VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI-590018, KARNATAKA

A PROJECT REPORT(15CS85) ON

"Human Activity Recognition "

Submitted in partial fulfilment of the requirement

for the award of the degree of

Bachelor of Engineering

In

Computer Science & Engineering

For the academic year 2019-20

Submitted by

USN	Name
1CR16CS021	ANKIT KESHRI
1CR16CS006	ABHISHEK KUMAR
1CR16CS005	ABHISHEK DASGUPTA
1CR16IS120	VAIBHAV SANKRIT

Under the guidance of

Mrs. VIVIA JOHN

ASSISTANT PROFESSOR, DEPARTMENT OF CSE, CMRIT BANGALORE

Department of Computer Science & Engineering

CMR Institute of Technology, Bengaluru – 560 037

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to Certify that the dissertation work **"Human Activity Recognition using Machine Learning"** carried out by **Mr. Ankit Keshri, Mr. Abhishek Kumar, Mr. Vaibhav Sankrit Gupta, Mr. Abhishek Das Gupta** having **USN: 1CR16CS021, 1CR16CS006, 1CR16IS120, 1CR16CS005** respectively are bonafide students of **CMRIT** in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science & Engineering** of the **Visvesvaraya Technological University, Belagavi,** during the academic year **2019-20**. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said degree.

Signature of Guide	Signature of HOD	Signature of Principal
Mrs. Mrs Vivia Jhon	**Dr. Prem Kumar**	**Dr. Sanjay Jain**
Assistant Professor	**Head of the Department**	**Principal**
Dept. of CSE	**Dept. of CSE,**	**CMRIT,**
CMRIT, Bengaluru	**CMRIT, Bengaluru**	**Bengaluru**

DECLARATION

We, the students of Computer Science and Engineering, CMR Institute of Technology, Bangalore declare that the work entitled "**Human Activity Recognition using Machine Learning**" has been successfully completed under the guidance of Mrs Vivia John., Assistant Professor, Computer Science and Engineering Department, CMR Institute of technology, Bangalore. This dissertation work is submitted in partial fulfillment of the requirements for the award of Degree of Bachelor of Engineering in Computer Science and Engineering during the academic year 2019 - 2020. Further the matter embodied in the project report has not been submitted previously by anybody for the award of any degree or diploma to any university.

Place: Bangalore

Date: 06-Jun-2020

Team members:

ANKIT KESHRI (1CR16CS021)

ABHISHEK KUMAR (1CR16CS006)

VAIBHAV SANKRIT(1CR16IS120)

ABHISHEK DAS GUPTA (1CR16CS005)

ABSTRACT

The goal of our project is to detect the movement and actions of a person using image detection taken from a website or the local computer itself. In our project , the image is treated as an object. We are mainly focusing on four poses of human i.e sitting , sleeping ,standing and bending via the image.

Human activity recognition (HAR) aims to recognize activities from a series of observations on the actions of subjects and the environmental conditions. The vision-based HAR research is the basis of many applications including video surveillance, health care, and human-computer interaction (HCI).

Our project also focuses on the camera inbuilt in our laptop to work as a tracking device or equipment where the movement of the human is tracked as soon as he changes its position which makes our project more advanced and futuristic.

Finally, we investigate the directions for future research where its application will be beneficial and deployable .

ACKNOWLEDGEMENT

We take this opportunity to express our sincere gratitude and respect to **CMR Institute of Technology, Bengaluru** for providing us a platform to pursue our studies and carry out our final year project.

We take great pleasure in expressing our deep sense of gratitude to **Dr. Sanjay Jain,** Principal, CMRIT, Bangalore for his constant encouragement.

We would like to thank **Dr. Prem Kumar ,** Professor and Head, Department of Computer Science &Engineering, CMRIT, Bangalore, who has been a constant support and encouragement throughout the course of this project.

We express our sincere gratitude and we are greatly indebted to **Mrs Kavita , Assistant Professor,** Department of Computer Science & Engineering, CMRIT, Bangalore, for her invaluable co-operation and guidance at each point in the project without whom quick progression in our project was not possible.

We are also deeply thankful to our project guide **Mrs. Vivia John , Assistant Professor,** Department of Computer Science & Engineering, CMRIT, Bangalore, for critically evaluating each step in the development of this project and providing valuable guidance through our mistakes.

We also extend our thanks to all the faculty of Computer Science & Engineering who directly or indirectly encouraged us.

Finally, we would like to thank our parents and friends for all their moral support they have given us during the completion of this work.

TABLE OF CONTENTS Page No

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In today ‘s world, the computers have become an important aspect of life and are used in various fields however, the systems and methods that we use to interact with computers are outdated and have various issues, which we will discuss a little later in this paper. Hence, a very new field trying to overcome these issues has emerged namely HUMAN COMPUTER INTERACTIONS (HCI). Although, computers have made numerous advancement in both fields of Software and Hardware, Still the basic way in which Humans interact with computers remains the same , using basic pointing device (mouse) and Keyboard or advanced Voice Recognition System, or maybe Natural Language processing in really advanced cases to make this communication more human and easy for us. Our proposed project is activity recognition of humans based on its pose and movements.

The proposed system uses images as objects where the images are broken into 2.5D coordinate axes and detection is based on the learning of the axes frames.

Human activities have an inherent hierarchical structure that indicates the different levels of it, which can be considered as a three-level categorization. First, for the bottom level, there is an atomic element and these action primitives constitute more complex human activities. After the action primitive level, the action/activity comes as the second level. Finally, the complex interactions form the top level, which refers to the human activities that involve more than two persons and objects. In this paper, we follow this three-level categorization namely action primitives, actions/activities, and interactions. This three-level categorization varies a little from previous surveys and maintains a consistent theme. Action primitives are those atomic actions at the limb level, such as “stretching the left arm,” and “raising the right leg.” Atomic actions are performed by a specific part of the human body, such as the hands, arms, or upper body part . With the upgrades of camera devices, especially the launch of RGBD cameras in the year 2010, depth image-based representations have been a new research topic and have drawn growing concern years.

1.1 Digital Image Processing

Image processing is reckon as one of the most rapidly involving fields of the software industry with growing applications in all areas of work. It holds the possibility of developing the ultimate machine in the future, which would be able to perform the visual function of living beings. As such, it forms the basis of all kinds of visual automation.

1.2 Activity Detection and Recognition

A Human activity Recognition System recognizes the Shapes and or orientation depending on implementation to task the system into performing some job. Movement is a form of nonverbal information. A person can make numerous movements at a time. As humans through vision perceive human gestures and for computers we need an image and a camera, it is a subject of great interest for computer vision researchers such as performing an action based on activity of the person.

1.2.1 Detection

Activity detection is related to the position of a human at a given time in a stiff image or sequence of images i.e. moving images. In case of a moving sequence, it can be followed by tracking of the movement in the scene but this is more relevant to the applications such as sign language. The underlying concept of activity detection is that human eyes can detect objects, which machines cannot, with that much accuracy as that of humans. From a machine point of view it is just like a man fumble with his senses to find an object.

The factors, which make the activity detection task difficult to solve are:

Variations in image plane and pose

The human pose in the image varies due to its changing position whether it be sitting , standing ,bending or sleeping .The rotation can be both in and out of the plane.

Lighting Condition and Background

As shown in Figure 1.1 the accuracy of the image processed can vary depending on the lighting condition in which the image was uploaded.

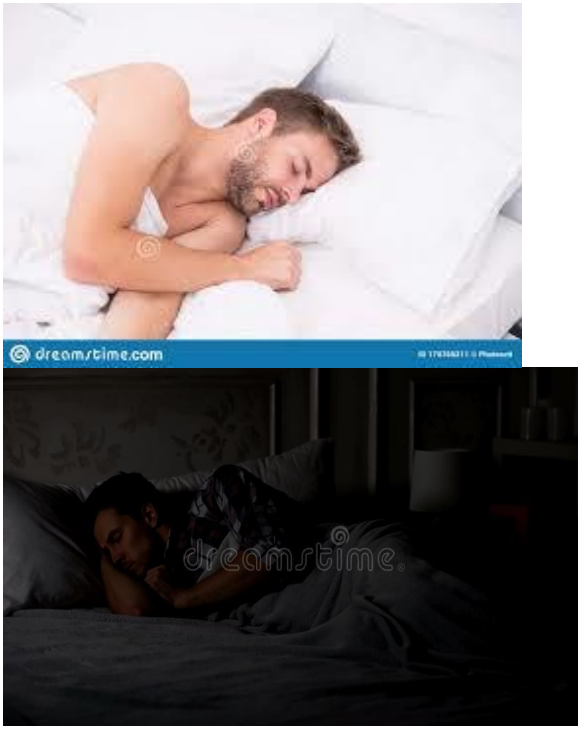


Figure 1.1: Lighting Condition and Background

1.2.2 Recognition

Human activity detection and recognition has been a significant subject in the field of computer vision and image processing in the past 30 years. There have been considerable achievements and numerous approaches developed in this field.

Movement Recognition is a topic in computer science and language technology with the goal of interpreting human actions via mathematical algorithms using images and camera samples. However, the identification and recognition of posture, gait, proxemics, and human behaviours is also the subject of gesture recognition techniques. However, the typical approach of a recognition system has been shown in the below figure:

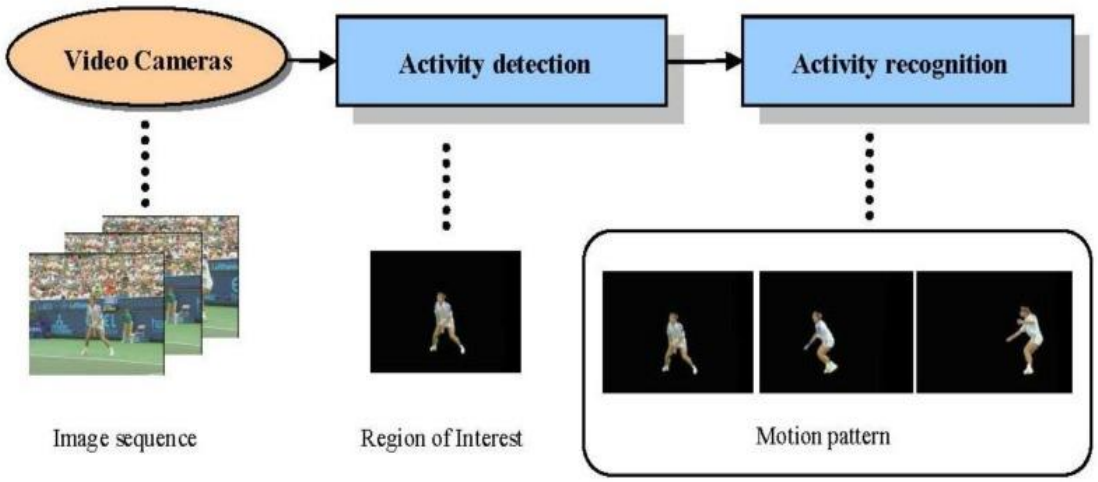


Figure 1.2: Hand Gesture Recognition Flow Chart

1.3 Objectives

The objectives of the project are:

1) Study and apply the needed tools namely:
   a) Image downloaded from computer or locally saved.
   b) Flask Deployed Server and python 3.6.8 Community
   c) Algorithms for computer vision and machine learning.

2) Develop a front-end website to upload images to process.
3) Test the computer application and website running
4) Document the result of the project.

1.4 Scope

The scope of our project is to develop a real time activity recognition system which ultimately controls the image with a jpg extension and the camera samples of real-time webcam method. During the project, four gestures were chosen to represent four navigational commands that are sitting, standing, bending and sleeping. A simple computer vision application was written for the

detection and recognition of the four gestures and their translation into the corresponding commands for the actions and tracking. . Thereafter, the program was tested on a webcam with actual movement of the person in real-time and the results were observed.

CHAPTER 2

LITERATURE SURVEY

2.1 Computer Vision and Digital Image Processing

The sense of sight is arguably the most important of man's five senses. It provides a huge amount of information about the world that is rich in detail and delivered at the speed of light. However, human vision is not without its limitations, both physical and psychological. Through digital imaging technology and computers, man has transcending many visual limitations. He can see into far galaxies, the microscopic world, the sub-atomic world, and even “observe” infrared, x-ray, ultraviolet and other spectra for medical diagnosis, meteorology, surveillance, and military uses, all with great success.

While computers have been central to this success, for the most part man is the sole interpreter of all the digital data. For a long time, the central question has been

whether computers can be design to analyse and acquire information from images autonomously in the same natural way humans can.

The main difficulty for computer vision as a relatively young discipline is the current lack of a final scientific paradigm or model for human intelligence and human vision itself on which to build a infrastructure for computer or machine learning. The use of images has an obvious drawback. Humans perceive the world in 3D, but current visual sensors like cameras capture the world in 2D images. The result is the natural loss of a good deal of information in the captured images. Without a proper paradigm to explain the mystery of human vision and perception, the recovery of lost information (reconstruction of the world) from 2D images represents a difficult

hurdle for machine vision. However, despite this limitation, computer vision has progressed, riding mainly on the remarkable advancement of decade old digital image processing techniques, using the science and methods contributed by other disciplines such as optics, neurobiology, psychology, physics, mathematics, electronics, computer science, artificial intelligence and others.

Computer vision techniques and digital image processing methods both draw the proverbial water from the same pool, which is the digital image, and therefore necessarily overlap. Image processing takes a digital image and subjects it to processes, such as noise reduction, detail enhancement, or filtering, for producing another desired image as the result. For example, the blurred image of a car registration plate might be enhanced by imaging techniques to produce a clear photo of the same so the police might identify the owner of the car. On the other hand, computer vision takes a digital image and subjects it to the same digital imaging techniques but for the purpose of analysing and understanding what the image depicts. For example, the image of a building can be fed to a computer and thereafter be identified by the computer as a residential house, a stadium, high-rise office tower, shopping mall, or a farm barn.

Russell and Norvig identified three broad approaches used in computer vision to distil useful information from the raw data provided by images. The first is the feature extraction approach, which focuses on simple computations applied directly to digital images to measure some useable characteristic, such as size. This relies on generally known image processing algorithms for noise reduction, filtering, object detection, edge detection, texture analysis, computation of optical flow, and segmentation, which techniques are commonly used to pre-process images for subsequent image analysis. This is also considered as an "uninformed" approach.

The second is the recognition approach, where the focus is on distinguishing and labelling objects based on knowledge of characteristics that sets of similar objects have in common, such as shape or appearance or patterns of elements, sufficient to form classes. Here computer vision uses the techniques of artificial intelligence in knowledge representation to enable a "classifier" to match classes to objects based on the pattern of their features or structural description. A classifier has to "learn" the patterns by being fed a training set of objects, their classes, achieving the goal of minimizing mistakes, and maximizing successes through a systematic process of improvement. There are many techniques in artificial intelligence that can be used for object or pattern recognition, including statistical pattern recognition, neural nets, genetic algorithms and fuzzy systems.

The third is the reconstruction approach, where the focus is on building a geometric model of the world suggested by the image or images and which is used as a basis for action. This corresponds to the stage of image understanding, which represents the highest and most complex level of computer vision processing. Here the emphasis is on enabling the computer vision system to construct internal models based on the data supplied by the images and to discard or update these internal models as they are verified against the real world or some other criteria. If the internal model is consistent with the real world, then image understanding takes place. Thus, image understanding requires the construction, manipulation and

control of models and now relies heavily upon the science and technology of artificial intelligence.

2.2 Open CV in Image Processing

OpenCv is a widely used tool in computer vision. It is a computer vision library for real-time applications, written in C and C++, which works with the Windows, Linux and Mac platforms. It is freely available as open source software from http://sourceforge.net/projects/opencvlibrary/.

OpenCv was start by Gary Brodsky at Intel in 1999 to encourage computer vision research and commercial applications and, side-by-side with these, promote the use of ever-faster processors from Intel. OpenCV contains optimised code for a basic computer vision infrastructure so developers do not have to re-invent the proverbial wheel. Brodsky and Koehler provide the basic tutorial documentation. According to its website, OpenCV has been downloaded more than two million times and has a user group of more than 40,000 members. This attests to its popularity.

A digital image is generally understood as a discrete number of light intensities captured by a device such as a camera and organized into a two-dimensional matrix of picture elements or pixels, each of which may be represented by number and all of which may be stored in a particular file format (such as jpg or gif). OpenCV goes beyond representing an image as an array of pixels. It represents an image as a data structure called an IplImage that makes immediately accessible useful image data or fields, such as:

- width – an integer showing the width of the image in pixels
- height – an integer showing the height of the image in pixels
- imageData – a pointer to an array of pixel values
- nChannels – an integer showing the number of colours per pixel
- depth – an integer showing the number of bits per pixel
- widthStep – an integer showing the number of bytes per image row
- geSize – an integer showing the size of in bytes
- roi – a pointer to a structure that defines a region of interest within the image.

OpenCV has a module containing basic image processing and computer vision algorithms. These include:

- smoothing (blurring) functions to reduce noise,
- dilation and erosion functions for isolation of individual elements,
- flood fill functions to isolate certain portions of the image for further

- processing,
- filter functions, including Sobel, Laplace and Canny for edge detection,
- Hough transform functions for finding lines and circles,
- Affine transform functions to stretch, shrink, warp and rotate images,
- Integral image function for summing sub regions (computing Haar wavelets),
- Histogram equalization function for uniform distribution of intensity values,
- Contour functions to connect edges into curves,
- Bounding boxes, circles and ellipses,
- Moments functions to compute Hu's moment invariants,
- Optical flow functions (Lucas-Kanade method),
- Motion tracking functions (Kalman filters), and
- Face detection/ Haar classifier.

OpenCV also has an ML (machine learning) module containing well-known statistical classifiers and clustering tools. These include:

- Normal/ naïve Bayes classifier,
- Decision trees classifier,
- Boosting group of classifiers,
- Neural networks algorithm, and

2.3 Pattern Recognition and Classifiers

In computer, vision a physical object maps to a particular segmented region in the image from which object descriptors or features may be derive. A feature is any characteristic of an image, or any region within it, that can be measure. Objects with common features may be group into classes, where the combination of features may be considered a pattern. Object recognition may be understood to be the assignment of classes to objects based on their respective patterns. The program that does this assignment is called a classifier.

The general steps in pattern recognition may be summarized in Figure below:

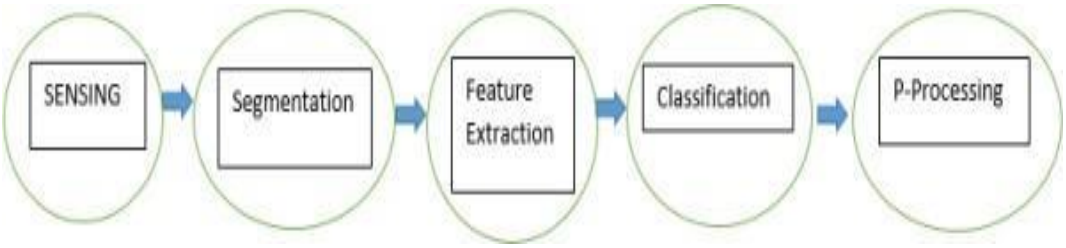


Fig 2.1 Flow Chart

The most important step is the design of the formal descriptors because choices have to be made on which characteristics, quantitative or qualitative, would best suit the target object and in turn determines the success of the classifier.

In statistical pattern recognition, quantitative descriptions called features are used. The set of features constitutes the pattern vector or feature vector, and the set of all possible patterns for the object form the pattern space X (also known as feature space). Quantitatively, similar objects in each class will be located near each other in the feature space forming clusters, which may ideally be separated from dissimilar objects by lines or curves called discrimination functions.

Determining the most suitable discrimination function or discriminant to use is part of classifier design.

A statistical classifier accepts n features as inputs and gives 1 output, which is the classification or decision about the class of the object. The relationship between the inputs and the output is a decision rule, which is a function that puts in one space or subset those feature vectors that are associated with a particular output. The decision rule is based on the particular discrimination function used for separating the subsets from each other.

The ability of a classifier to classify objects based on its decision rule may be understood as classifier learning, and the set of the feature vectors (objects) inputs and corresponding outputs of classifications (both positive and negative results) is called the training set. It is expected that a well-designed classifier should get 100% correct answers on its training set. A large training set is generally desirable to optimize the training of the classifier, so that it may be tested on objects it has not encountered before, which constitutes its test set. If

the classifier does not perform well on the test set, modifications to the design of the recognition system may be needed.

2.4 Moment Invariants in Image Processing

As mentioned previously, feature extraction is one approach used in computer vision. According to A.L.C. Baraka, feature extraction refers to the process of distilling a limited number of features that would be sufficient to describe a large set of data, such as the pixels in a digital image . The idea is to use the features as a unique representation of the image.

Since a digital image is a two-dimensional matrix of pixels values, region-based object descriptions are affected by geometric transformations, such as scaling, translation,

and rotation. For example, the numerical features describing the shape of a 2D object would change if the shape of the same object changes as seen from a different angle or perspective. However, to be useful in computer vision applications, object descriptions must be able to identify the same object irrespective of its position, orientation, or distortion.

One of the most popular quantitative object descriptors are moments. Hu first formulated the concept of statistical characteristics or moments that would be indifferent to geometric transformations in 1962. Moments are polynomials of increasing order that describe the shape of a statistical distribution . Its exponent indicates the order of a moment. The geometric moments of different orders represent different spatial characteristics of the image intensity distribution. A set of moments can thus form a global shape descriptor of an image.

Hu proposed that the following seven functions (called 2D moment invariants) were invariant to translation, scale variation, and rotation of an image:

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

A System Requirement Specification (SRS) is an organization's understanding of a customer or potential client's system requirements and dependencies at a particular point prior to any actual design or development work. The information gathered during the analysis is translated into a document that defines a set of requirements. It gives the brief description of the services that the system should provide and the constraints under which, the system should operate. Generally, SRS is a document that completely describes what the proposed software should do without describing how the software will do it. A two-way insurance policy assures that both the client

and the organization understand the other's requirements from that perspective at a given point in time.

SRS document itself states in precise and explicit language those functions and capabilities a software system (i.e., a software application, an ecommerce website and so on) must provide, as well as states any required constraints by which the system must abide. SRS also functions as a blueprint for completing a project with as little cost growth as possible. SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it.

Requirement is a condition or capability to which the system must conform. Requirement Management is a systematic approach towards eliciting, organizing and documenting the requirements of the system clearly along with the applicable attributes. The elusive difficulties of requirements are not always obvious and can come from any number of sources.

3.1 General Description

In this section of the presented thesis, the introduction of software product under consideration has been presented. It presents the basic characteristics and factors influencing the software product or system model and its requirements.

3.1.1 Product Perspective

In this project or research work, we have proposed a highly robust and efficient mechanism for image detection using human activity system. The proposed system has been emphasized on developing an efficient scheme that can accomplish activity

recognition without introducing any training related overheads. The proposed system has to take into consideration of geometrical shape of human pose and based on defined thresholds and real time parametric variation, the segmentation for human position is accomplished. Based on retrieved specific shape, certain application-oriented commands have to be generated. The predominant uniqueness of the proposed scheme is that it does not employ any kind of prior training and it is functional in real time without having any databases or training datasets. Unlike tradition approaches of images, datasets-based recognition system; this approach achieves human activity recognition in real time, and responds correspondingly. This developed mechanism neither introduce any computational complexity nor does it cause any user interferences to achieve tracing of human gesture.

3.1.2 User Characteristics

The user should have at least a basic knowledge of windows and web browsers, such as install software like Python Pycharm, Python 3.6.8 etc. and executing a program, and the ability to follow on screen instructions. The user will not need any technical expertise in order to use this program.

3.2 Functional Requirement

- The image used will be uploaded into the website from where result will be displayed
- with the machine accuracy.
- The software will be able to produce multiple frames and display the image in the RGB colour space.
- The software will be able to detect the contours of the detected positions.
- The software, which act as an intermediate in passing these, processed image in order to control the image sent.

3.3 Non-Functional Requirement

- **Usability:** The user is facilitated with the control section for the entire process in which they can arrange the position of hand at the centre of ROI under consideration, the variation of palm position and respective command generation etc. can be effectively facilitated by mean of user interface. The implementation and calibration of camera and its resolution can also be done as per quality and preciseness requirement.

  The frame size, flow rate and its command variation with respect to threshold developed and colour component of hand colour, can be easily calibrated by means of certain defined thresholds.

- **Security and support:** Application will be permissible to be used only in secure network so there is less feasibility of insecurity over the functionality of the application. On the other hand, the system functions in a real time application scenario, therefore the camera, colour and platform compatibility is must in this case. IN case of command transfer using certain connected devices or wireless communication, the proper port assignment would also be a predominant factor to be consider.

- **Maintainability:** The installation and operation manual of the project will be provided to the user.

- **Extensibility:** The project work is also open for any future modification and hence the work could be define as the one of the extensible work.

3.5 External Interface Requirement

An interface description for short is a specification used for describing a software component's interface. IDLs are commonly used in remote procedure call software. In these issues, the machines on moreover last part of the "link" might be utilizing

Dissimilar Interface Description recommends a bridge between the two diverse systems. These descriptions are classified into following types:

- **User Interface:** The external or operating user is an individual who is interested to introduce a novel Algorithm for shape based hand gesture recognition in real time application scenario. The user interface would be like axis presenting real time movement of human hand and its relative position with respect to defined centroid or morphological thresholds.

- **Restoration with Text Removal Software Interface:** The Operating Systems can be any version of Windows, Linux, UNIX or Mac.

- **Hardware Interface:** In the execution of this project, the hardware interface used is a normal 32/64 bit operating system supported along with better integration with network interface card for better communication with other workstations. For better and precise outcome, a high definition camera with calibrated functioning with

CHAPTER 4

SYSTEM ANALYSIS

Analysis is the process of finding the best solution to the problem. System analysis is the process by which we learn about the existing problems, define objects and requirements and evaluates the solutions. It is the way of thinking about the organization

and the problem it involves, a set of technologies that helps in solving these problems. Feasibility study plays an important role in system analysis, which gives the target for design and development.

4.1 Feasibility Study

All systems are feasible when provided with unlimited resource and infinite time. But unfortunately, this condition does not prevail in practical world. So it is both necessary and prudent to evaluate the feasibility of the system at the earliest possible time. Months or years of effort, thousands of rupees and untold professional embarrassment can be averted if an ill- conceived system is recognized early in the definition phase. Feasibility & risk analysis are related in many ways. If project risk is great, the feasibility of producing quality software is reduced. In this case three key considerations involved in the feasibility analysis are:

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

4.1.1 Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

4.1.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

4.1.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcome, as he is the final user of the system.

4.2 Analysis

4.2.1 Performance Analysis

For the complete functionality of the project work, the project is run with the help of healthy networking environment. Performance analysis is done to find out whether the proposed system. It is essential that the process of performance analysis and definition must be conduct in parallel.

4.2.2 Technical Analysis

System is only beneficial only if it can be turn into information systems that will meet the organization's technical requirement. Simply stated this test of feasibility asks whether the system will work or not when developed & installed, whether there are implementation. Regarding all these issues in technical analysis there are several points to focus on:

Changes to bring in the system: All changes should be in positive direction, there would be increased level of efficiency and better customer service.

Required skills: Platforms & tools used in this project are widely used. Therefore, the skilled work force is readily available in the industry.

Acceptability: The structure of the system is kept feasible enough so that there should not be any problem from the user's point of view.

4.2.3 Economic Analysis

Economic analysis is perform to evaluate the development cost weighed against the ultimate income or benefits derived from the developed system. For running this system, we need not have any routers, which are highly economical. Therefore, the system is economically feasible enough.

CHAPTER 5

SYSTEM DESIGN

Design is a meaningful engineering representation of something that is to be built. It is the most crucial phase in the developments of a system. Software design is a process through which the requirements are translated into a representation of software. Design is a place where design is fostered in software Engineering. Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. Design is the perfect way to accurately translate a customer's requirement in the finished software product. Design creates a representation or model, provides details about software data structure, architecture,

interfaces and components that are necessary to implement a system. The logical system design arrived at because of systems analysis is converted into physical system design.

5.1 System Development methodology

System development method is a process through which a product will get completed or a product gets rid from any problem. Software development process is described as a number of phases, procedures and steps that gives the complete software. It follows series of steps, which are used for product progress. The development method followed in this project is waterfall model.

5.1.1 Model Phases

The waterfall model is a sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Requirement initiation, Analysis, Design, Implementation, Testing and maintenance.

Requirement Analysis: This phase is concerned about collection of requirements of the system. This process involves generating document and requirement review.

System Design: Keeping the requirements in mind the system specifications are translate in to a software representation. In this phase, the designer emphasizes on-algorithm, data structure, software architecture etc.

Coding: In this phase, programmer starts his coding in order to give a full sketch of product. In other words, system specifications are only convert in to machine-readable compute code.

Implementation: The implementation phase involves the actual coding or programming of the software. The output of this phase is typically the library, executables, user manuals and additional software documentation

Testing: In this phase, all programs (models) are integrated and tested to ensure that the complete system meets the software requirements. The testing is concerned with verification and validation.

Maintenance: The maintenance phase is the longest phase in which the software is updated to fulfil the changing customer need, adapt to accommodate change in the external environment, correct errors and oversights previously undetected in the testing phase,

5.1.2 Advantages of Waterfall Model

- Clear project objectives.
- Stable project requirements.
- Progress of system is measurable.
- Strict sign-off requirements.
- Helps you to be perfect.
- Logic of software development is clearly understood.
- Production of a formal specification.
- Better resource allocation.

- Improves quality. The emphasis on requirements and design before writing a single line of code ensures minimal wastage of time and effort and reduces the risk of schedule slippage.

-

- Less human resources required as once one phase is finished those people can start working on to the next phase

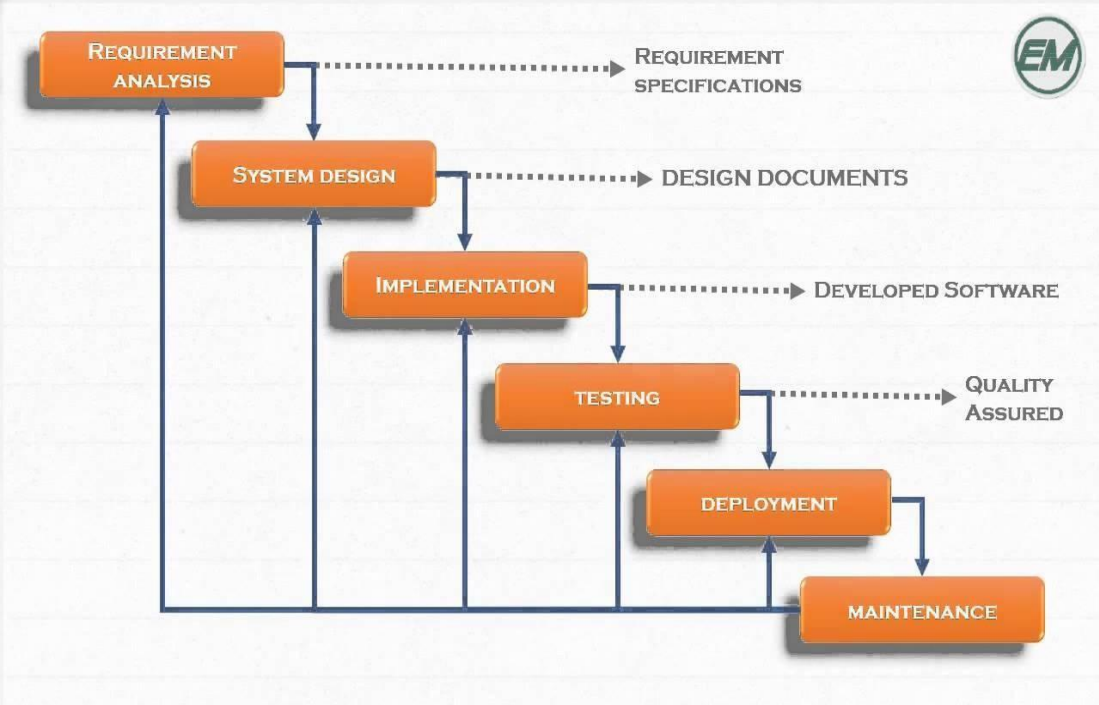


Fig 5.1 Waterfall Model

5.2 Design Using UML

Designing UML diagram specifies, how the process within the system communicates along with how the objects with in the process collaborate using both static as well as dynamic UML diagrams since in this ever-changing world of Object Oriented application development, it has been getting harder and harder to develop and manage high quality applications in reasonable amount of time. Because of this challenge and the need for a universal object modelling language

everyone could use, the Unified Modelling Language (UML) is the Information industries version of blue print. It is a method for describing the systems architecture in detail. Easier to build or maintains system, and to ensure that the system will hold up to the requirement changes.

5.3 Data Flow Diagram

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generate by the system.

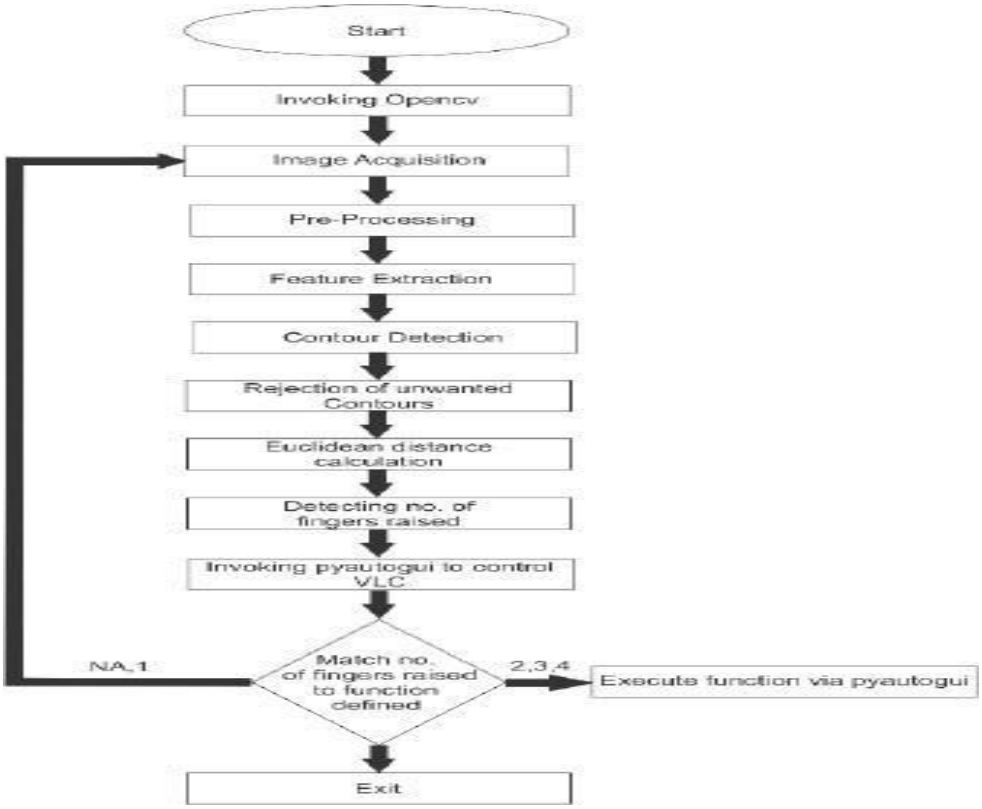


Fig 5.2 Data Flow Model

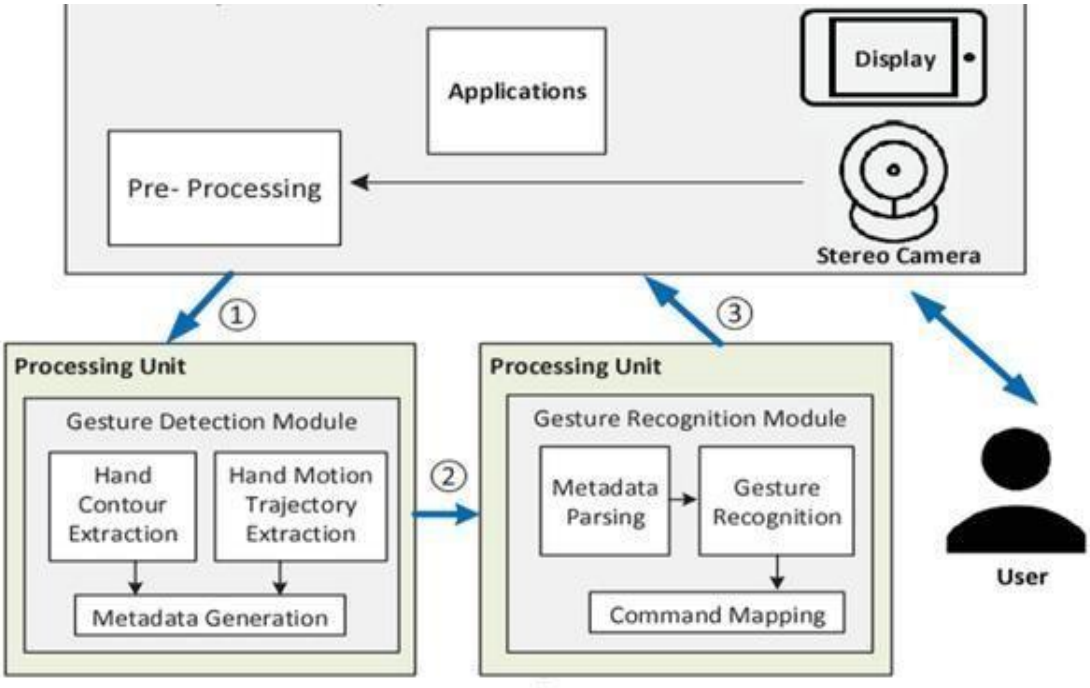
The data flow diagram essentially shows how the data control flows from one module to another. Unless the input filenames are correctly given the program cannot proceed to the next module. Once the user gives, the correct input filenames parsing is done individually for each file. The required information is taken in parsing and an adjacency matrix is generated for that. From the adjacency matrix, a lookup table is generated giving paths for blocks. In addition, the final sequence is computed with the

lookup table and the final required code is generated in an output file. In case of multiple file inputs, the code for each is generated and combined together.

5.4 COMPONENT DIAGRAM

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Fig 5.3 Component Diagram



The component diagram for the decentralized system ideally consists of different modules that are represented together via a common module for the user. The user is required to have the input files in the current folder where the application is being used.

It is interesting to note that all the sequence of activities that are taking place are via this module itself, i.e. the parsing and the process of computing the final sequence. The parsing redirects across the other modules until the final code is generated.

5.5 Use Case Diagram

A use case defines a goal-oriented set of interactions between external entities and the system under consideration. The external entities, which interact with the system, are its actors. A set of use cases describe the complete functionality of the system at a particular level of detail and the use case diagram can graphically denote it.

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

In software and systems engineering, a use case is a list of steps, typically defining interactions between a role (known in Unified Modelling Language (UML) as an "actor") and a system, to achieve a goal. The actor can be a human, an external system, or time.

In systems engineering, use cases are use at a higher level than within software engineering, often representing missions or stakeholder goals. The detailed requirements may then be capture in Systems Modelling Language (SysML) or as contractual statements.

The Sequence of activities that are carried out are the same as the other diagrams .Use case for this module indicates the users interaction with the system as a whole rather than individual modules .All the encryption mechanisms are carried out via the login page that redirects the user to the particular functionality that he or she wishes to implement.

5.6 Activity Diagram

An activity diagram shows the sequence of steps that make up a complex process. An activity is shown as a round box containing the name of the operation. An outgoing solid arrow attached to the end of the activity symbol indicates a transition triggered by the completion.

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams are intended to model both computational and organisational processes (i.e. workflows). Activity diagrams show the overall flow of control.

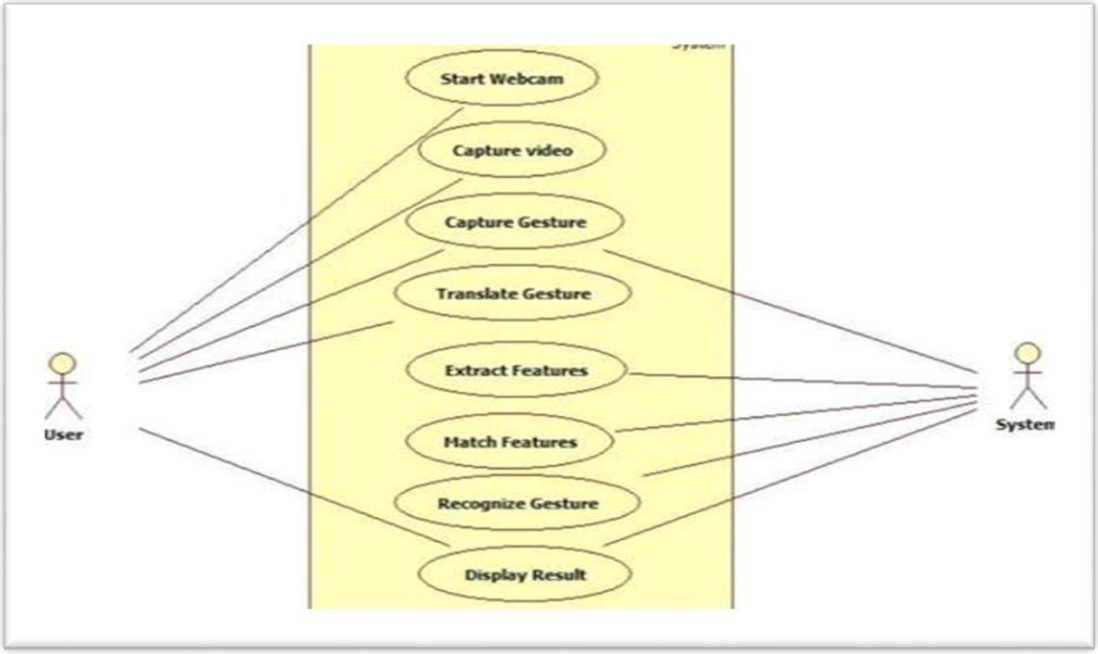


Fig 5.4 Use Case Diagram

Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

- rounded rectangles represent actions;
- diamonds represent decisions;

- bars represent the start (split) or end (join) of concurrent activities;
- a black circle represents the start (initial state) of the workflow;
- An encircled black circle represents the end (final state).

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behaviour of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only use for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part.

CHAPTER 6

PROPOSED SYSTEM

6.1 Data Obtaining

The initial move is to capture the image from camera and to define a region of Interest in the frame, it is important as the image can contain a lot of variables and these variables can result in unwanted results and the data that needs to be processed is reduced to a large extent. To capture the image a web-camera is used that continuously captures frames and is used to get the raw data for processing. The input picture we have here is uint8. The Procured image is RGB and must be process before i.e. pre-processed before the components are separated and acknowledgement is made.

6.2 Feature Engineering

Our preliminary analysis emphasizes on the dire need of new features. We transform the raw data to a new feature space, where the classification task is better defined. This is an extremely crucial step since the choice of our new features strongly influence the final

prediction . Human Activity Recognition: Feature generation To enhance the feature space, we calculated the pitch, roll and normal value for each of the accelerometer using the following equations,

$$pitch = \text{atan}\left(y \sqrt{x^2 + z^2} \right)$$

$$roll = \text{atan}\left(-x / z \right)$$

$$norm = \sqrt{x^2 + y^2 + z^2}$$

This increased the feature space from 12 attributes to 24 attributes. Segmentation For segmentation, our approach was to consider sliding windows [5], of various

sizes, to aggregate the data points within the window. This helps us capture the chronological variation between the data points. Performing this activity, mitigates the risk that the classifier itself isn't temporal in nature. Figure 6. Sliding Window For our analysis, we considered both overlapping and non-overlapping sliding windows. We found that the accuracy for prediction for non-overlapping windows is higher, however the confidence interval is also wide. Here, if the window size is n, then the new data set would be 1/n the original data set. This increases the risks of over-fitting, due to decrease in training points. Next, we took overlapping windows, wherein though the size of dataset still reduces, it is greater than 1/n. Also overlapping windows ensure that the transition of time is maintained and the data points are not independent of each other. We used mean and standard deviation as aggregation functions initially and performed segmentation on raw dataset. This new dataset was then tested against Naïve Bayes and Random Forest, the top two classifiers from our preliminary analysis. Figure 7. Non-overlapping windows vs. overlapping window In both the cases, we found the peak to be around window size of 13-14, i.e. 1.95s -2.10s. Further to confirm our findings, we tested the overlapping window for newly generated features. Again, Human Activity Recognition: Group C | 6 this time we considered the mean and variance for each attribute across the window. This time we only considered Random Forest, since it had given us the best results in previous case. Figure 8. Accuracy across different window sizes for Random Forest with new features Finally, we concluded that

window size of 14 gave us the best results. Next, we moved on to feature extraction. 7.3. Feature Extraction Having multiple accelerometer increases redundancy in the data being observed. Thus, as the first step of feature extraction we decided to test which accelerometers give us new information and are relevant as opposed to the redundant ones. For carry out this analysis, we took all the 48 new features, and grouped them by the accelerometer number. Then we performed the performance test on the exhaustive combination of sensors using Random Forest. Accuracy comparison for exhaustive combination set of sensors. From the Figure 9, it is clear that if we were to consider single accelerometer then sensor 1, the accelerometer placed on waist gives the best results. Also, we find that the best results are obtained using combination of sensor 1 and 3, i.e. ones placed on waist and right arm. All our further analysis is carried by considered just the features corresponding to sensor 1 and 3. In

our next step to reduce the feature space, we carried our principle component analysis as well as backward elimination of features.

Initially we performed backward feature elimination and found that the accuracy peaked at feature subset of size 11. The attributes in consideration at this point were, mean and variance of pitch and normal of sensor 1 and 3 and the variance of roll of both sensor but mean of roll of only sensor 1. To evaluate further, we decided to perform principle components analysis, to check if that yields better performance. We found that even with PCA, the optimal number of components considered would be 12.

6.3 Classification

Classification was a continuous ongoing process in our analysis. We performed 4-folds cross validation to ensure that we were not over-fitting. Instead of typical 10-cross validation, we took 4- folds by considering each subject as test case for one iteration and the remaining as training set. While performing cross validation to retain the temporal aspect of the data, we did not randomize our folds. In case of 10-folds, the accuracy of the classifier would increase, but that would be overfitting since a section of the test subject's data would already be there in the training set. Initially, we considered four classification methods for HAR analysis, however we kept on eliminating them if there was no significant improvement in prediction power, with increasing complexity of feature transformation. Human Activity Recognition:

the final comparison between the models generated by random forest for raw attributed and processed final 11 attributes. They show the error rate of the models as well as the variable importance plot. Model for these plots was trained using data from all the four subjects.

6.4 Model Used

RECURRENT NEUTRAL NETWORKS: RNN's are a type of artificial neural network designed to recognise patterns in a sequence of data, such as text, grammar, handwriting, spoken word or numeric time series data emanating from sensors, stock markets & government agencies.

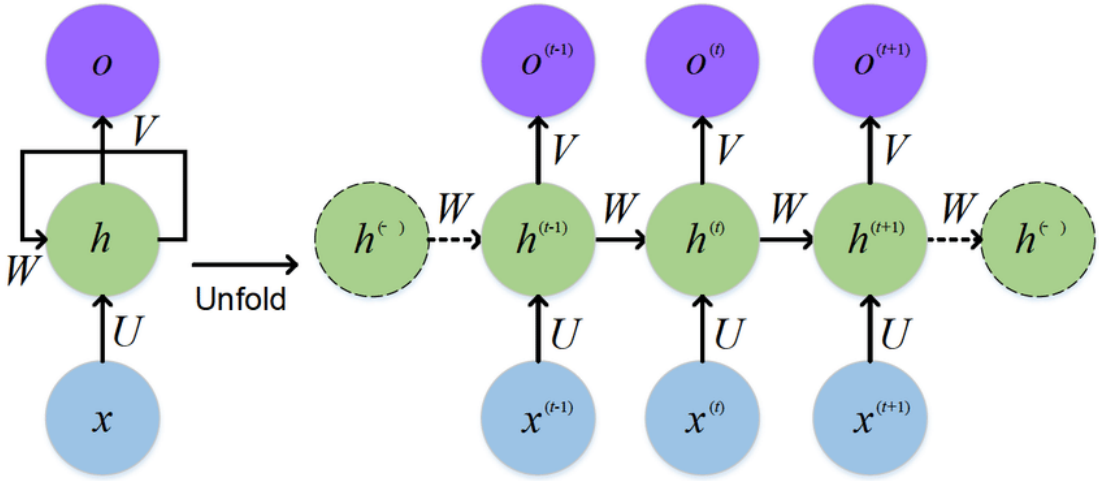
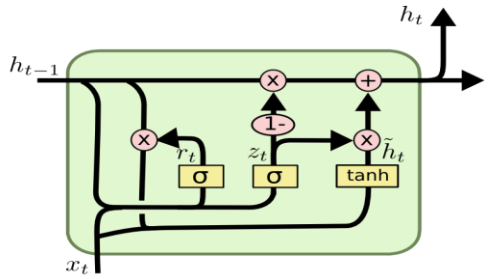


Fig 6.1 RNN Flow

LONG SHORT-TERM MEMORY NETWORKS: They are special kind of recurrent neutral networks. They are capable of learning and handling long term dependencies.

The compact forms of the equations for the forward pass of an LSTM unit with a forget gate are:



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Fig 6.2 LSTM EQUATIONS

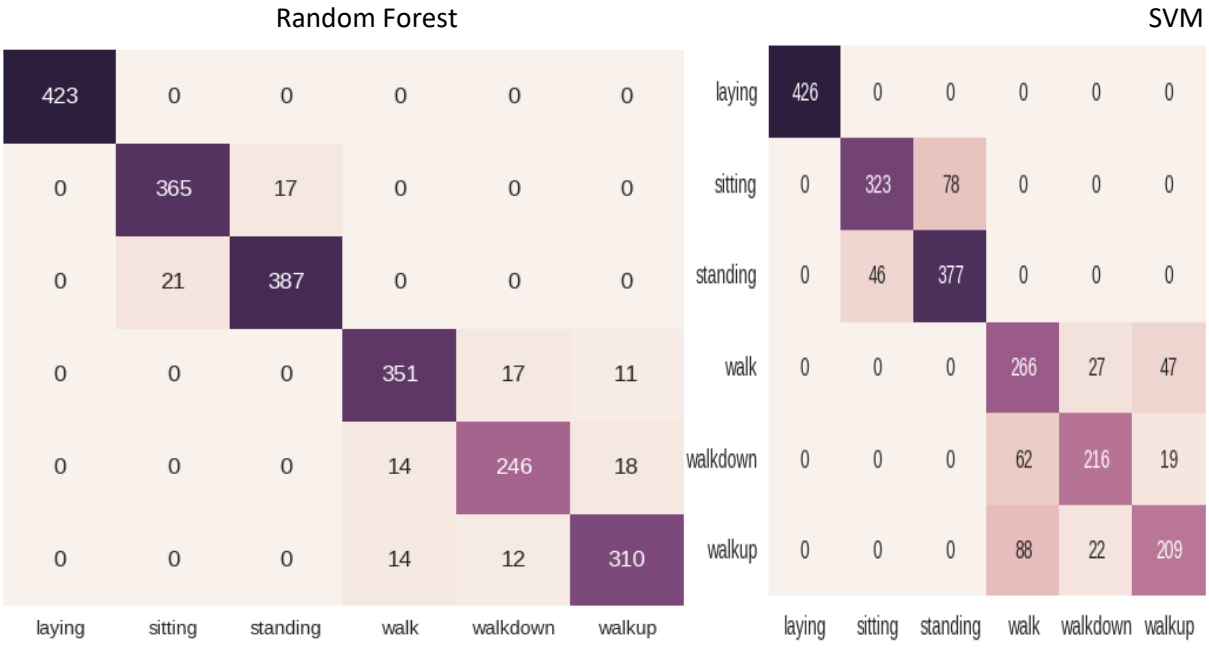
where the initial values are and the operator denotes the Hadamard product (element-wise product). The subscript indexes the time step.

Variables

- : input vector to the LSTM unit
- : forget gate's activation vector
- : input/update gate's activation vector
- : output gate's activation vector
- : hidden state vector also known as output vector of the LSTM unit
- : cell state vector
- : weight matrices and bias vector parameters which need to be learned during training.

where the superscripts and refer to the number of input features and number of hidden units, respectively.

Fig 6.3 Data Used



6.5 Explanation

LSTM is a recurrent neural network (RNN) architecture that **REMEMBERS** values over arbitrary intervals. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration. Relative **insensitivity to gap length** gives an advantage to LSTM over alternative RNNs, hidden Markov models and other sequence learning methods.

RNN structure

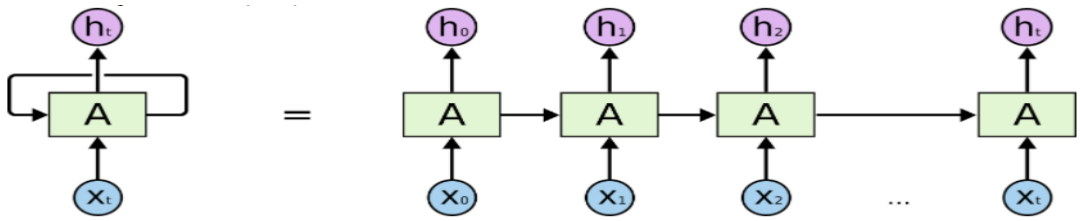


Fig 6.4 RNN Structure

The structure of RNN is very similar to hidden Markov model. However, the main difference is with how parameters are calculated and constructed. One of the advantage with LSTM is **insensitivity to gap length**. RNN and HMM rely on the hidden state before emission / sequence. If we want to predict the sequence after 1,000 intervals instead of 10, the model forgot the starting point by then.

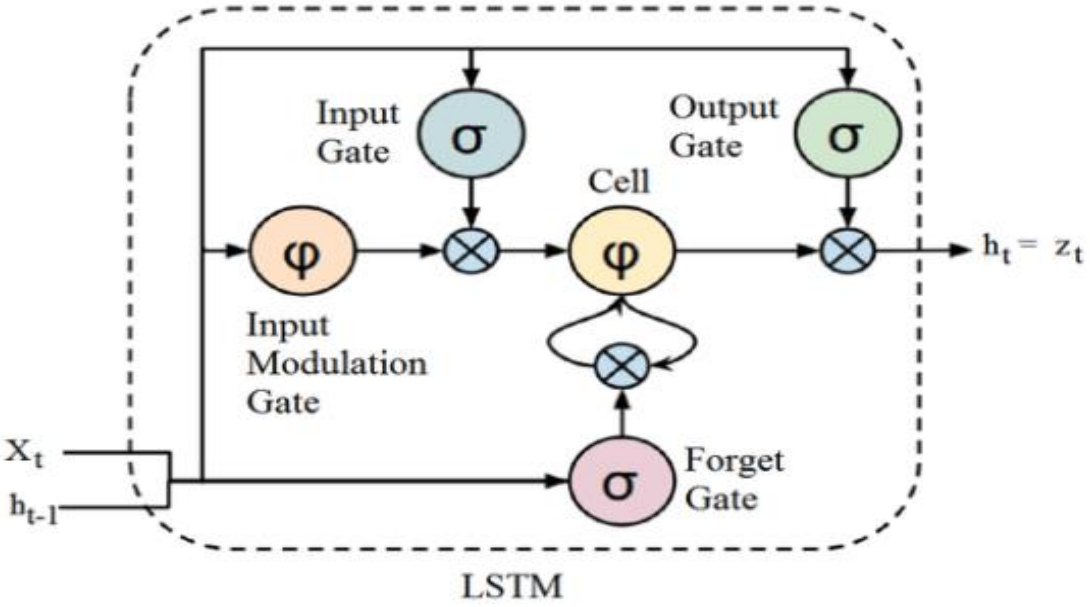


Fig 6.5 Cell Diagram

LSTM Cell

The **long-term memory** is usually called the **cell state**. The looping arrows indicate recursive nature of the cell. This allows information from previous intervals to be stored with in the LSTM cell. Cell state is modified by the forget gate placed below the cell state and also adjust by the input modulation gate. From equation, the previous cell state forgets by multiply with the forget gate and adds new information through the output of the input gates.

Forget Gate Equation

The **remember vector** is usually called the **forget gate**. The output of the forget gate tells the cell state which information to forget by multiplying 0 to a position in the matrix. If the output of the forget gate is 1, the information is kept in the cell state. From equation, sigmoid function is applied to the weighted input/observation and previous hidden state.

The **save vector** Is usually called the **input gate**. These gates determine which information should enter the cell state / long-term memory. The important parts are the activation functions for each gates. The input gate is a **sigmoid** function and have a range of [0,1]. Because the equation of the cell state is a summation between the previous cell state, sigmoid function alone will only add memory and not be able to remove/forget memory. If you can only add a float number between [0,1], that number will never be zero / turned-off / forget. This is why the input modulation gate has an **tanh** activation function. Tanh has a range of [-1, 1] and allows the cell state to forget memory.

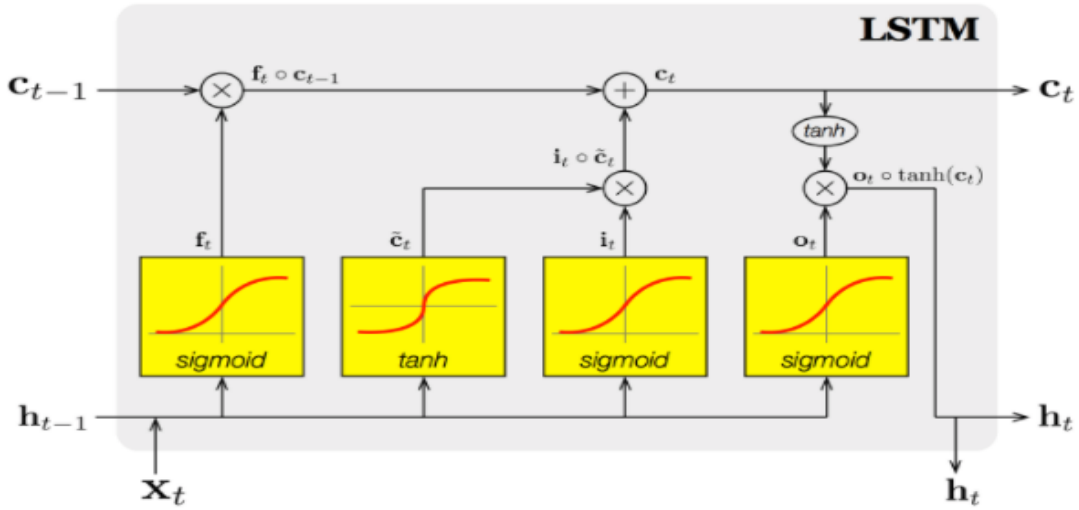


Fig 6.6 LSTM NETWORKING

The first sigmoid activation function is the **forget gate**. Which information should be forgotten from the previous cell state (Ct-1). The second sigmoid and first tanh activation function is our **input gate**. Which information should be saved to the cell state or should be forgotten? The last sigmoid is the **output gate** and highlights which information should be going to the next **hidden state**.

Proposed System Implementation Code

```
import collections

import os

import time

import numpy as np

import tensorflow as tf

from object_detection.core import box_list

from object_detection.core import box_list_ops

from object_detection.core import keypoint_ops

from object_detection.core import standard_fields as fields

from object_detection.metrics import coco_evaluation
```

```
from object_detection.utils import label_map_util

from object_detection.utils import object_detection_evaluation

from object_detection.utils import ops

from object_detection.utils import shape_utils

from object_detection.utils import visualization_utils as vis_utils

slim = tf.contrib.slim

EVAL_METRICS_CLASS_DICT = {

    'coco_detection_metrics':

        coco_evaluation.CocoDetectionEvaluator,

    'coco_mask_metrics':

        coco_evaluation.CocoMaskEvaluator,

    'oid_challenge_detection_metrics':

        object_detection_evaluation.OpenImagesDetectionChallengeEvaluator,

    'pascal_voc_detection_metrics':

        object_detection_evaluation.PascalDetectionEvaluator,

    'weighted_pascal_voc_detection_metrics':

        object_detection_evaluation.WeightedPascalDetectionEvaluator,

    'pascal_voc_instance_segmentation_metrics':
```

```
      object_detection_evaluation.PascalInstanceSegmentationEvaluator,
  'weighted_pascal_voc_instance_segmentation_metrics':
      object_detection_evaluation.WeightedPascalInstanceSegmentationEvaluator,
  'oid_V2_detection_metrics':
      object_detection_evaluation.OpenImagesDetectionEvaluator,
}
EVAL_DEFAULT_METRIC = 'coco_detection_metrics'
def write_metrics(metrics, global_step, summary_dir):
 tf.logging.info('Writing metrics to tf summary.')
 summary_writer = tf.summary.FileWriterCache.get(summary_dir)
 for key in sorted(metrics):
  summary = tf.Summary(value=[
      tf.Summary.Value(tag=key, simple_value=metrics[key]),
  ])
  summary_writer.add_summary(summary, global_step)
  tf.logging.info('%s: %f', key, metrics[key])
 tf.logging.info('Metrics written to tf summary.')
def visualize_detection_results(result_dict,
```

```
                    tag,
                    global_step,
                    categories,
                    summary_dir='',
                    export_dir='',
                    agnostic_mode=False,
                    show_groundtruth=False,
                    groundtruth_box_visualization_color='black',
                    min_score_thresh=.5,
                    max_num_predictions=20,
                    skip_scores=False,
                    skip_labels=False,
                    keep_image_id_for_visualization_export=False):
detection_fields = fields.DetectionResultFields
  input_fields = fields.InputDataFields
  if not set([
      input_fields.original_image,
      detection_fields.detection_boxes,
```

```
    detection_fields.detection_scores,
    detection_fields.detection_classes,
]).issubset(set(result_dict.keys())):
  raise ValueError('result_dict does not contain all expected keys.')
if show_groundtruth and input_fields.groundtruth_boxes not in result_dict:
  raise ValueError('If show_groundtruth is enabled, result_dict must contain '
                   'groundtruth_boxes.')
tf.logging.info('Creating detection visualizations.')
category_index = label_map_util.create_category_index(categories)

image = np.squeeze(result_dict[input_fields.original_image], axis=0)
if image.shape[2] == 1:  # If one channel image, repeat in RGB.
  image = np.tile(image, [1, 1, 3])
detection_boxes = result_dict[detection_fields.detection_boxes]
detection_scores = result_dict[detection_fields.detection_scores]
detection_classes = np.int32((result_dict[
    detection_fields.detection_classes]))
detection_keypoints = result_dict.get(detection_fields.detection_keypoints)
```

```
detection_masks = result_dict.get(detection_fields.detection_masks)

detection_boundaries = result_dict.get(detection_fields.detection_boundaries)



# Plot groundtruth underneath detections

if show_groundtruth:

  groundtruth_boxes = result_dict[input_fields.groundtruth_boxes]

  groundtruth_keypoints = result_dict.get(input_fields.groundtruth_keypoints)

  vis_utils.visualize_boxes_and_labels_on_image_array(

      image=image,

      boxes=groundtruth_boxes,

      classes=None,

      scores=None,

      category_index=category_index,

      keypoints=groundtruth_keypoints,

      use_normalized_coordinates=False,

      max_boxes_to_draw=None,

      groundtruth_box_visualization_color=groundtruth_box_visualization_color)

 vis_utils.visualize_boxes_and_labels_on_image_array(
```

```
    image,
    detection_boxes,
    detection_classes,
    detection_scores,
    category_index,
    instance_masks=detection_masks,
    instance_boundaries=detection_boundaries,
    keypoints=detection_keypoints,
    use_normalized_coordinates=False,
    max_boxes_to_draw=max_num_predictions,
    min_score_thresh=min_score_thresh,
    agnostic_mode=agnostic_mode,
    skip_scores=skip_scores,
    skip_labels=skip_labels)


if export_dir:
  if keep_image_id_for_visualization_export and result_dict[fields.
                                InputDataFields()
```

```
                                        .key]:
    export_path = os.path.join(export_dir, 'export-{}-{}.png'.format(
        tag, result_dict[fields.InputDataFields().key]))
  else:
    export_path = os.path.join(export_dir, 'export-{}.png'.format(tag))
  vis_utils.save_image_array_as_png(image, export_path)

summary = tf.Summary(value=[
    tf.Summary.Value(
        tag=tag,
        image=tf.Summary.Image(
            encoded_image_string=vis_utils.encode_image_array_as_png_str(
                image)))
])
summary_writer = tf.summary.FileWriterCache.get(summary_dir)
summary_writer.add_summary(summary, global_step)

tf.logging.info('Detection visualizations written to summary with tag %s.',
```

```
                tag)
def _run_checkpoint_once(tensor_dict,
                         evaluators=None,
                         batch_processor=None,
                         checkpoint_dirs=None,
                         variables_to_restore=None,
                         restore_fn=None,
                         num_batches=1,
                         master='',
                         save_graph=False,
                         save_graph_dir='',
                         losses_dict=None,
                         eval_export_path=None):
  if max_number_of_evaluations and max_number_of_evaluations <= 0:
    raise ValueError(
        '`number_of_steps` must be either None or a positive number.')


  if not checkpoint_dirs:
```

```
  raise ValueError('`checkpoint_dirs` must have at least one entry.')


last_evaluated_model_path = None
number_of_evaluations = 0
while True:
  start = time.time()
  tf.logging.info('Starting evaluation at ' + time.strftime(
      '%Y-%m-%d-%H:%M:%S', time.gmtime()))
  model_path = tf.train.latest_checkpoint(checkpoint_dirs[0])
  if not model_path:
    tf.logging.info('No model found in %s. Will try again in %d seconds',
                    checkpoint_dirs[0], eval_interval_secs)
  elif model_path == last_evaluated_model_path:
    tf.logging.info('Found already evaluated checkpoint. Will try again in '
                    '%d seconds', eval_interval_secs)
  else:
    last_evaluated_model_path = model_path
    global_step, metrics = _run_checkpoint_once(
```

```
        tensor_dict,
        evaluators,
        batch_processor,
        checkpoint_dirs,
        variables_to_restore,
        restore_fn,
        num_batches,
        master,
        save_graph,
        save_graph_dir,
        losses_dict=losses_dict,
        eval_export_path=eval_export_path)
    write_metrics(metrics, global_step, summary_dir)
  number_of_evaluations += 1


  if (max_number_of_evaluations and
      number_of_evaluations >= max_number_of_evaluations):
    tf.logging.info('Finished evaluation!')
```

```
      break

    time_to_next_eval = start + eval_interval_secs - time.time()

    if time_to_next_eval > 0:

      time.sleep(time_to_next_eval)



  return metrics




def _scale_box_to_absolute(args):

  boxes, image_shape = args

  return box_list_ops.to_absolute_coordinates(

      box_list.BoxList(boxes), image_shape[0], image_shape[1]).get()




def _resize_detection_masks(args):

  detection_boxes, detection_masks, image_shape = args

  detection_masks_reframed = ops.reframe_box_masks_to_image_masks(

      detection_masks, detection_boxes, image_shape[0], image_shape[1])
```

```
  return tf.cast(tf.greater(detection_masks_reframed, 0.5), tf.uint8)


def _resize_groundtruth_masks(args):
  mask, image_shape = args
  mask = tf.expand_dims(mask, 3)
  mask = tf.image.resize_images(
      mask,
      image_shape,
      method=tf.image.ResizeMethod.NEAREST_NEIGHBOR,
      align_corners=True)
  return tf.cast(tf.squeeze(mask, 3), tf.uint8)


def _scale_keypoint_to_absolute(args):
  keypoints, image_shape = args
  return keypoint_ops.scale(keypoints, image_shape[0], image_shape[1])
```

```
def result_dict_for_single_example(image,
                                   key,
                                   detections,
                                   groundtruth=None,
                                   class_agnostic=False,
                                   scale_to_absolute=False): if groundtruth:
  max_gt_boxes = tf.shape(
    groundtruth[fields.InputDataFields.groundtruth_boxes])[0]
  for gt_key in groundtruth:
   # expand groundtruth dict along the batch dimension.
   groundtruth[gt_key] = tf.expand_dims(groundtruth[gt_key], 0)


 for detection_key in detections:
  detections[detection_key] = tf.expand_dims(
    detections[detection_key][0], axis=0)


 batched_output_dict = result_dict_for_batched_example(
```

```
    image,
    tf.expand_dims(key, 0),
    detections,
    groundtruth,
    class_agnostic,
    scale_to_absolute,
    max_gt_boxes=max_gt_boxes)

exclude_keys = [
    fields.InputDataFields.original_image,
    fields.DetectionResultFields.num_detections,
    fields.InputDataFields.num_groundtruth_boxes
]

output_dict = {
    fields.InputDataFields.original_image:
        batched_output_dict[fields.InputDataFields.original_image]
}
```

```
  for key in batched_output_dict:
    # remove the batch dimension.
    if key not in exclude_keys:
      output_dict[key] = tf.squeeze(batched_output_dict[key], 0)
  return output_dict



def result_dict_for_batched_example(images,
                                    keys,
                                    detections,
                                    groundtruth=None,
                                    class_agnostic=False,
                                    scale_to_absolute=False,
                                    original_image_spatial_shapes=None,
                                    true_image_shapes=None,
                                    max_gt_boxes=None):
  """Merges all detection and groundtruth information for a single example.
```

Note that evaluation tools require classes that are 1-indexed, and so this function performs the offset. If `class_agnostic` is True, all output classes have label 1.

Args:

images: A single 4D uint8 image tensor of shape [batch_size, H, W, C].

keys: A [batch_size] string tensor with image identifier.

detections: A dictionary of detections, returned from DetectionModel.postprocess().

groundtruth: (Optional) Dictionary of groundtruth items, with fields:

'groundtruth_boxes': [batch_size, max_number_of_boxes, 4] float32 tensor of boxes, in normalized coordinates.

'groundtruth_classes': [batch_size, max_number_of_boxes] int64 tensor of 1-indexed classes.

'groundtruth_area': [batch_size, max_number_of_boxes] float32 tensor of bbox area. (Optional)

'groundtruth_is_crowd':[batch_size, max_number_of_boxes] int64

tensor. (Optional)

'groundtruth_difficult': [batch_size, max_number_of_boxes] int64

tensor. (Optional)

'groundtruth_group_of': [batch_size, max_number_of_boxes] int64

tensor. (Optional)

'groundtruth_instance_masks': 4D int64 tensor of instance

masks (Optional).

class_agnostic: Boolean indicating whether the detections are class-agnostic

(i.e. binary). Default False.

scale_to_absolute: Boolean indicating whether boxes and keypoints should be

scaled to absolute coordinates. Note that for IoU based evaluations, it

does not matter whether boxes are expressed in absolute or relative

coordinates. Default False.

original_image_spatial_shapes: A 2D int32 tensor of shape [batch_size, 2]

used to resize the image. When set to None, the image size is retained.

true_image_shapes: A 2D int32 tensor of shape [batch_size, 3]

containing the size of the unpadded original_image.

max_gt_boxes: [batch_size] tensor representing the maximum number of

groundtruth boxes to pad.

Returns:

A dictionary with:

'original_image': A [batch_size, H, W, C] uint8 image tensor.

'original_image_spatial_shape': A [batch_size, 2] tensor containing the

original image sizes.

'true_image_shape': A [batch_size, 3] tensor containing the size of

the unpadded original_image.

'key': A [batch_size] string tensor with image identifier.

'detection_boxes': [batch_size, max_detections, 4] float32 tensor of boxes,

in normalized or absolute coordinates, depending on the value of

`scale_to_absolute`.

'detection_scores': [batch_size, max_detections] float32 tensor of scores.

'detection_classes': [batch_size, max_detections] int64 tensor of 1-indexed

classes.

'detection_masks': [batch_size, max_detections, H, W] float32 tensor of

binarized masks, reframed to full image masks.

'num_detections': [batch_size] int64 tensor containing number of valid detections.

'groundtruth_boxes': [batch_size, num_boxes, 4] float32 tensor of boxes, in normalized or absolute coordinates, depending on the value of `scale_to_absolute`. (Optional)

'groundtruth_classes': [batch_size, num_boxes] int64 tensor of 1-indexed classes. (Optional)

'groundtruth_area': [batch_size, num_boxes] float32 tensor of bbox area. (Optional)

'groundtruth_is_crowd': [batch_size, num_boxes] int64 tensor. (Optional)

'groundtruth_difficult': [batch_size, num_boxes] int64 tensor. (Optional)

'groundtruth_group_of': [batch_size, num_boxes] int64 tensor. (Optional)

'groundtruth_instance_masks': 4D int64 tensor of instance masks (Optional).

'num_groundtruth_boxes': [batch_size] tensor containing the maximum number of groundtruth boxes per image.

Raises:

```
  ValueError: if original_image_spatial_shape is not 2D int32 tensor of shape
    [2].
  ValueError: if true_image_shapes is not 2D int32 tensor of shape
    [3].
"""
label_id_offset = 1  # Applying label id offset (b/63711816)

input_data_fields = fields.InputDataFields
if original_image_spatial_shapes is None:
  original_image_spatial_shapes = tf.tile(
      tf.expand_dims(tf.shape(images)[1:3], axis=0),
      multiples=[tf.shape(images)[0], 1])
else:
  if (len(original_image_spatial_shapes.shape) != 2 and
      original_image_spatial_shapes.shape[1] != 2):
    raise ValueError(
        '`original_image_spatial_shape` should be a 2D tensor of shape '
        '[batch_size, 2].')
```

```
if true_image_shapes is None:
  true_image_shapes = tf.tile(
      tf.expand_dims(tf.shape(images)[1:4], axis=0),
      multiples=[tf.shape(images)[0], 1])
else:
  if (len(true_image_shapes.shape) != 2
      and true_image_shapes.shape[1] != 3):
    raise ValueError(`true_image_shapes` should be a 2D tensor of '
                     'shape [batch_size, 3].')


output_dict = {
    input_data_fields.original_image:
        images,
    input_data_fields.key:
        keys,
    input_data_fields.original_image_spatial_shape: (
        original_image_spatial_shapes),
```

```
    input_data_fields.true_image_shape:
      true_image_shapes
}

detection_fields = fields.DetectionResultFields
detection_boxes = detections[detection_fields.detection_boxes]
detection_scores = detections[detection_fields.detection_scores]
num_detections = tf.to_int32(detections[detection_fields.num_detections])

if class_agnostic:
  detection_classes = tf.ones_like(detection_scores, dtype=tf.int64)
else:
  detection_classes = (
      tf.to_int64(detections[detection_fields.detection_classes]) +
      label_id_offset)

if scale_to_absolute:
  output_dict[detection_fields.detection_boxes] = (
```

```
    shape_utils.static_or_dynamic_map_fn(
        _scale_box_to_absolute,
        elems=[detection_boxes, original_image_spatial_shapes],
        dtype=tf.float32))
else:
  output_dict[detection_fields.detection_boxes] = detection_boxes
output_dict[detection_fields.detection_classes] = detection_classes
output_dict[detection_fields.detection_scores] = detection_scores
output_dict[detection_fields.num_detections] = num_detections


if detection_fields.detection_masks in detections:
  detection_masks = detections[detection_fields.detection_masks]
  # TODO(rathodv): This should be done in model's postprocess
  # function ideally.
  output_dict[detection_fields.detection_masks] = (
    shape_utils.static_or_dynamic_map_fn(
        _resize_detection_masks,
        elems=[detection_boxes, detection_masks,
```

```
            original_image_spatial_shapes],
        dtype=tf.uint8))

if detection_fields.detection_keypoints in detections:
  detection_keypoints = detections[detection_fields.detection_keypoints]
  output_dict[detection_fields.detection_keypoints] = detection_keypoints
  if scale_to_absolute:
    output_dict[detection_fields.detection_keypoints] = (
      shape_utils.static_or_dynamic_map_fn(
        _scale_keypoint_to_absolute,
        elems=[detection_keypoints, original_image_spatial_shapes],
        dtype=tf.float32))

if groundtruth:
  if max_gt_boxes is None:
    if input_data_fields.num_groundtruth_boxes in groundtruth:
      max_gt_boxes = groundtruth[input_data_fields.num_groundtruth_boxes]
    else:
```

```
    raise ValueError(
        'max_gt_boxes must be provided when processing batched examples.')


if input_data_fields.groundtruth_instance_masks in groundtruth:
  masks = groundtruth[input_data_fields.groundtruth_instance_masks]
  groundtruth[input_data_fields.groundtruth_instance_masks] = (
      shape_utils.static_or_dynamic_map_fn(
          _resize_groundtruth_masks,
          elems=[masks, original_image_spatial_shapes],
          dtype=tf.uint8))


output_dict.update(groundtruth)
if scale_to_absolute:
  groundtruth_boxes = groundtruth[input_data_fields.groundtruth_boxes]
  output_dict[input_data_fields.groundtruth_boxes] = (
      shape_utils.static_or_dynamic_map_fn(
          _scale_box_to_absolute,
          elems=[groundtruth_boxes, original_image_spatial_shapes],
```

```
            dtype=tf.float32))

    # For class-agnostic models, groundtruth classes all become 1.
    if class_agnostic:
      groundtruth_classes = groundtruth[input_data_fields.groundtruth_classes]
      groundtruth_classes = tf.ones_like(groundtruth_classes, dtype=tf.int64)
      output_dict[input_data_fields.groundtruth_classes] = groundtruth_classes

    output_dict[input_data_fields.num_groundtruth_boxes] = max_gt_boxes

  return output_dict


def get_evaluators(eval_config, categories, evaluator_options=None):
  """Returns the evaluator class according to eval_config, valid for categories.

  Args:
    eval_config: An `eval_pb2.EvalConfig`.
```

categories: A list of dicts, each of which has the following keys -

'id': (required) an integer id uniquely identifying this category.

'name': (required) string representing category name e.g., 'cat', 'dog'.

evaluator_options: A dictionary of metric names (see

EVAL_METRICS_CLASS_DICT) to `DetectionEvaluator` initialization

keyword arguments. For example:

evalator_options = {

'coco_detection_metrics': {'include_metrics_per_category': True}

}

Returns:

An list of instances of DetectionEvaluator.

Raises:

ValueError: if metric is not in the metric class dictionary.

"""

evaluator_options = evaluator_options or {}

eval_metric_fn_keys = eval_config.metrics_set

```
  if not eval_metric_fn_keys:
    eval_metric_fn_keys = [EVAL_DEFAULT_METRIC]
  evaluators_list = []
  for eval_metric_fn_key in eval_metric_fn_keys:
    if eval_metric_fn_key not in EVAL_METRICS_CLASS_DICT:
      raise ValueError('Metric not found: {}'.format(eval_metric_fn_key))
    kwargs_dict = (evaluator_options[eval_metric_fn_key] if eval_metric_fn_key
                   in evaluator_options else {})
    evaluators_list.append(EVAL_METRICS_CLASS_DICT[eval_metric_fn_key](
        categories,
        **kwargs_dict))
  return evaluators_list


def get_eval_metric_ops_for_evaluators(eval_config,
                                       categories,
                                       eval_dict):
  """Returns eval metrics ops to use with `tf.estimator.EstimatorSpec`.
```

Args:

eval_config: An `eval_pb2.EvalConfig`.

categories: A list of dicts, each of which has the following keys -

'id': (required) an integer id uniquely identifying this category.

'name': (required) string representing category name e.g., 'cat', 'dog'.

eval_dict: An evaluation dictionary, returned from

result_dict_for_single_example().

Returns:

A dictionary of metric names to tuple of value_op and update_op that can be

used as eval metric ops in tf.EstimatorSpec.

"""

```
eval_metric_ops = {}
evaluator_options = evaluator_options_from_eval_config(eval_config)
evaluators_list = get_evaluators(eval_config, categories, evaluator_options)
for evaluator in evaluators_list:
  eval_metric_ops.update(evaluator.get_estimator_eval_metric_ops(
```

```
      eval_dict))

  return eval_metric_ops




def evaluator_options_from_eval_config(eval_config):

  """Produces a dictionary of evaluation options for each eval metric.



  Args:

    eval_config: An `eval_pb2.EvalConfig`.



  Returns:

    evaluator_options: A dictionary of metric names (see

      EVAL_METRICS_CLASS_DICT) to `DetectionEvaluator` initialization

      keyword arguments. For example:

      evalator_options = {

        'coco_detection_metrics': {'include_metrics_per_category': True}

      }

  """
```

```
eval_metric_fn_keys = eval_config.metrics_set

evaluator_options = {}

for eval_metric_fn_key in eval_metric_fn_keys:

  if eval_metric_fn_key in ('coco_detection_metrics', 'coco_mask_metrics'):

    evaluator_options[eval_metric_fn_key] = {

        'include_metrics_per_category': (

            eval_config.include_metrics_per_category)

    }

return evaluator_options
```

CHAPTER 7

RESULTS AND DISCUSSIONS

Here is what we have done, we have integrated our project into a folder inside which it contains a flask designed folder which indeed contains the desired files to run the project. The flask contains both the front-end and the back-end codes. We now go to command prompt and run the flask folder which generates a localhost to access the front-end visual first look of our project. Once we enter our localhost, we need to upload image into the choose file option and the machine learning will learn from the image and then generate the final desired output.

Thus, in our project succeeds only when the accuracy is maintained between 80-100% .

NO. OF FINGERS	*ACTION - PERFORMED*
1	**Standing**
2	**Sitting**
	Bending
3	
	Bending Backward
4	
	Sleeping

In real time by using a web camera the input video is taken and converted into frames then some of the steps are carried out as shown in the figures to count the number of fingers. The experimental results are shown below:

1. The Procured image is RGB and must be processed before i.e. pre-processed before the components are separated and acknowledgement is made and is shown in figure 7.1.

Screenshots Section

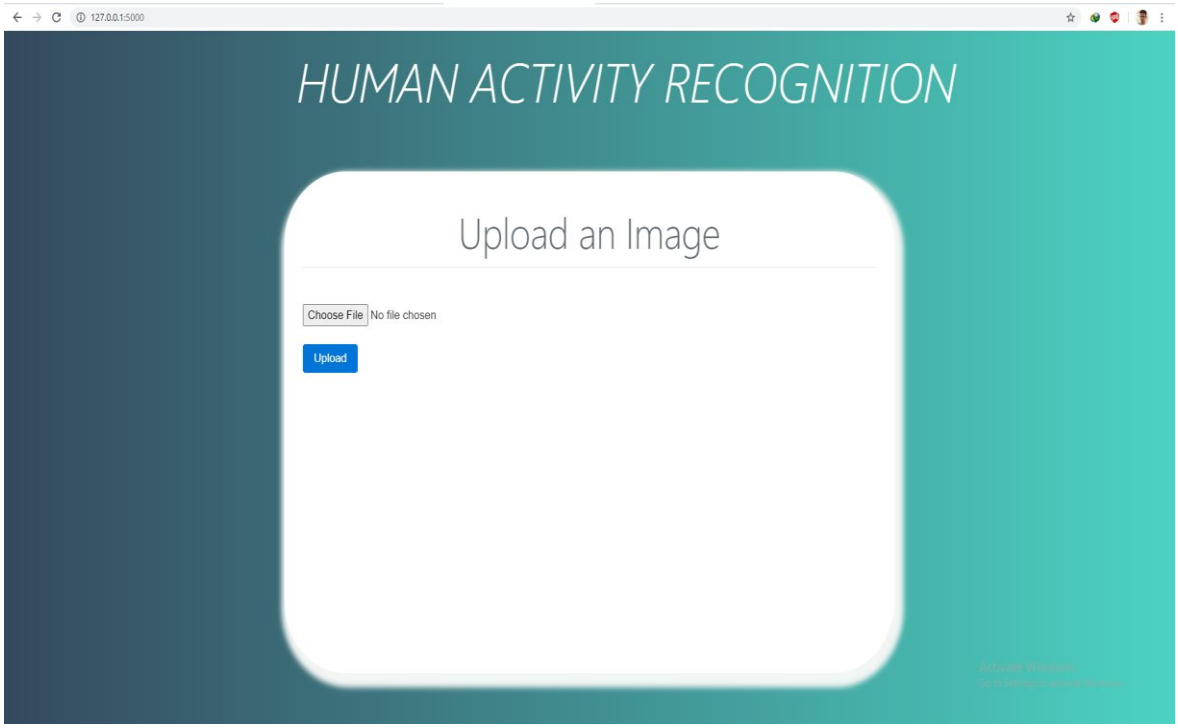


Fig 7.1 Base Front-End

Fig 7.2 Activity Test Standing

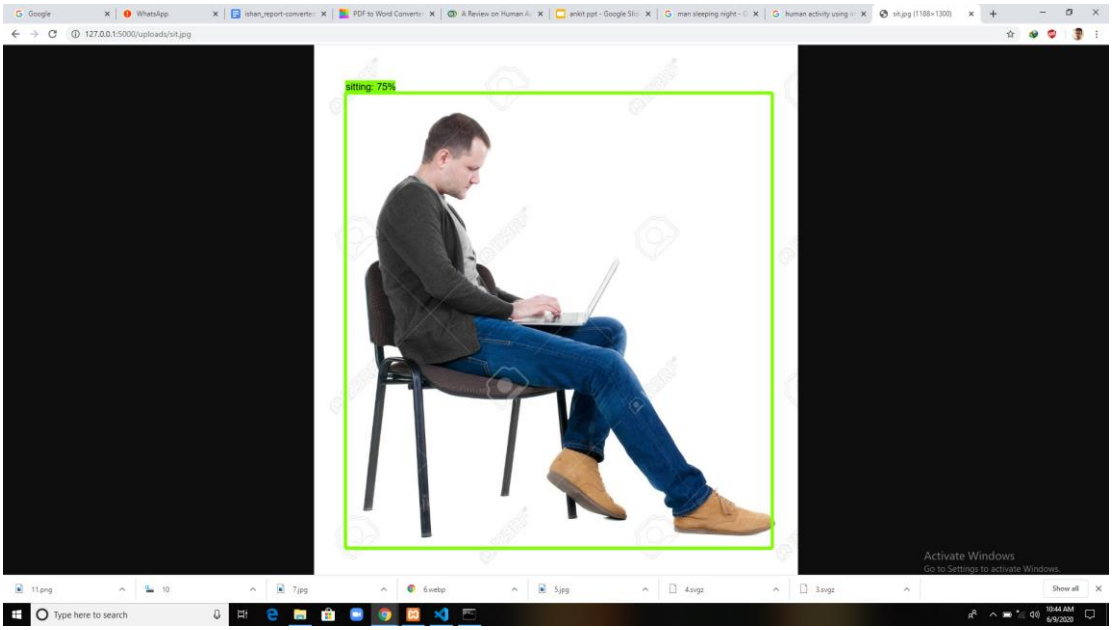


Fig 7.3 Activity Test sitting

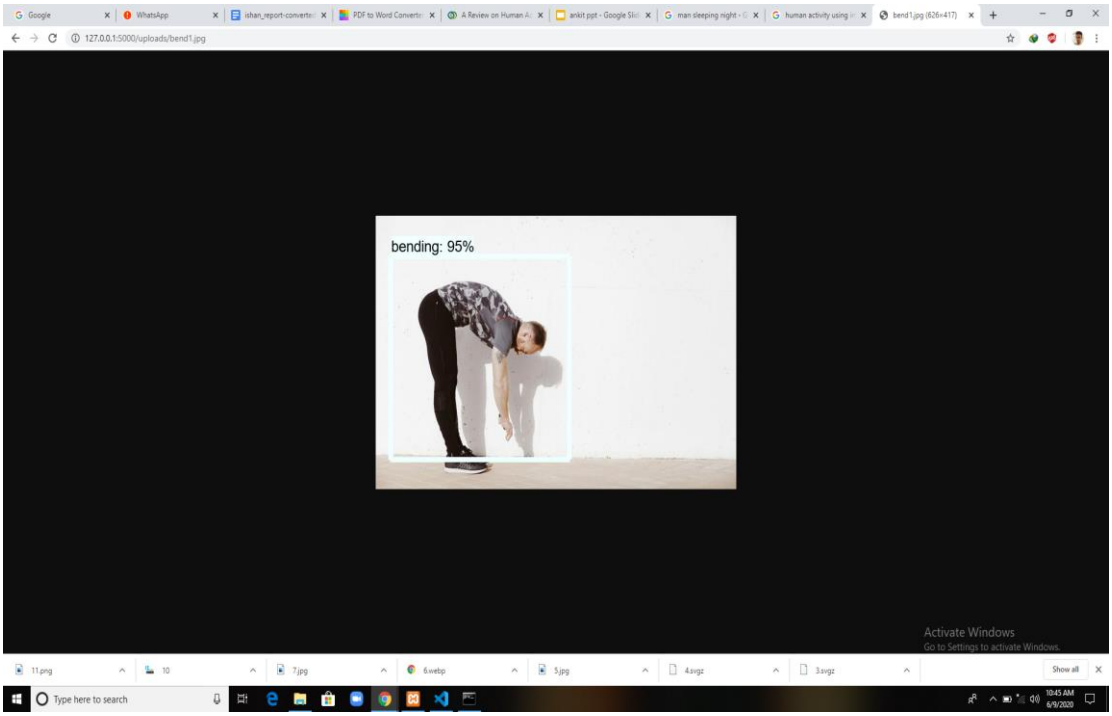


Fig 7.4 Activity Test Bending

CHAPTER 8

TESTING

Human Activity Recognition System testing is actually a image detection technique to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all the system elements have been properly integrated and perform allocated functions. The testing process is actually carried out to make sure that the algorithms and implementations exactly does the same thing it is supposed to do. In the testing stage following goals are tried to achieve: -

- To affirm the quality of the project.
- To find and eliminate any residual errors from previous stages.
- To validate the software as a solution to the original problem.
- To provide operational reliability of the system.

8.1 Quality Assurance

Quality assurance consists of the auditing and reporting functions of management. The goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confidence that the product quality is meeting its goals. This is an “umbrella activity” that is applied throughout the engineering process**.** Software quality assurance encompasses: -

- Analysis, design, coding and testing methods and tools
- Formal technical reviews that are applied during each software engineering
- Multi -tiered testing strategy
- Control of software documentation and the change made to it.
- Measurement and reporting mechanisms.

8.2 Quality Factors

An important objective of quality assurance is to track the software quality and assess the

impact of methodological and procedural changes on improved software quality. The factors that affect the quality can be categorized into two broad groups:

- Factors that can be directly measured.
- Factors that can be indirectly measured
- Effectiveness or efficiency in performing its missio
- Its ability to undergo changes
- Its adaptability to a new environment.
- Duration of its use by its customer.

8.3 Functional Test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is cantered on the following items:

Valid Input identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised. Systems/Procedures: Interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

CHAPTER 9

CONCLUSION AND FUTURE

9.1 Conclusion

The proposed system is a real time image processing of a person that is based on a real time application system. This can allow the system to keep track of a person and its movements using a webcam of the laptop and the image that it uploads

In this project we have planned, designed and implemented the system for Human activity recognition system for controlling UI which is a standalone application for controlling the various user interface controls and/or programs like Flak. In the analysis phase we gathered four major actions done by a human on his daily basis whether it be standing, sitting, sleeping and bending and the techniques and algorithms they employ and the success/failure rate of these systems. Accordingly, we made a detailed comparison of these systems and analysed their efficiency. In the design phase we designed the system architecture diagrams and also the data flow diagram of the system. We studied and analysed the different phases involved and accordingly designed and studied the algorithms to be used for the same.

With the help of observations that we have, we can conclude that the results should be depend upon:

- The accuracy of the machine to detect the current action of the human at that

particular moment of time. This accuracy is the success of the machine that it learns from the detection techniques the accuracy can thus vary based on the loss and success ratios. Finally, we were able to achieve a maximum accuracy of 98% which proved our deployment to be precise and accurate

- Background of the pictures should be plain to get accurate analysis of recognition of gestures and poses .
- The image should be a jpg extension to keep things straight forward and code to be clean and easily usable because jpg/png images are easily available.

9.2 Future Scope

The future scope of Human activity recognition is a vast demonstration of categories on which its application can be achieved:-

- CCTV cameras fixed on the pillars of a bank can be a human tracking device which tracks the movements of each person in the bank and captures their actions and if something goes way past the limits it can send data to the manager.
- Traffic lights can be encapsulated with cameras to track car movements and speed limit accuracies.
- Working on true surveillance video tracks, sport videos, movies, and online video data, will help to discover the real requirements for action recognition, and it will help researchers to shift focus to other important issues involved in action recognition.

REFERENCES

[1] H. Renuka and B.Goutam ―Hand Gestures Recognition system to control soft front panels, International Journal of Engineering Research and Technology, December 2014.

[2] :Henrik Birk and Thomas Baltzer Moeslund, ―Recognizing Gestures From the Hand Alphabet Using Principal Component Analysis‖, Master‘s Thesis, Laboratory of Image Analysis, Aalborg University, Denmark, 1996.

[3] :Punam Thakare / International Journal on Computer Science and Engineering (IJCSE) Assistant Professor: MCA Dept , ICEM, Parandwadi Pune, India.

[4]: Prof. Praveen D. Hasalkar1, Rohit S. Chougule2, Vrushabh B. Madake3, Vishal S. Magdum/ *International Journal of Advanced Research in Computer and Communication Engineering*, Department of Computer Science and Engineering, W.I.T, Solapur, Maharashtra, India