 -2
7.766756e-001 -6.037904e-001 5.056313e-001
2
2
0 19 4 5 0 -1
2 19 2 5 0 2
haar_x2
-5.862498e-003 0 1
2
14 2 16 12 0 -1
18 2 8 12 0 2
haar_x4
1.581463e-002 -1 -2
6.115332e-001 -7.563937e-001 2.522278e-001
2
2
7 0 16 2 0 -1
7 1 16 1 0 2
haar_y2
-6.345820e-003 0 1
2
3 6 6 3 0 -1
5 8 2 3 0 3
tilted_haar_x3
8.004997e-003 -1 -2
7.960591e-001 -6.085112e-001 4.185199e-001
2
2
19 5 11 4 0 -1
19 6 11 2 0 2
haar_y4
7.373910e-005 1 0
2
0 9 2 7 0 -1
1 9 1 7 0 2
haar_x2
-4.271274e-003 -1 -2
3.488162e-001 4.570810e-001 -5.034378e-001
-1.259054e+000

4
-1
```

f) **Stage 5**


AdaBoostCARTHaarClassifier - Notepad
File Edit Format View Help

```
7
2
2
28 20 2 2 0 -1
29 20 1 2 0 2
haar_x2
1.908587e-003 0 1
3
0 22 4 6 0 -1
0 22 2 3 0 2
2 25 2 3 0 2
haar_x2_y2
2.429726e-003 -1 -2
-4.743590e-001 -5.172414e-001 9.719716e-001
2
2
27 3 2 2 0 -1
27 3 2 1 0 2
tilted_haar_y2
2.729953e-003 1 0
2
28 9 2 2 0 -1
29 9 1 2 0 2
haar_x2
-8.458080e-004 -1 -2
6.718048e-001 5.205837e-001 -5.029097e-001
2
2
24 24 3 2 0 -1
25 25 1 2 0 3
tilted_haar_x3
-1.183213e-003 0 1
2
26 22 3 3 0 -1
27 23 1 3 0 3
tilted_haar_x3
2.201362e-003 -1 -2
4.453660e-001 -7.298406e-001 2.999535e-001
2
2
0 26 6 4 0 -1
3 26 3 4 0 2
haar_x2
-1.225784e-004 0 1
3
0 23 6 6 0 -1
0 23 3 3 0 2
3 26 3 3 0 2
haar_x2_y2
-2.303078e-003 -1 -2
2.508226e-001 4.101384e-001 -8.109458e-001
2
3
0 18 2 8 0 -1
0 18 1 4 0 2
1 22 1 4 0 2
haar_x2_y2
1.222942e-003 1 0
2
8 0 4 2 0 -1
8 1 4 1 0 2
haar_y2
1.835294e-003 -1 -2
4.670110e-001 -5.529855e-001 4.822648e-001
2
2
27 23 2 7 0 -1
28 23 1 7 0 2
haar_x2
7.031768e-006 1 0
2
18 13 12 5 0 -1
21 16 6 5 0 2
tilted_haar_x4
3.969505e-002 -1 -2
1.716146e-001 -6.523921e-001 5.610172e-001
2
2
2 0 9 28 0 -1
2 14 9 14 0 2
haar_y2
-2.313831e-004 1 0
2
2 2 2 2 0 -1
2 2 2 1 0 2
tilted_haar_y2
-6.230023e-004 -1 -2
-7.645581e-001 -8.650132e-001 3.401466e-001
-1.497511e+000

5
-1
```

g) **Stage 6**


AdaBoostCARTHaarClassifier - Notepad
File Edit Format View Help

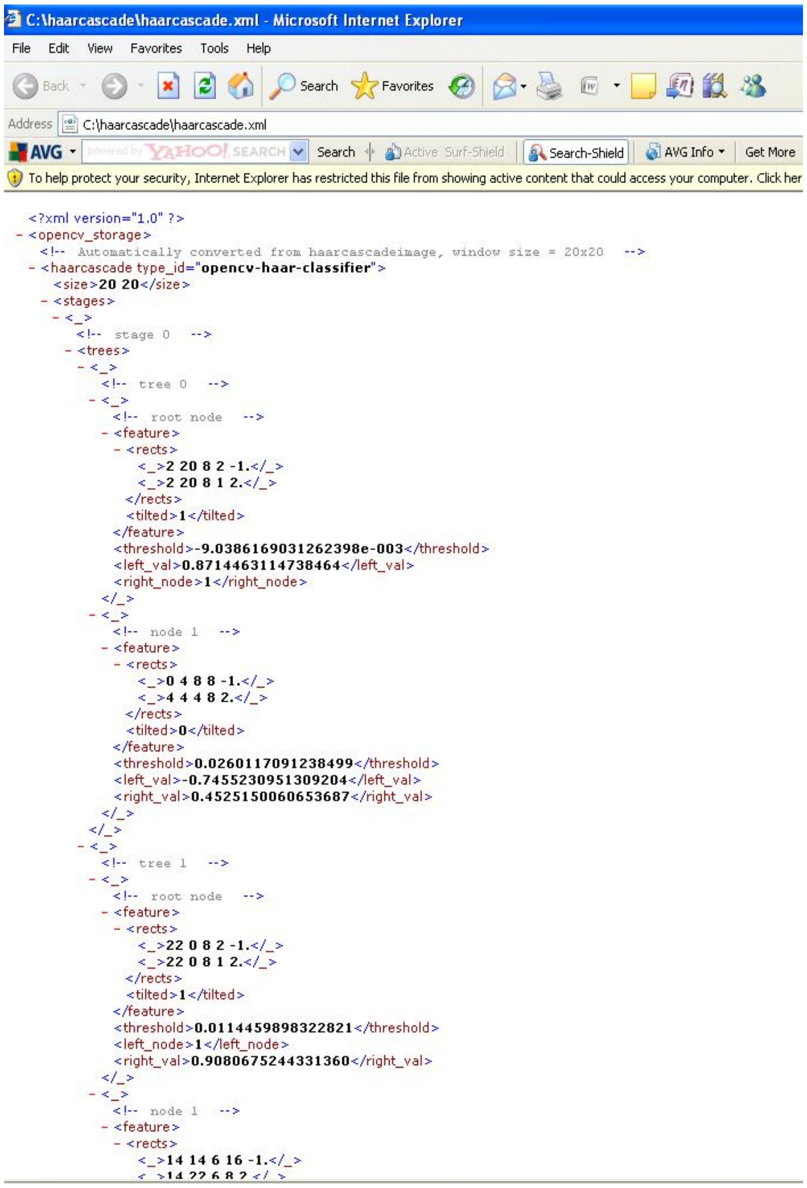
```
4
2
2
4 23 5 2 0 -1
4 23 5 1 0 2
tilted_haar_y2
-9.421763e-003 1 0
2
20 26 10 2 0 -1
25 26 5 2 0 2
haar_x2
7.089267e-003 -1 -2
-4.703948e-001 -6.170213e-001 9.148310e-001
2
2
28 0 2 6 0 -1
29 0 1 6 0 2
haar_x2
3.344020e-004 1 0
2
28 0 2 6 0 -1
29 0 1 6 0 2
haar_x2
-5.609709e-004 -1 -2
-7.709565e-001 -6.530063e-001 5.572063e-001
2
2
20 0 2 2 0 -1
20 1 2 1 0 2
haar_y2
-1.459905e-003 0 1
2
24 1 5 2 0 -1
24 1 5 1 0 2
tilted_haar_y2
-1.166785e-002 -1 -2
6.999934e-001 6.413925e-001 -4.518258e-001
2
3
28 25 2 4 0 -1
28 25 1 2 0 2
29 27 1 2 0 2
haar_x2_y2
5.268928e-004 1 0
3
28 24 2 6 0 -1
28 24 1 3 0 2
29 27 1 3 0 2
haar_x2_y2
-3.255245e-004 -1 -2
-9.266189e-001 -5.602260e-001 4.380252e-001
-1.255152e+000

6
-1
```

h) **Stage 7**

Fig 4.14: The files contains of Weak Constraints Calculation according to stages.

4.1.8 Results of xml file creation.

Xml file is generated when the process is completely finished. Xml file is a final result of training process. Implementations of the xml file to the coding start the coding process. Xml file contains a list of threshold created during the haartraining process. The example of xml file is shown in figure below.


C:\haarcascade\haarcascade.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address C:\haarcascade\haarcascade.xml
AVG
YAHOO! SEARCH
Search
Active Surf-Shield
Search-Shield
AVG Info
Get More

To help protect your security, Internet Explorer has restricted this file from showing active content that could access your computer. Click her

```
  <?xml version="1.0" ?>
- <opencv_storage>
    <!-- Automatically converted from haarcascadeimage, window size = 20x20   -->
  - <haarcascade type_id="opencv-haar-classifier">
      <size>20 20</size>
    - <stages>
      - <_>
          <!-- stage 0   -->
        - <trees>
          - <_>
              <!-- tree 0   -->
            - <_>
                <!-- root node   -->
              - <feature>
                - <rects>
                    <_>2 20 8 2 -1.</_>
                    <_>2 20 8 1 2.</_>
                  </rects>
                  <tilted>1</tilted>
                </feature>
                <threshold>-9.0386169031262398e-003</threshold>
                <left_val>0.8714463114738464</left_val>
                <right_node>1</right_node>
              </_>
            - <_>
                <!-- node 1   -->
              - <feature>
                - <rects>
                    <_>0 4 8 8 -1.</_>
                    <_>4 4 4 8 2.</_>
                  </rects>
                  <tilted>0</tilted>
                </feature>
                <threshold>0.0260117091238499</threshold>
                <left_val>-0.7455230951309204</left_val>
                <right_val>0.4525150060653687</right_val>
              </_>
            </_>
          - <_>
              <!-- tree 1   -->
            - <_>
                <!-- root node   -->
              - <feature>
                - <rects>
                    <_>22 0 8 2 -1.</_>
                    <_>22 0 8 1 2.</_>
                  </rects>
                  <tilted>1</tilted>
                </feature>
                <threshold>0.0114459898322821</threshold>
                <left_node>1</left_node>
                <right_val>0.9080675244331360</right_val>
              </_>
            - <_>
                <!-- node 1   -->
              - <feature>
                - <rects>
                    <_>14 14 6 16 -1.</_>
                    <_>14 22 6 8 2.</_>
```

Fig 4.15: Xml file contains of calculation of weak constraints created during Haartraining process.

4.2 Execution phase results

Execution phase cause the xml file to be implemented in the coding development. The detection process is done in execution phase. The simple user's interface had been created for detection process. When the button "Detect ball" is clicked, the pop up window of result after detection process is generated automatically. The figure below shows the simple user's interface of object detection system.

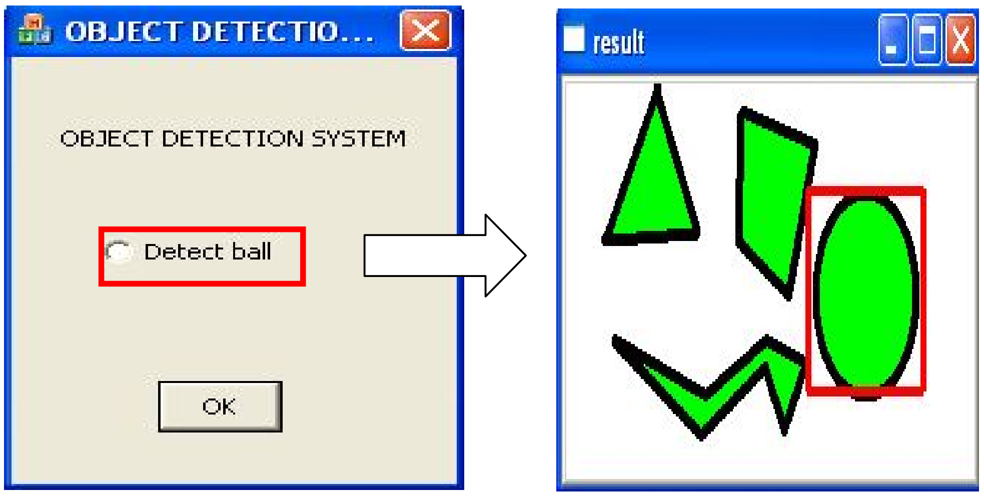


Figure 4.16: "Result" window will generate automatically when the button "Detect Ball" is clicked.

Another example of tested result is shown in figure below:

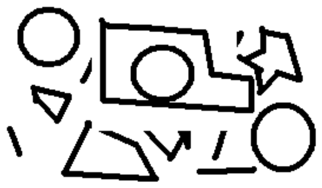
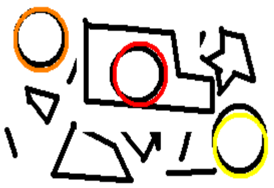
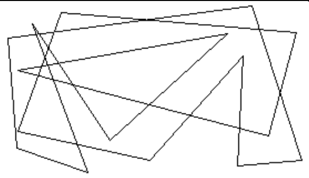
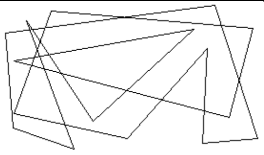
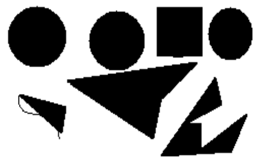
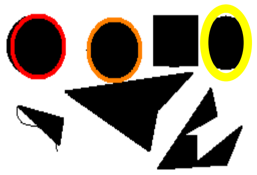
No.	Image Before Detection	Image After Detection
1.		
2.		
3.		

Fig 4.18: Detection Results of accurate detection

In object detection system, the problem occurs is the matter of accuracy. In some images, the circle line can also been drawn at the empty space in the images. Figure below will show the example of this matter.

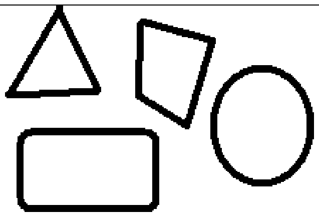
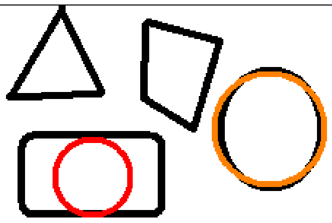
Before Detection	After Detection
	

Fig 4.18: Detection Results of not accurate detection

The matter of accuracy can be solved by increasing the numbers of sample collected and must make sure that each negative image never has any image of object of interest. The suggested number of samples is 7000 of both positive and negative images.

4.3 Costing

The overall cost during the project development is about RM 5000. This is the price value of the software used which is Visual Studio 2005. The price is valuable as the desired result can be achieved. Apart from that, there are no other cost contribute in the development of this system.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

The final chapter summarizes the whole system's overview. All the method used in this system satisfied the system development. Haar-Classifier is a very effective algorithm that can contribute to high detection rate.

5.1 Conclusion

In the process of developing this system, few process and features are needed as an attribution to complete the system. For object detection process, the image should be scaled and stored to the computer before the detection process can be done. The Haar-like features is used as a algorithm or classifier to find weak classifier to differentiate the desired object from others objects.

Compared with raw pixel values, Haar-like features can reduce/increase the in-class/out-of-class variability, and thus making classification easier. OpenCV has a Haar features based face detection module. Use local features such as edges and line patterns. It scans a given image at different scales as in template matching. Scale, translation and light invariant.

5.2 Future Recommendation

As to improve other system with higher detection rate and more interactive system, there are some recommendations listed which are:

i) The accuracy rate of Haar-Classifier will perform better if the sample collected is increased. The suggested numbers of sample collected is 7000 for both positive and negative images. Therefore this system needs the algorithm and classifier that can detect object faster with just a small numbers of image collections.

ii) This system application is not enough exposed with the user friendly concept because user need to change the image file before detection process could be done. Real-time system is a solution attribute to the user friendly concept.

5.2.1 Commercialization

Application of the system can be widely used as the way to improve the used of technologies in the country. Systems approach can be applied to the robotic system for robot to verify object and help in kids learning process by applying this system to kids play tools

REFERENCES

I) Paul Viola, Michael J.Jones, Robust Real- time Object Detection, Cambridge Research Laboratory Technology Reports Series, CRL 2001/01, Feb, 2001

II) Rainer Lienhart, Alexander Kuranov, Vadim Pisarevsky, Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection, Intel Labs MRL Technology Report, May 2002

III) Paul Viola, Michael J.Jones, Robust Real- time Object Detection, ICCV 2001

IV) Paul Viola and Michael Jones, Fast and Robust Classification using Asymmetric AdaBoost and a Detector Cascade, Neural Information Processing Systems 14, December 2001

V) Rainer Lienhart, Luhong Liang, and Alexander Kuranov, A Detector Tree of Boosted Classifiers for Real-time Object detection and Tracking, ICME 2003

VI) Azriel Rosenfeld, *Picture Processing by Computer*, New York: Academic Press, 1969

VII) Axt, P. (1959) On a Subrecursive Hierarchy and Primitive Recursive Degrees, *Transactions of the American Mathematical Society* 92, pp. 85–105

VIII) Canny, J., A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714

APPENDIX

APPENDIX A - DETECTION PROCESS SOURCE CODE

```
// OpenCV Sample Application: objectdetect.c

// Include header files
#include "cv.h"
#include "highgui.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <math.h>
#include <float.h>
#include <limits.h>
#include <time.h>
#include <ctype.h>

static CvMemStorage* storage = 0;
static CvHaarClassifierCascade* cascade = 0;
void detect_and_draw( IplImage* image );
const char* cascade_name =
    "haarcascade.xml";
/*    "haarcascade.xml";*/
int main( int argc, char** argv )
{
    IplImage *frame, *frame_copy = 0;
    const char* input_name;
    if( argc > 1 && strncmp( argv[1], "--cascade=", optlen ) == 0 )
    {
        cascade_name = argv[1] + optlen;
        input_name = argc > 2 ? argv[2] : 0;
    }
    else
    {
        fprintf( stderr,
        "Usage: objectdetect --cascade=\"<cascade_path>\"
        return -1;
    cascade = (CvHaarClassifierCascade*)cvLoad( cascade_name, 0, 0, 0
);
    if( !cascade )
    {
        fprintf( stderr, "ERROR: Could not load classifier cascade\n"
);
        return -1;
    }

    storage = cvCreateMemStorage(0);
    if( !input_name || (isdigit(input_name[0]) && input_name[1] ==
        capture = cvCaptureFromAVI( input_name );

    cvNamedWindow( "result", 1 );
    if( capture )
    {
        for(;;)
        {
            if( !cvGrabFrame( capture ))
```

```
                break;
            frame = cvRetrieveFrame(
            if( !frame )
                break;
            if( !frame_copy )
            frame_copy = cvCreateImage( cvSize(frame->width,frame-
            >height),
            IPL_DEPTH_8U, frame->nChannels );
frame to frame_copy.
            if( frame->origin == IPL_ORIGIN_TL )
                cvCopy( frame, frame_copy, 0 );
            else
                cvFlip( frame, frame_copy, 0 );

            detect_and_draw( frame_copy );
            if( cvWaitKey( 10 ) >= 0 )
                break;
        }
        cvReleaseImage( &frame_copy );
    }
    else
    {
        const char* filename = input_name ? input_name : (char
        IplImage* image = cvLoadImage( filename, 1 );
        if( image )
        {
            detect_and_draw( image );

            cvWaitKey(0);

            cvReleaseImage( &image );
        }
        else
        {

            FILE* f = fopen( filename, "rt" );
            if( f )
            {
                char buf[1000+1];


                while( fgets( buf, 1000, f ) )
                {

                    int len = (int)strlen(buf);
                    while( len > 0 && isspace(buf[len-1]) )
                        len--;
                    buf[len] = '\0';


                    image = cvLoadImage( buf, 1 );


                    if( image )
                    {

                        detect_and_draw( image );
                        cvWaitKey(0);
                        cvReleaseImage( &image );
                    }
                }
                fclose(f);
            }
        }

    }


    cvDestroyWindow("result");


    return 0;
}

void detect_and_draw( IplImage* img )
{
    int scale = 1;

    IplImage* temp = cvCreateImage( cvSize(img->width/scale,img-
>height/scale), 8, 3 );
    CvPoint pt1, pt2;
    int i;
    cvClearMemStorage( storage );


    if( cascade )
    {
            CvSeq* balls = cvHaarDetectObjects( img, cascade, storage,
            1.1, 2, CV_HAAR_DO_CANNY_PRUNING, cvSize(40, 40) );
            for( i = 0; i < (balls ? balls->total : 0); i++ )
        {
            CvRect* r = (CvRect*)cvGetSeqElem( balls, i );


            pt1.x = r->x*scale;
            pt2.x = (r->x+r->width)*scale;
            pt1.y = r->y*scale;
            pt2.y = (r->y+r->height)*scale;

            // Draw the rectangle in the input image
            cvCircle( img, pt1, pt2, CV_RGB(255,0,0), 3, 8, 0 );
        }
    }

    // Show the image in the window named "result"
    cvShowImage( "result", img );

    // Release the temp image created.
    cvReleaseImage( &temp );
}
```