A MAJOR PROJECT REPORT

ON

HELMET DETECTION USING MACHINE LEARNING

SUBMITTED IN PARTIAL FULFILLMENT FOR THE AWARD OF DEGREE OF

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted By:

ANKITA PURKAYASTHA (9915102140)

DIKSHANT MANOCHA (9915102071)

YATIN CHACHRA (9915102088)

Under the Guidance Of

MR. VARUN GOEL

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA (U.P.)

December, 2018

CERTIFICATE

This is to certify that the major project report entitled, "**HELMET DETECTION USING MACHINE LEARNING**" submitted by **ANKITA PURKAYASTHA (9915102140), DIKSHANT MANOCHA (9915102071), YATIN CHACHRA (9915102088)** in partial fulfilment of the requirements for the award of Bachelor of Technology Degree in **Electronics and Communication Engineering** of the Jaypee Institute of Information Technology, Noida isan authentic work carried out by them under my supervision and guidance. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Signature of Supervisor:

Name of the Supervisor:

ECE Department,

JIIT, Sec-128,

Noida-201304

Dated:

DECLARATION

We hereby declare that this written submission represents our own ideas in our own words,and where others ideas or words have been included have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission.

Place:

Date:

Name: ANKITA PURKAYASTHA

Enrolment: 9915102140

Name: DIKSHANT MANOCHA

Enrolment: 9915102071

Name: YATIN CHACHRA

Enrolment: 9915102088

ABSTRACT

Road traffic safety refers to the methods and measures used toprevent road users from being killed or seriously injured. Typical road users include pedestrians, cyclists, motorists, vehicle passengers and passengers of on-road public transport(mainly buses and trams).

Many countries around the world are facing the problem of a rapidly rising number of people injured or killed while riding two-wheelers – motorcycles and bicycles. A large proportion of the deaths and severe injuries result from injuries to the head. Helmets are effective in reducing the likelihood of head injuries, as well as their severity. Increasing helmet use in a country is thus an important way of improving road safety.

A motorcycle crash can result in deadly head injuries that can often be fatal. Doctors say, when your head hits the pavement or the ground your brain is moves forward, hitting up against the bones inside the skull. It gets deformed and tears nerve fibers. The torn ones cannot heal. When you lose a brain cell, there is no replacement for it. That's where permanent damage occurs. People who have an accident like that, and survive, often don't fully recover. They may lose some intelligence, and the capacity to take care of themselves because of the damage to the system that controls their muscles. They may have a behavior change – have difficulty dealing with other people, and having proper social relationships. All this, just because a motorist did not wear a helmet.

Our aim is to judge people riding their two wheelers without helmets. The people found violating the two wheeler norms will be penalized.

ACKNOWLDGEMENT

Education along with process of gaining knowledge and stronghold subject is a continuous and on-going process. It is an appropriate blend of mind-set, learnt skills, experience and knowledge gained from various resources. The project would not have been possible without the support of many people.

We would like to express our heartfelt gratitude and indebtedness to our mentor, **Mr. Varun Goel** for being a constant pillar of support and guidance. We would also like to thank our friends without whom this project would have been a distant reality.

Thank You.

Table of Contents

List of Figures

CHAPTER - 1

INTRODUCTION

1.1Introduction

Two-wheeler is a very popular mode of transportation inalmost every country. However,there is a high risk involvedbecause of less protection. Since, motorcycles are affordable and a daily mode oftransport, there has been a rapid increase in motorcycle accidentsdue to the fact that most of the motorcyclists do notwear a helmet which makes it an ever-present danger everyday to travel by motorcycle. To reduce the involved risk, it ishighly desirable for bike-riders to use helmet. In the last couple of yearsalone most of the deaths in accidents are due to damage in the head. Observing theusefulness of helmet, Governments have made it a punishableoffense to ride a bike without helmet and have adopted manualstrategies to catch the violators. Stillalarge number of motorcyclists do not obey the rule.Automation of this process is highlydesirable for reliable and robust monitoring of these violationsas well as it also significantly reduces the amount of humanresources needed. Also, many countries are adopting systemsinvolving surveillance cameras at public places. So, the solutionfor detecting violators using the existing infrastructure isalso cost-effective.

1.2Drawbacks of existing system

The existing system is completely non reliable as it is not automated, traffic cops are appointed to issue the challans which results in corruption and brings along many problems. The current system also dosen't involve the facility of paying challans from e=anywhere round the globe but our system will do.So the system now a days is completely undependable.

1.3 Our Methodology

- Real time image capturing of road traffic and processing to check whether the rider and pillion rider are wearing helmet or not.
- If any one of the rider and pillion rider found not wearing the helmet then the real time decision making process starts.
- After finding the riders not wearing the helmets their vehicle number plate will be processed.
- After extracting the vehicle registration number, a challan will be generated against respective vehicle and all the details of the challan will be sent via E-mail and SMS to the concerned person.
- Now its upto the owner whether he/she wants to pay the fine via our mobile app or via our website or by visiting the nearest RTO.

CHAPTER- 2

LITERATURE SURVEY

Motorcycle accidents have been rapidly growing throughout the years in many countries. Due to various social and economic factors, this type of vehicle is becoming increasingly popular. The helmet is the main safety equipment of motorcyclists, however many drivers do not use it. The main goal of helmet is to protect the drivers head in case of accident. In case of accident, if the motorcyclist does not use can be fatal. This paper aims to propose a system for detection of motorcyclist without helmet. For this, we have applied the circular Hough transform and the Histogram of Oriented Gradients descriptor to extract the image attributes.[1]

In this paper, we propose an approach for automatic detection of bike-riders without helmet using surveillance videos in real time. The proposed approach first detects bike riders from surveillance video using background subtraction and object segmentation. Then it determines whether bike-rider is using a helmet or not using visual features and binary classifier. Also, we present a consolidation approach for violation reporting which helps in improving reliability of the proposed approach. In order to evaluate our approach, we have provided a performance comparison of three widely used feature representations namely histogram of oriented gradients (HOG), scale-invariant feature transform (SIFT), and local binary patterns (LBP) for classification. The experimental results show detection accuracy of 93.80% on the real world surveillance data.[2]

Motorcycle accidents have been rapidly growing throughout the years in many countries. Due to various social and economic factors, this type of vehicle is becoming increasingly popular. The helmet is the main safety equipment of motorcyclists, but many drivers do not use it. If an motorcyclist is without helmet an accident can be fatal. This paper aims to explain and illustrate an automatic method for motorcycles detection and classification on

public roads and a system for automatic detection of motorcyclists without helmet. For this, a hybrid descriptor for features extraction is proposed based in Local Binary Pattern, Histograms of Oriented Gradients and the Hough Transform descriptors. Traffic images captured by cameras were used. The best result obtained from classification was an accuracy rate of 0.9767, and the best result obtained from helmet detection was an accuracy rate of 0.9423.[3]

CHAPTER - 3

TOOLS & TECHNIQUES

3.1 Tools Used

3.1.1 OpenCV

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing.

Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

3.1.1.1 Introduction

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

3.1.1.2 Haar Cascades

OpenCV comes with a trainer as well as detector. If you want to train your own classifier for any object like car, planes etc. you can use OpenCV to create one. Its full details are given here: Cascade Classifier Training.

Here we will deal with detection. OpenCV already contains many pre-trained classifiers for face, eyes, smiles, etc. Those XML files are stored in the opencv/data/haarcascades/ folder. Let's create a face and eye detector with OpenCV.

First we need to load the required XML classifiers. Then load our input image (or video) in grayscale mode.

```
importnumpyas np
import cv2 as cv
face_cascade =
cv.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv.CascadeClassifier('haarcascade_eye.xml')
img = cv.imread('sachin.jpg')
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
```

Now we find the faces in the image. If faces are found, it returns the positions of detected faces as Rect(x,y,w,h). Once we get these locations, we can create a ROI for the face and apply eye detection on this ROI (since eyes are always on the face !!!).

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
cv.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
roi_gray = gray[y:y+h, x:x+w]
roi_color = img[y:y+h, x:x+w]
eyes = eye_cascade.detectMultiScale(roi_gray)
for (ex,ey,ew,eh) in eyes:
cv.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
cv.imshow('img',img)
cv.waitKey(0)
cv.destroyAllWindows()
```

Result looks like below:

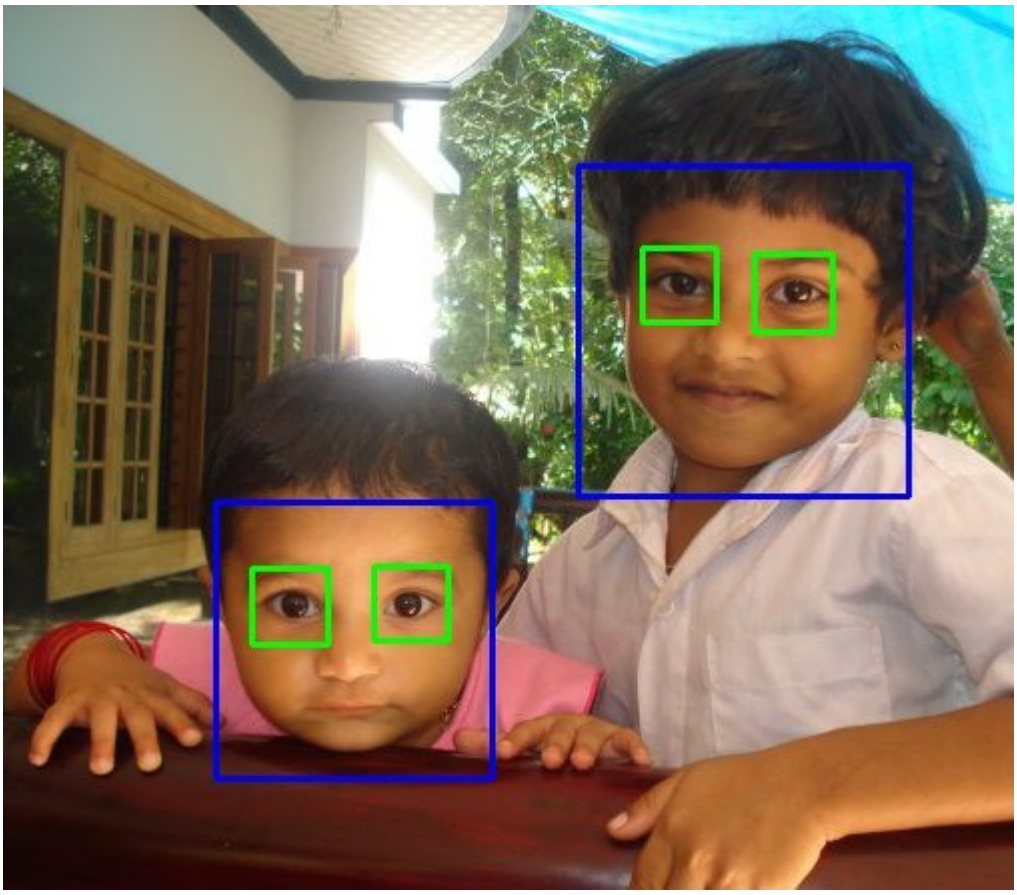


Fig 3.1. Face detection[4]

3.1.2 Java

Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented and specifically designed to have as few implementation dependencies as possible.

Java is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture.

The key features are:

- **Simple**

Java is easy to learn and its syntax is quite simple, clean and easy to understand. The confusing and ambiguous concepts of C++ are either left out in Java or they have been re-implemented in a cleaner way.

E.g.: Pointers and Operator Overloading are not there in java but were an important part of C++.

- **Object Oriented**

In java everything is Object which has some data and behaviour. Java can be easily extended as it is based on Object Model[4].

- **Robust**

Java makes an effort to eliminate error prone codes by emphasizing mainly on compile time error checking and runtime checking. But the main areas which Java improved were Memory Management and mishandled exceptions by introducing automatic Garbage Collector and Exception Handling.

- **Platform Independent**

Unlike other programming languages such as C, C++ etc. which are compiled into platform specific machines, Java is guaranteed to be write-once, run-anywhere language.

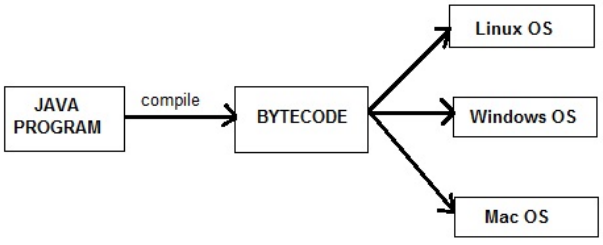


Fig 3.2. Java Platform Independent Diagram

On compilation Java program is compiled into bytecode. This bytecode is platform independent and can be run on any machine, plus this bytecode format also provides security. Any machine with Java Runtime Environment can run Java Programs.

- **Secure**

When it comes to security, Java is always the first choice. With java secure features it enables us to develop virus free, temper free system. Java program always runs in Java runtime environment with almost null interaction with system OS, hence it is more secure[5].

- **Multithreading**

Java multithreading feature makes it possible to write program that can do many tasks simultaneously. Benefit of multithreading is that it utilizes same memory and other resources to execute multiple threads at the same time, like While typing, grammatical errors are checked along[5].

- **Architectural Neutral**

Compiler generates bytecodes, which have nothing to do with particular computer architecture; hence a Java program is easy to interpret on any machine[5].

PortableJava Byte code can be carried to any platform. No implementation dependent features. Everything related to storage is predefined, example: size of primitive data types

- **High Performance**

Java is an interpreted language, so it will never be as fast as a compiled language like C or C++. But, Java enables high performance with the use of just-in-time compiler.

3.1.2.1 Java Runtime Environment Of Web

Java has strong support for web development. If one develop a web application (independent of the programming language one is using), one typically put their web application on a dedicated server (and not their local computer). The web application runs on the server and people can access it there. The server is either a real machine (with CPU, memory, harddisk, etc.) or a virtual server which is basically a machine which is separated by software into smaller machines.

Java web applications are typically not running directly on the server. Java web applications are running inside a web container on the server. The container provides a runtime environment for Java web applications. The container is for Java web applications what the JVM (Java Virtual Machine) is for local running Java applications. The container itself runs in the JVM. In general, Java distinguishes two containers: the web container and the Java EE container. Typical web containers in the Java world are Tomcat or Jetty. A web container supports the execution of Java servlets and JavaServer Pages. A Java EE container supports additional functionality, for example, distribution of server load.

Most of the modern Java web frameworks are based on servlets. Popular Java web frameworks are GWT, JavaServer Faces, Struts and the Spring framework. These web frameworks usually require as a minimum container a web container.

3.1.3 Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.The following features are provided in the current stable version:

- **Gradle-Based Build Support**

Gradle is an open-source build automation system that builds upon the concepts of Apache Ant and Apache Maven.Build automation is the process of automating the creation of a software build.

- **Android-Specific Code Refactoring and Quick Fixes**

Code refactoring is the process of restructuring existing computer code—changing the factoring—without changing its external behavior. Refactoring improves nonfunctional attributes of the software. Advantages include improved code readability and reduced complexity.

- **Lint tools** to catch performance, usability, version compatibility and other problems
- **ProguardIntegrationand App-Signing Capabilities**

ProGuard is an open source command-line tool that shrinks, optimizes and obfuscates Java code. It is able to optimize bytecode as well as detect and remove unused instructions

Note: Obfuscation is the deliberate act of creating source or machine code that is difficult for humans to understand.

Some other Features of Android Studio

- Template-based wizards to create common Android designs and components.
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations.
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine.
- Android Virtual Device (Emulator) to run and debug apps in the Android studio[8].
- Android Studio supports all the same programming languages of IntelliJ, Python, and Kotlin and Android Studio 3.0 supports "Java 7 language features and a subset of

Java8 language features that vary by platform version. External projects backport some Java 9 features.

3.1.3.1 TheAndroid Studio Working Environment

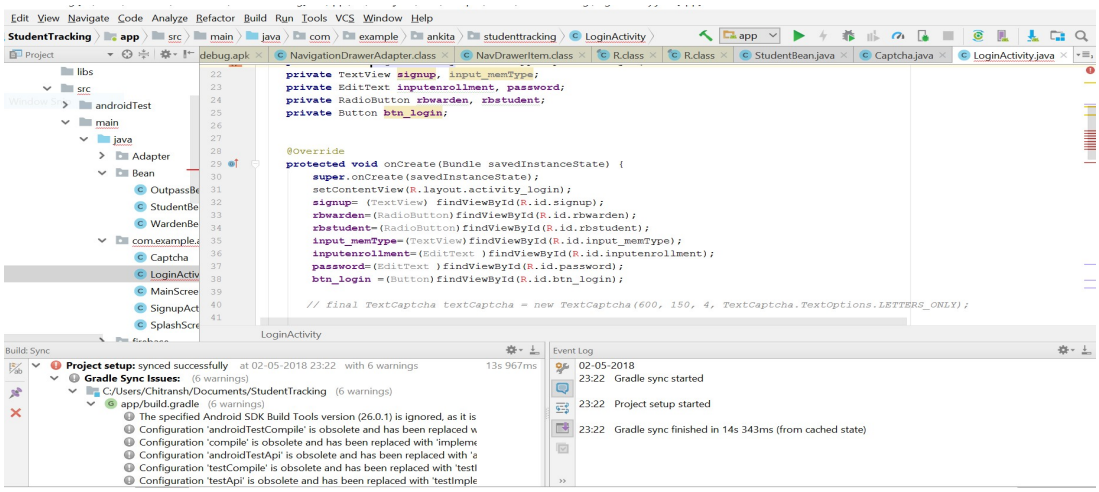


Fig 3.3 The Android IDE Desktop

- Android Studio is the official IDE for android application development. It works based on IntelliJ IDEA
- Android system initiates its program with in an **Activity** starting with a call on onCreate() callback method, which is similar to main() function in C/C++/Java[9].
- An activity class loads all the UI components using the XML file available in res/layout folder of the project. Activities are effectively ‘screens’ in an app.

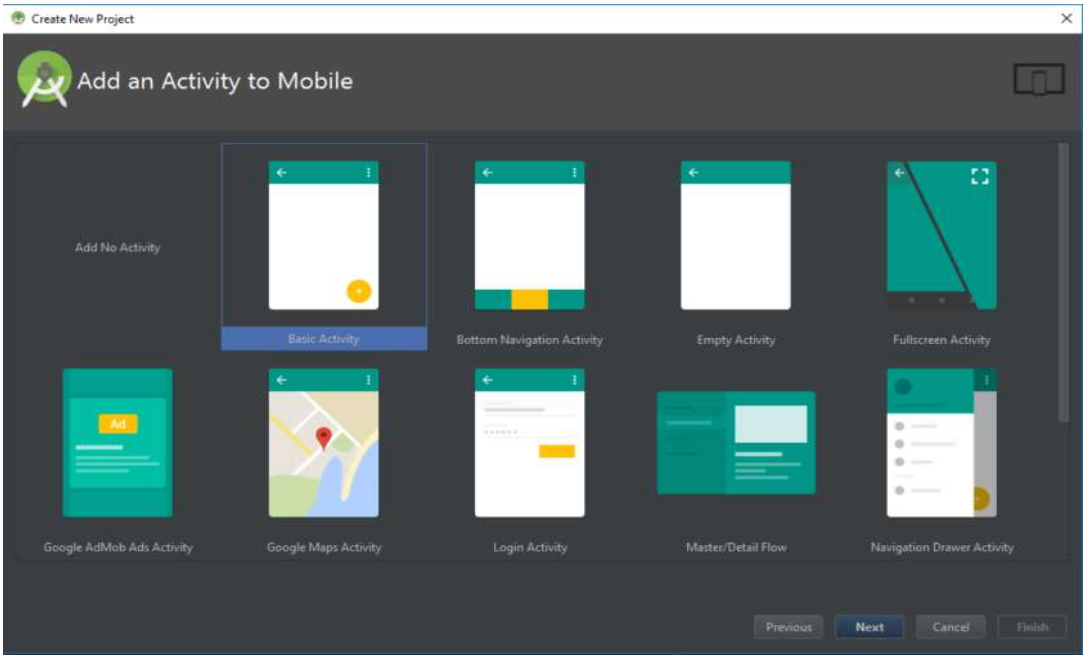


Fig. 3.4. Adding activity in Android app

- Often one will choose a 'Basic Activity', which is the default look and feel for a new Android App. This will include a menu in the top right corner, as well as a FAB button – Floating Action Button.
- If one wishes to open something new, then one will be able to do that through the file hierarchy on the left. Here one will find all the folders and the folders inside them. The project's Java files are housed under java and then the package name of the app.
- One can Double click on MainActivity.java and it will come to foreground in the window on the right.
- 'res' folder: This is short for 'resources' and that includes 'drawable' (images that one will place in the app) as well as 'layout' which is where the XML files go.

3.1.4 Heroku

Heroku is a cloud platform as a service (PaaS) supporting several programming languages. Heroku, one of the first cloud platforms, has been in development since June 2007, when it supported only the Ruby programming language, but now supports Java, Node.js, Scala, Clojure, Python, PHP, and Go. For this reason, Heroku is said to be a polyglot platform as it lets the developer build, run and scale applications in a similar manner across all the languages.

A diagrammatic view of the working of Heroku Platform:

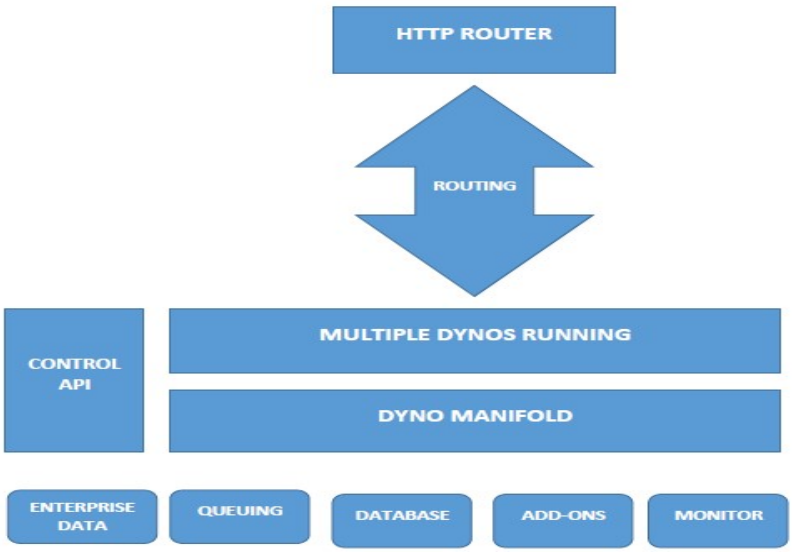


Fig 3.5. Working of Heroku Platform [10]

Applications that are run on Heroku typically have a unique domain (typically "applicationname.herokuapp.com") used to route HTTP requests to the correct dyno. Each of the application containers, or dynos, are spread across a "dyno grid" which consists of several servers. Heroku's Git server handles application repository pushes from permitted users. All Heroku services are hosted on Amazon's EC2 cloud-computing platform.

3.1.4.1 Working of Heroku Platform

The working can be summarized into two major categories:

1.**Deploy**;

The main content of the development are the source code, related dependencies if they exist, and a Procfile for the command.The application is sent to Heroku using either of the following: Git, GitHub, Dropbox, or via an API.There are packets which take the application along with all the dependencies, and the language runtime, and produce slugs. These are known as build-packs and are the means for the slug compilation process.A slug is a combination/bundle of the source code, built dependencies, the runtime, and compiled/generated output of the build system which is ready for execution.Next is the Config vars which contain the customizable configuration data that can be changed independently of the source code.Add-ons are third party, specialized, value-added cloud services that can be easily attached to an application, extending its functionality.A release is a combination of a slug (the application), config vars and add-ons.Heroku maintains a log known as the append-only ledger of releases the developer makes.

2. **Runtime**

The main unit which provides the run environment are the Dynos which are isolated, virtualized unix containers.The application's dyno formation is the total number of currently-executing dynos, divided between the various process types the developer has scaled.The dyno manager is responsible for managing dynos across all applications running on Heroku.Applications that use the free dyno type will sleep after 30 minutes of inactivity. Scaling to multiple web dynos, or a different dyno type, will avoid this.

One-off Dynos are temporary dynos that run with their input/output attached to the local terminal. They're loaded with the latest release.

Each dyno gets its own ephemeral filesystem with a fresh copy of the most recent release. It can be used as temporary scratchpad, but changes to the filesystem are not reflected to other dynos.Logplex automatically collates log entries from all the running dynos of the app, as well as other components such as the routers, providing a single source of activity.Scaling an application involves varying the number of dynos of each process type

3.1.4.2 Architecture of Heroku

The definition of the application i.e. the source code and the description is built on the framework provided by Heroku which converts it into an application. The dependency mechanisms vary across languages: for Ruby the developer uses a Gemfile, in Python a requirements.txt, in Node.js a package.json, in Java a pom.xml, and so on.

Developers don't need to make many changes to an application in order to run it on Heroku. One requirement is informing the platform as to which parts of the application are runnable. This is done in a Procfile, a text file that accompanies the source code.Each line of the Procfile declares a process type — a named command that can be executed against the built application.

3.1.4.3 Deploying application

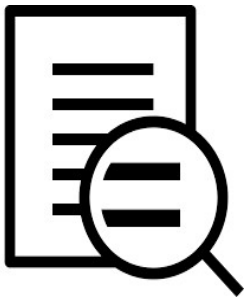
Application development on Heroku is primarily done through git. The application gets a new git remote typically named as Heroku along with its local git repository where the application was made. Hence to deploy heroku application is similar to using the git push command.

There are many other ways of deploying applications too. For example, developers can enable GitHub integration so that each new pull request is associated with its own new application, which enables all sorts of continuous integration scenarios. Dropbox Sync

lets developers deploy the contents of Dropbox folders to Heroku, or the Heroku API can be used to build and release apps.

Deployment then, is about moving the application from a local system to Heroku.

3.1.5 OCR (Optical Character Recognition)



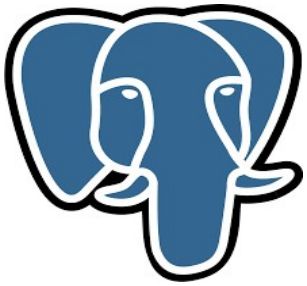
Optical character recognition (also optical character reader, OCR) is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast).

In OCR processing, the scanned-in image or bitmap is analyzed for light and dark areas in order to identify each alphabetic letter or numeric digit. When a character is recognized, it is converted into an ASCII code. Special circuit boards and computer chips designed expressly for OCR are used to speed up the recognition process.

OCR is being used by libraries to digitize and preserve their holdings. OCR is also used to process checks and credit card slips and sort the mail. Billions of magazines and letters are sorted every day by OCR machines, considerably speeding up mail delivery.

Fig 3.6. Extraction of vehicle number plate

3.1.6 Postgre SQL



PostgreSQL, often simply Postgres, is an object-relational database management system (ORDBMS) with an emphasis on extensibility and standards compliance. It can handle workloads ranging from small single-machine applications to large Internet-facing applications (or for data warehousing) with many concurrent users on macOS Server PostgreSQL is the default database;and it is also available for Microsoft Windows and Linux (supplied in most distributions).

PostgreSQL is ACID-compliant and transactional. PostgreSQL has updatable views and materialized views, triggers, foreign keys supports functions and stored procedures, and other expandability.

PostgreSQL is developed by the PostgreSQL Global Development Group, a diverse group of many companies and individual contributors. It is free and open-source, released under the terms of the PostgreSQL License, a permissive software license

PostgreSQL includes built-in binary replication based on shipping the changes to replica nodes asynchronously, with the ability to run read-only queries against these replicated nodes. This allows splitting read traffic among multiple nodes efficiently. Earlier replication software that allowed similar read scaling normally relied on adding replication triggers to the master, introducing additional load onto it.

PostgreSQL also includes built-in synchronous replication that ensures that, for each write transaction, the master waits until at least one replica node has written the data to its transaction log. Unlike other database systems, the durability of a transaction (whether it is asynchronous or synchronous) can be specified per-database, per-user, per-session or even per-transaction. This can be useful for work loads that do not require such guarantees, and may not be wanted for all data as it will have some negative effect on

performance due to the requirement of the confirmation of the transaction reaching the synchronous standby.

There can be a mixture of synchronous and asynchronous standby servers. A list of synchronous standby servers can be specified in the configuration which determines which servers are candidates for synchronous replication. The first in the list which is currently connected and actively streaming is the one that will be used as the current synchronous server. When this fails, it falls to the next in line.

3.2 Detailed Methodolgy

The problem for which we are trying to search a solution is to minimize the two-wheeler road accidents for which the rider must be wearing the helmet. The approach we followed is something like this:

1. Collecting positive and negative images for training our system to differentiate between two wheelers and other vehicles running on roads.
2. Similarly, collecting the positive sample images of two wheeler riders i.e. the images that comprise of people not wearing the helmet. These kind of images our positive for us because we have to detect the riders not wearing the helmet.
3. Now once the we have made th3e decision that the rider is wearing the helmet or not then we will let the system proceed to next image if the rider is wearing the helmet else the detected image will be again processed for number plate detection.
4. Now collecting positive and negative images again for in the same fashion for the vehicle's registration plate detection.
5. Once the registration pate is detected we need to proceed for registration number extraction by passing the detected registration plate to the OCR for text extraction.

The above stated process can be represented with the help of following flowchart:

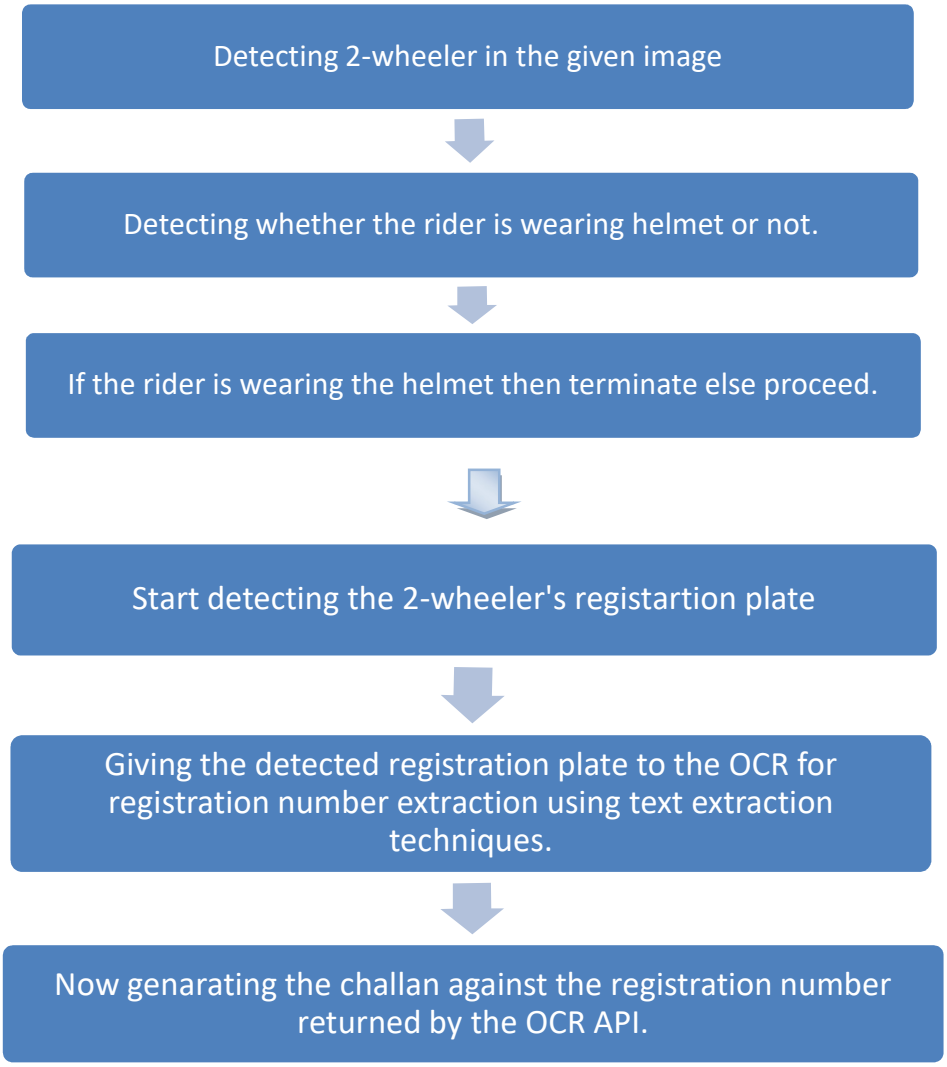


Fig. 3.7. Flowchart of the project

3.2.1 HAAR Training

This part is the heart of the complete project. The accuracy and reliability of our project depends on this part. This part helps the machine to understand and learn what we are trying to do and what kind of pictures are needed in the output and what not.

For this training two kinds of pictures are required:

1. Positive
2. Negative

Positive images comprises of all those images which are to be detected at the output for example in the case of two-wheeler detection, the scooter, the bikes are positive images.

Fig. 3.8. Rider wearing helmet

Negative images comprises of all those images which our to be ignored while object detection for example in case of two-wheeler detection , the bicycle, the car, the trucks all comprises to be a part of negative side. Also negative images include the background images suppose the tress, the roads, the buildings in the sample images should also be ignored.

Fig. 3.9. Rider without helmet

Now after collecting these images we have to train the machine which comprises of some predefined steps that the OpenCV library has provided. For all positive images we have to create a file which consists of number of positive objects and their locations in the given image. Now for this OpenCv has provided GUI based tool which helps to create this file.

Now this file along with the negative images is provided to OpenCV tool for it's understanding and learning our requirements and correspondingly producing cascades as output. These output cascades are then converted to the XML file which is used in java program for helmet detection.

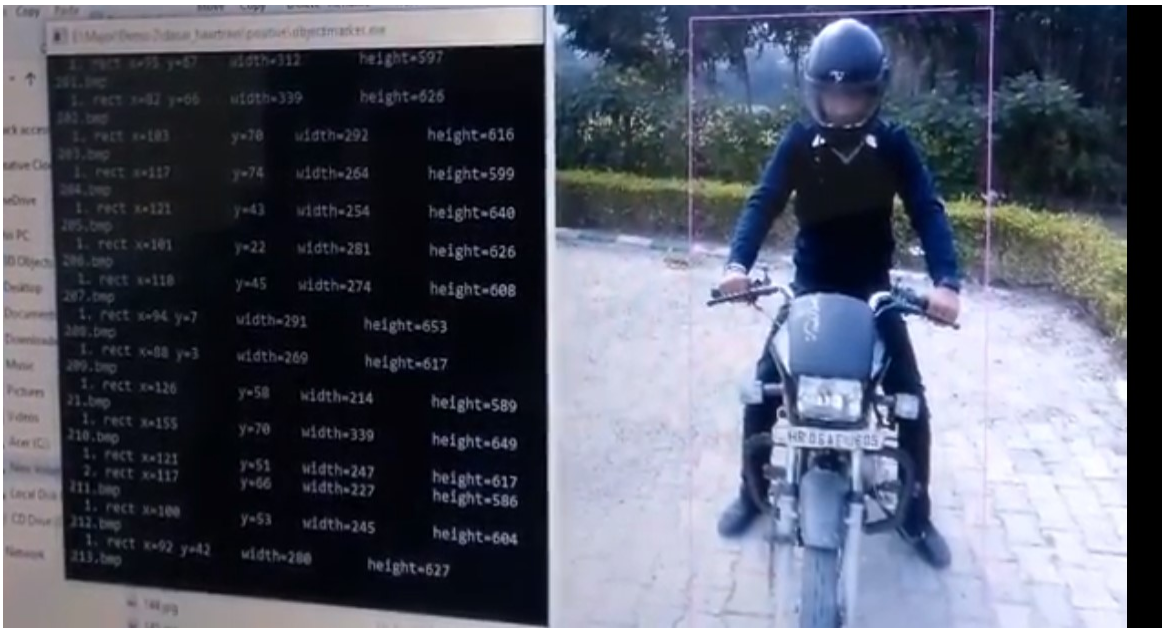


Fig. 3.10. Haar Training

3.2.2 Text Extraction using OCR

Now the detected number plate image is passed to OCR api for text extraction and the challan is generated against the detected registration number.

Fig. 3.11. Number plate detection for text extraction using OCR

3.2.3 Updating Database

Now after registration number is returned by the given API the next task is to report the particular registration number to our postgresql database for chalan entry and then sending the challan details along with the photo of the rider via SMS and email to the concerned owner of the vehicle

The postgresql database is provided by the heroku itself and is free upto a particular extent and then we can use it from the windows CLI by connecting the CLI with the heroku's server using our heroku credentials.

3.2.4 User Interface To Pay Fines Generated

Now the user can use the android app or the website which we will be designing to pay the fine or to check the challan details. The user can use any of the interface of his/her own choice to pay the fine and keep track of his/her vehicle's activites.

CHAPTER- 4

RESULTS

Currently we are on the verge of completing the image detection part using the OpenCV library. We have successfully detected the bike, the helmet and the registration plate of the vehicle.

The green marks in the images below represents that the person is notwearing the helmet and the number plate respectively.

Fig. 4.1 Successful detection of number plate and face

We have trained the machine for the same using HAAR training as specified in the Chapter-III. We trained the system using dataset of around 500 positive images for bike and helmet detection and around 150 images for registration plate detection and 600, 200 negative images resp. which comprises of all the objects which were to be ignored in the complete process as well as the background images.

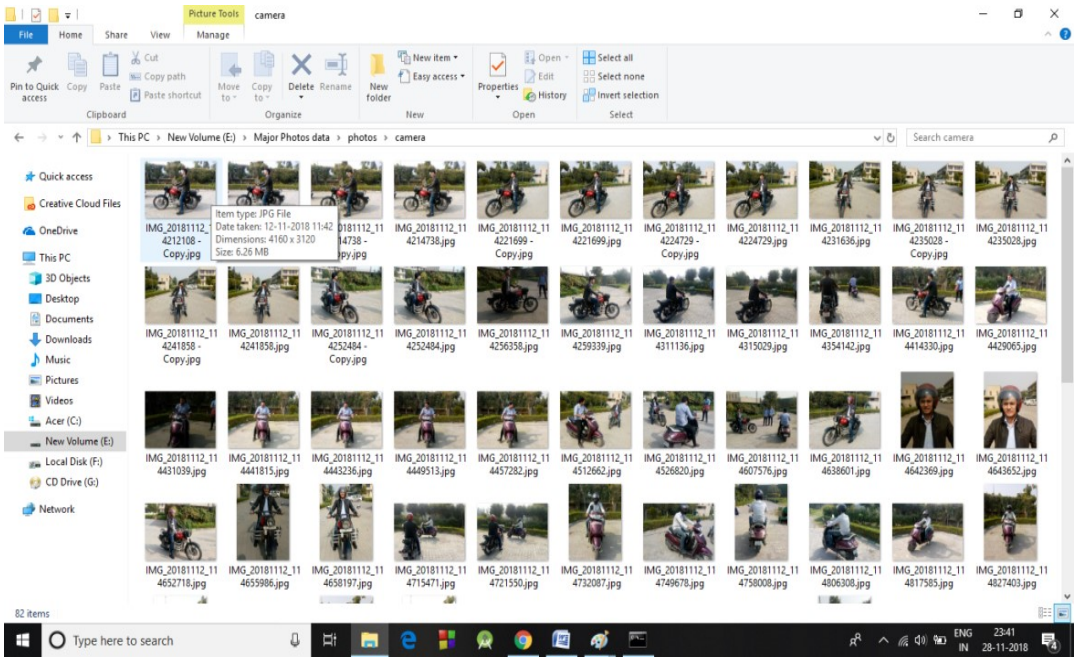


Fig. 4.2. Dataset Of Positive Images

As of now as discussed earlier we are done with the following task:

1. Detection of the two wheelers in the images provided as the inputs.
2. Crop images where two-wheelers are detected from the input.
3. Detection of the non-helmet wearing riders.
4. Detection of the vehicle registration number plates.
5. Reading the vehicle registration number plates using optical character recognition.
6. Sending the data of the violators to the database. In order to start the process of challan.

Most important thing in Machine Learning is to have a good amount of data in the dataset. By our experience of work during the project we acknowledged that no matter how powerful and efficient your code is but if you don't have a dataset of minimum 1500 images for one property detection, Machine Learning programs will not give the desired results.

CHAPTER - 5

CONCLUSION AND FUTURE SCOPE

The target of detecting the two-wheeler riders has been accomplished. The system designed by us is able to detect whether the rider is wearing the helmet or not and if the rider is not wearing the helmet the system proceeds to detect the number plate and the image of this number plate will be passed to the OCR for text extraction which would be the vehicle number and we will generate the challan for the respective vehicle.

But if we consider the real life scenario then we have some special cases which includes the following:

1. People wearing different kinds of helmet which are not safe and hence should be considered as violation of traffic rule.
2. People wearing different kinds of caps which should also be considered as violation.
3. People belonging to sikh community should be exempting from this traffic rule.

All these points should be kept in mind to use the project in real life in near future. If we consider these special cases in our project then the project is completely ready for commercial use and would be one of the biggest achievement in the field of automation.

REFERENCES

[1]Silva, R., Aires, K. and Veras, R. (2014). Helmet Detection on Motorcyclists Using Image Descriptors and Classifiers. *2014 27th SIBGRAPI Conference on Graphics, Patterns and Images*.

[2] Dahiya, K., Singh, D. and Mohan, C. (2016). Automatic detection of bike-riders without helmet using surveillance videos in real-time. *2016 International Joint Conference on Neural Networks (IJCNN)*.

[3] Silva, R., Aires, K., Santos, T., Abdala, K., Veras, R. and Soares, A. (2013). Automatic detection of motorcyclists without helmet. *2013 XXXIX Latin American Computing Conference (CLEI)*.

[4].https://docs.opencv.org/3.4.3/d7/d8b/tutorial_py_face_detection.html

[5]https://opencv-java-tutorials.readthedocs.io/en/latest

[6]https://www.tutorialspoint.com/dip/optical_character_recognition

[7]https://www.tutorialspoint.com/java/

[8]https://en.wikipedia.org/wiki/Android_(operating_system)

[9]https://en.wikipedia.org/wiki/PostgreSQL

[10]https://en.wikipedia.org/wiki/Heroku

[11]http://www.vogella.com/tutorials/JavaWebTerminology/article

[12]https://www.journaldev.com/1854/java-web-application-tutorial-for-beginners

[13]https://www.researchgate.net/publication/301585955_Automatic_Detection_of_Bike-riders_without_Helmet_using_Surveillance_Videos_in_Real-time

[14]http://www.ijettjournal.org/2016/volume-35/number-5/IJETT-V35P241.pdf

[15]https://arxiv.org/pdf/1802.00264.pdf

APPENDIX

Code for helmet and number plate detection

```
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.objdetect.CascadeClassifier;

class HelmetDetection
{
   public static void main(String[] args)
   {
System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
CascadeClassifierhelmetDetector = new CascadeClassifier();
helmetDetector.load("helmet.xml");

      // Input image
      Mat image = Imgcodecs.imread("E:\\Major\\final\\input\\1.jpg");

      // Detecting Helemt
MatOfRect helmet = new MatOfRect();
helmetDetector.detectMultiScale(image, helmet);

      // Creating a rectangular box showing faces detected

if(helmet.toArray().length!=0)
         {
```

```
        //String filename = "Ouput.jpg";
//Imgcodecs.imwrite("E:\\Major\\final\\output\\"+filename, image);

CascadeClassifiernumberPlate = new CascadeClassifier();
numberPlate.load("numberplate.xml");

MatnumberPlateImage = Imgcodecs.imread("E:\\Major\\final\\output\\Output.jpg");
MatOfRect number=new MatOfRect();
numberPlate.detectMultiScale(image, number);

for (Rectrect :number.toArray())
{
System.out.println("number plate");
        Imgproc.rectangle(image, new Point(rect.x, rect.y),
new Point(rect.x + rect.width, rect.y + rect.height),
        new Scalar(0, 255, 0));
}
for (Rectrect :helmet.toArray())
{
        System.out.println("No Helmet");
        Imgproc.rectangle(image, new Point(rect.x, rect.y),
        new Point(rect.x + rect.width, rect.y + rect.height),
        new Scalar(0, 255, 0));
}
        String filename="output.jpg";
Imgcodecs.imwrite("E:\\Major\\final\\output\\"+filename, image);

}
}
```