A Mini Project Report

on

PEOPLE COUNTER USING OPENCV

submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

CHINTAPALLI GANGADHAR (17211A0549)
BHEEMA PRAVEEN KUMAR (17211A0535)
BIRADAR TULSIDAS (17211A0539)
BOGGULA ANAND (18215A0506)

Under the guidance of

Dr. AMJAN SHAIK, Ph.D,
Professor, Associate HoD

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

B.V.RAJU INSTITUTE OF TECHNOLOGY

(UGC Autonomous, Accredited by NBA & NAAC)

Vishnupur, Narspur, Medak(Dist.), Telangana State, India - 502313

2019 - 2020

B. V. Raju Institute of Technology
(UGC Autonomous, Accredited By NBA & NAAC)
Vishnupur, Narspur, Medak (Dist.),
Telangana State, India – 502313

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Mini Project entitled **"PEOPLE COUNTER USING OPENCV"**, being submitted by

CHINTAPALLI GANGADHAR (17211A0549)

BHEEMA PRAVEEN KUMAR (17211A0535)

BIRADAR TULSIDAS (17211A0539)

BOGGULA ANAND (18215A0506)

In partial fulfillment of the requirements for the award of degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING to B.V.RAJU INSTITUTE OF TECHNOLOGY is a record of bonafide work carried out during a period from May 2019 to July 2020 by them under the guidance of **Dr. Amjan Shaik**, Professor & Associate HoD, CSE Department.

This is to certify that the above statement made by the students is/are correct to the best of my knowledge.

Dr. Amjan Shaik
Professor & Associate HoD

The Project Viva-Voce Examination of this team has been held on ____________________.

Mr. Karthik Kovuri	**Dr. Ch. Madhu Babu**	
Project Coordinator	Professor & HoD-CSE	EXTERNAL EXAMINER

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project entitled **"PEOPLE COUNTER USING OPENCV"** in partial fulfillment of the requirements for the award of Degree of Bachelor of Technology and submitted in the Department of Computer Science and Engineering, B. V. Raju Institute of Technology, Narsapur is an authentic record of my own work carried out during a period from May 2019 to July 2021 under the guidance of **Dr. Amjan Shaik**, Professor & Associate HoD. The work presented in this project report has not been submitted by us for the award of any other degree of this or any other Institute/University.

CHINTAPALLI GANGADHAR (17211A0549)
BHEEMA PRAVEEN KUMAR (17211A0535)
BIRADAR TULSIDAS (17211A0539)
BOGGULA ANAND (18215A0506)

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely fortunate to have got this all along the completion. Whatever we have done is due to such guidance and assistance. We would not forget to thank them.

We thank **Dr. Amjan Shaik** for guiding us and providing all the support in completing this project. We are thankful to **Mrs. Sreedevi**, our section project coordinator for supporting us in doing this project. We are thankful to **Mr. Karthik Kovuri**, project coordinator for helping us in completing the project in time. We thank the person who has our utmost gratitude is **Dr. Ch. Madhu Babu**, Head of CSE Department.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from all the staff members of CSE Department.

CHINTAPALLI GANGADHAR (17211A0549)
BHEEMA PRAVEEN KUMAR (17211A0535)
BIRADAR TULSIDAS (17211A0539)
BOGGULA ANAND (18215A0506)

PEOPLE COUNTER USING OPENCV

ABSTRACT

A Human Computer Interaction system for an automatic face recognition has attracted increasing attention from researchers in psychology, computer science, linguistics, neuroscience, and related disciplines. The computer vision community has expended a great amount of effort in recent years towards the goal of tracking people in images. Much more recently, algorithms have been developed to track multiple people robustly and in real-time. The goal of this project is to implement a system based on one of those algorithms, in order to the people in a database of image footage. Due to several constraints and performance issues, however, a more straightforward algorithm based on background subtraction is implemented and shows acceptable performance levels People Counter is an end user application which detects the number of people in the image that is being given to the system. The purpose of the present project was to develop an intelligent system for Counting the number of people present in an image. A human-computer interaction system for an automatic face recognition has attracted increasing attention from researchers in psychology, computer science, linguistics, neuroscience, and related discipline.

Key Words: OpenCV, Face recognition, Haar Classifier.

TEAM MEMBERS:

1. CHINTAPALLI GANGADHAR – 17211A0549
2. BHEEMA PRAVEEN KUMAR – 17211A0535
3. BIRADAR TULSIDAS – 17211A0539
4. BOGGULA ANAND – 18215A0506

GUIDE:

Dr. Amjan Shaik, Professor & Associate HoD

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

1.	**FACS**	**-**	Facial Action Coding System
2.	**AU**	**-**	Action Units
3.	**GUI**	**-**	Graphical User Interface
4.	**ML**	**-**	Machine Learning
5.	**API**	**-**	Application Program Interface
6.	**CPU**	**-**	Central Processing Unit
7.	**GPU**	**-**	Graphics Processing Unit

CONTENTS

1. INTRODUCTION

Face recognition is one of the many wonders that AI research has brought forward to the world. It is a subject of curiosity for many techies who would like to have a basic understanding of how things work. People counting is a crucial and challenging problem in visual surveillance. Automatic monitoring of the number of people in public areas is important for safety control and urban planning. For this purpose many techniques and methods have been proposed. These techniques are not producing high performance and better accuracy for complicated scenes counting. Recently Foreground Extraction and Expectation Maximization (EM) based methods are proposed, which provides a better accurate solution for people counting.

Reliable people counting and human detection is an important problem in visual surveillance. An accurate and real time estimation of people in a shopping mall can provide valuable information for managers. In recent years, this field has seen many advances, but the solutions have restrictions: people must be moving, the background must be simple, or the image resolution must be high. However, real scenes always include both moving and stationary human beings, the background may be complicated, and most videos in a visual surveillance system have a relatively low resolution. In a key factor in the solutions described in the use of global or semiglobal pixel intensity values to infer crowd behaviour avoiding recognition and tracking of individual pedestrians. Human beings perceive images through their properties like colour, shape, size, and texture.

1.1 MOTIVATION

To predict "The People count in a image" is the main motive of this project. The technique is demonstrated by developing a system that locates people in cluttered scenes. In particular, the system detects the components of a person's body in an image, i.e., the head.

Whenever the user gives Image , it is taken as input , he/she would be getting the Number of people present in an image as output. Results are of highest accuracy. Conclusion is made on the based on HAAR classifier for face detection.

1.2 PROBLEM STATEMENT

People counter is an end user application which detects the Face of the person in the image being given to the system. The relevant Number of persons in the image is shown on the screen which changes with the change in the given image. We implement HAAR classifier for face detection.

1.3 OBJECTIVE

- Our main objective is to understand the facial detection and count them.
- To investigate Number of people using facial detection and give the relevant output. HAAR are used for face detection respectively.

2. LITERATURE SURVEY

Current population of the world is approximately 7.7 billion according to recent statistical report. Today all regions in the world are connected with some form of transport systems cities in all countries are filled with luxury multi-purpose malls, stadiums and so on. Where-ever we go all over the world we are facing one or other problems with the crowd due to increasing population and more modern development in technology. So there is a need for some responsible technology to overcome the problems created by increasing population; like automated systems for finding Tourists flow estimation in-order to provide proper resources to them which in-turn attracts many Tourists who will again increase the countries revenue; for actively managing city services for public comfort; Crowd behavior modeling, disaster prevention and crowd control for public safety; some statistical applications like allocation of resources for public events, usage statistics to public transport systems, finding occupancy limit of a building and crowd behavior can help architects and town-planners to design safer buildings and real-time estimation of people in a shopping mall can provide valuable information for managers.

Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio proposed a general example-based framework for detecting objects in static images by components. The technique is demonstrated by developing a system that locates people in cluttered scenes. In particular, the system detects the components of a person’s body in an image, i.e., the head, the left and right arms, and the legs, instead of the full body by using four distinct example based detectors. The system then checks to ensure that the detected components are in the proper geometric configuration.

Automated crowd density estimation and counting are popular and important topic in crowd analysis.The last decades witnessed different of many significant publications in this field and it has been and stilla challenging problem for automatic visual surveillance over many years. This paper presents a survey oncrowd density estimation and counting methods employed for visual surveillance in the perspective of computer vision research. This surveycovers two main approaches which are direct approach (i.e., objectbased target detection) and indirect approach (e.g. pixel-based, texture-based, and corner points basedanalysis).

The categories of crowd counting in image falls in two broad categories: (a) ROI counting which estimates the total number of people in some regions at certain time instance (b) LOI counting which counts people who crosses a detecting line in certain time duration. The LOI counting can be developed using feature tracking techniques where the features are either tracked into trajectories and these trajectories are clustered into object tracks or based on extracting and counting crowd blobs from a temporal slice of the video. And the ROI counting can be developed using two techniques: Detection Based and Feature Based and Pixel Regression Techniques. Detection based methods detect people individually and count them. It utilizes any of the following methods:- Background Differencing, Motion and Appearance joint segmentation, Silhouette or shape matching and Standard object recognition method. Regression approaches extract the features such as foreground pixels and interest points, and vectors are formed with those features and it uses machine learning algorithms to subside the number of pedestrians or people. Some of the common features according to recent survey are edges, wavelet coefficients, and combination of large set of features. Some of the common Regressions are Linear Regression, Neural Networks, Gaussian Process Regression and Discrete Classifiers. This paper aims at presenting a decade survey on people (crowd) counting in surveillance videos.

3. ANALYSIS

The aim in this phase is studying the existing system and later is to interpret the necessities, requirements and domain of the new system. The mentioned both activities are balancingly salient, but the first activity serves as a basis of giving the functional specifications and then successful design of the proposed system. Understanding the effects and essentials of a new system is more difficult and requires creative thinking and understanding of existing running system is also strenuous, improper understanding of present system can lead deflection from solution.

3.1 EXISTING SYSTEM

Identification of the Face detection is made possible by various techniques, but these usually don't always generate the desired results. Therefore, pointed out the need for a general application which predicts the most accurate results. The existing system is also more time consuming and less accurate. The system only provides face detection, it doesn't give the count. The existing system doesn't change with the given image.

DISADVANTAGES

- Most of these approaches are time consuming.
- The results produced are also less accurate, i.e., the number of people predicted are not correct .
- Complexity in code for building.

3.2 PROPOSED SYSTEM

The proposed system is an application where in the user can predict the number of people. The time consumed is less in this and also the results obtained are comparatively more accurate to the existing system. Proposed system usually would include the use of open source technologies that would help out in many aspects. Open source technology OpenCv is used to detect the faces and count the number of people in it.

ADVANTAGES:

- Best strategy is to finding the number of people.
- Accurate and less time consuming
- The project is very useful in Security surveillance and facial detection.

3.3 SOFTWARE REQUIREMENT SPECIFICATION.

3.3.1.PURPOSE

A system with any operating system must have python (>= 3.3), OpenCV (python3-version) and PyCharm installed as Functional Requirement. As a part of Nonfunctional requirements, or performance requirements or quality of service requirements, the system must be usable, available, reliable, supportable, testable and maintainable.

3.3.2.SCOPE

The fundamental extension is to increase the dataset. Also the time taken to generate the result is very less. The accuracy of the project may be increased with the by providing the dataset. At present, the project is detecting facial and counting the people further crowd analysis can be added for increased performance. Robust spontaneous person recognizer can be developed and deployed in real-time systems. This is going to have an impact in Security surveillance for counting the number of people.

3.3.3 OVERALL DESCRIPTION

People counting is a crucial and challenging problem in visual surveillance. Automatic monitoring of the number of people in public areas is important for safety control and urban planning. For this purpose many techniques and methods have been proposed. These techniques are not producing high performance and better accuracy for complicated scenes counting. People counter is an end user application which detects the Face of the person in the image being given to the system. The relevant Number of persons in the image is shown on the screen which changes with the change in the given image. We implement HAAR classifier for face detection.

3.4 ECONOMIC FEASIBILITY

The system is economically feasible. It does not require any addition hardware or software besides the normal working system with Python (>= 3.3),OpenCV (python3-version), Pycharm GUI. Since the interface for this system is developed using the existing resources and technologies available online. There is no expenditure for certain.

3.5 SOFTWARE REQUIREMENTS

These software requirements are the specification of the system.. The software requirements provide a basis for creating the software requirements specification.

□ Operating System	:	Windows XP□
□ Coding Language	:	Python□
□ Front End	:	PyCharm GUI□

3.6 HARDWARE REQUIREMENTS

These hardware requirements serve as the basis for a contract for the implementation of the system and hence are a complete and consistent specification of the whole system.

□ Processor	-	Pentium –IV□
□ Speed	-	1.1 Ghz□
□ Ram	-	256 Mb□
□ Hard Disk	-	20 Gb□
□ Key Board	-	Standard Windows Keyboard□
□ Mouse	-	Two or Three Button Mouse□

4. IMPLEMENTATION

The application is based on face detection and counting processes.Face detection locates a face in a given image and separates it from the remaining scene, while a general definition of face recognition can be formulated as follows: “Given an image capture of a scene with one or more faces, detect and classify each face and shows a count relevant to it.”

Face recognition application will be able to connect and send a photo to a server. The server will detect faces in that image, perform face recognition and, using the recognized persons , extract information about those persons. The count relevant to that image will be displayed. Image face detection and recognition software which will be used to generate face images and to test recognition performances.

Modules

The following **modules** are developed:

1. **The Face Detection module** will locate faces in a given image and separate them from the scene. The extracted faces will be further transformed using processing techniques like grayscale transformation (all faces used in recognition will be grayscale images because are less sensitive in background changes and more computationally efficient), histogram equalization (for consistent brightness and contrast), resizing (to have the same dimension as images in the training database). Histogram equalization and resizing are used to reduce the recognition module’s lighting and scaling sensitivity.
2. **The Face Recognition module** will recognize faces provided by the detection module. It will use the Eigenfaces algorithm for recognition and Principal Component Analysis for dimensionality reduction. The training phase needs a subjects database – each person should have a different folder (unique folder name – ID) with more face pictures in different rotation and lighting conditions. The recognition module will return a list of IDs for the recognized persons, IDs used to extract information from the MySQL database.

3. **Application module** will be developed using Java Micro Edition for platform independence. The user will be able to choose server's address and the number of person candidates for each detected face. Using a file browser, the user will select a face image and send it to the server waiting for results. Finally, the application will display the detected faces and displays the count.

4. **Capturing module:** will perform a face recognition. It will integrate
   detection, recognition and database modules. Running in training mode, the software will display an interactive console asking for person's name, age, telephone, occupation. After that, it will detect and capture faces from webcam. The faces are further processed to reduce recognition sensitivity and stored into the corresponding person's folder from the training database. In recognition mode, the application will detect faces, recognize them and displays person's count.

The face detection and recognition modules are written in C++ using Object Oriented style to provide more clarity and re-usability. OpenCV it is used for implementing face detection and recognition algorithms and for advanced image processing techniques. More information on the implementation of the project can be found on the wiki page of each page.

4.1 TECHNOLOGIES USED

This Face emotion detection is implemented using Python technology. Python language is one of the most flexible languages and can be used for various purposes. It is one of the best languages which is used to implement machine learning techniques as it contains a lots of special libraries for machine learning namely scipy and numpy that can be used for linear algebra and getting to know kernel methods of machine learning. The language is considerable to use when waged with machine learning algorithms and has uncomplicated syntax relatively.

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace.

The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. Due to concern about the amount of code written for Python 2, support for Python 2.7 (the last release in the 2.x series) was extended to 2020. Language developer Guido van Rossum shouldered sole responsibility for the project until July 2018 but now shares his leadership as a member of a five-person steering council.

The Python 2 language, i.e. Python 2.7.x, is "sunsetting" on January 1, 2020, and the Python team of volunteers will not fix security issues, or improve in other ways after that date. With the end-of-life, only Python 3.6.x and later, e.g. Python 3.8 which should be released in October 2019 (currently in beta), will be supported

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation.

A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming.

Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Python Tools

Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy. Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas Numpy intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more.

OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. Officially launched in 1999, the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time and 3D walls.

In the early days of OpenCV, the goals of the project were described as:

- Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.
- Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. A version 1.1 "pre-release" was released in October 2008.

The second major release of the OpenCV was in October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months] and development is now done by an independent Russian team supported by commercial corporations.

In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site. On May 2016, Intel signed an agreement to acquire Itseez, a leading developer of OpenCV. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Andriod 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. 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. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

PYCHARM (IDE)

Pycharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data Science with Anaconda.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition with extra features – released under a proprietary license.

PyCharm Features

- **Project Code Navigation -** Instantly navigate from one file to another, from method to its declaration or usages, and through classes hierarchy. Learn keyboard shortcuts to be even more productive
- **Code Analysis -** Take advantage of on-the-fly code syntax, error highlighting, intelligent inspections and one-click quick-fix suggestions to make code better
- **Python Refactoring -** Make project-wide code changes painlessly with rename, extract method/superclass, introduce field/variable/constant, move and pull up/push down refactorings
- **Web Development with Django -** Even more rapid Web development with Django framework backed up with excellent HTML, CSS and JavaScript editors. Also with CoffeeScript, Mako and Jinja2 support
- **Google App Engine Support -** Develop applications for Google App Engine and delegate routine deployment tasks to the IDE. Choose between Python 2.5 or 2.7 runtime
- **Version Control Integration -** Check in, check-out, view diffs, merge all in the unified VCS user interface for Mercurial, Subversion, Git, Perforce and other SCMs
- **Graphical Debugger -** Fine tune Python or Django applications and unit tests using a full-featured debugger with breakpoints, stepping, frames view, watches and evaluate expressions
- **Integrated Unit Testing -** Run a test file, a single test class, a method, or all tests in a folder.
- **Customizable & Extensible -** Bundled Textmate, NetBeans, Eclipse & Emacs keyboard schemes, and Vi/Vim emulation plugin

4.1 UML DESIGN

4.1.1 UML Diagram

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

4.2.1. Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

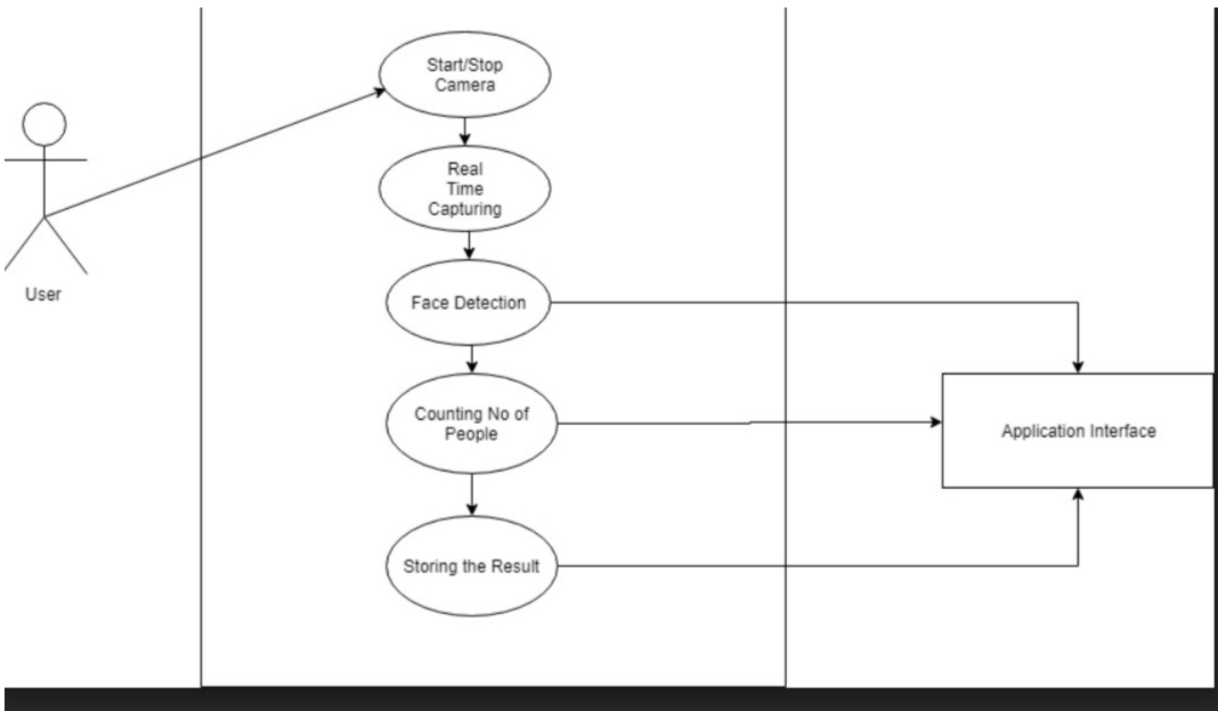

Figure 4.1.1 USE CASE DIAGRAM

Sequence diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

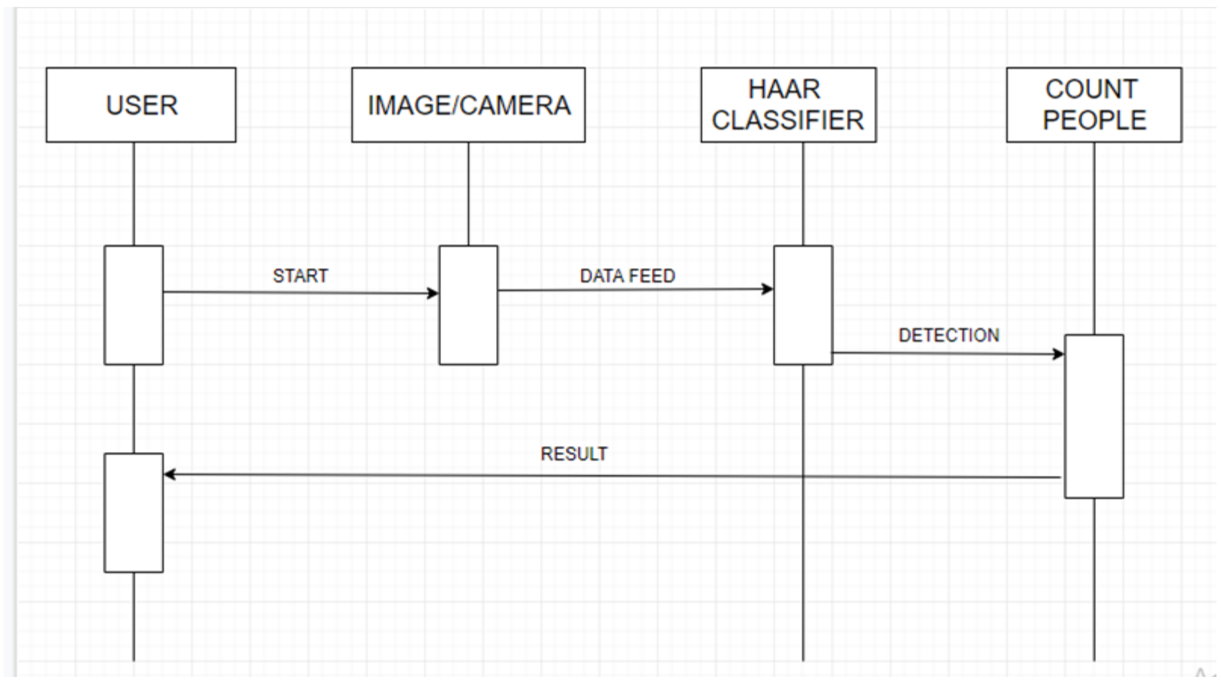

Figure 4.1.2 SEQUENCE DIAGRAM

Architecture design of the Project

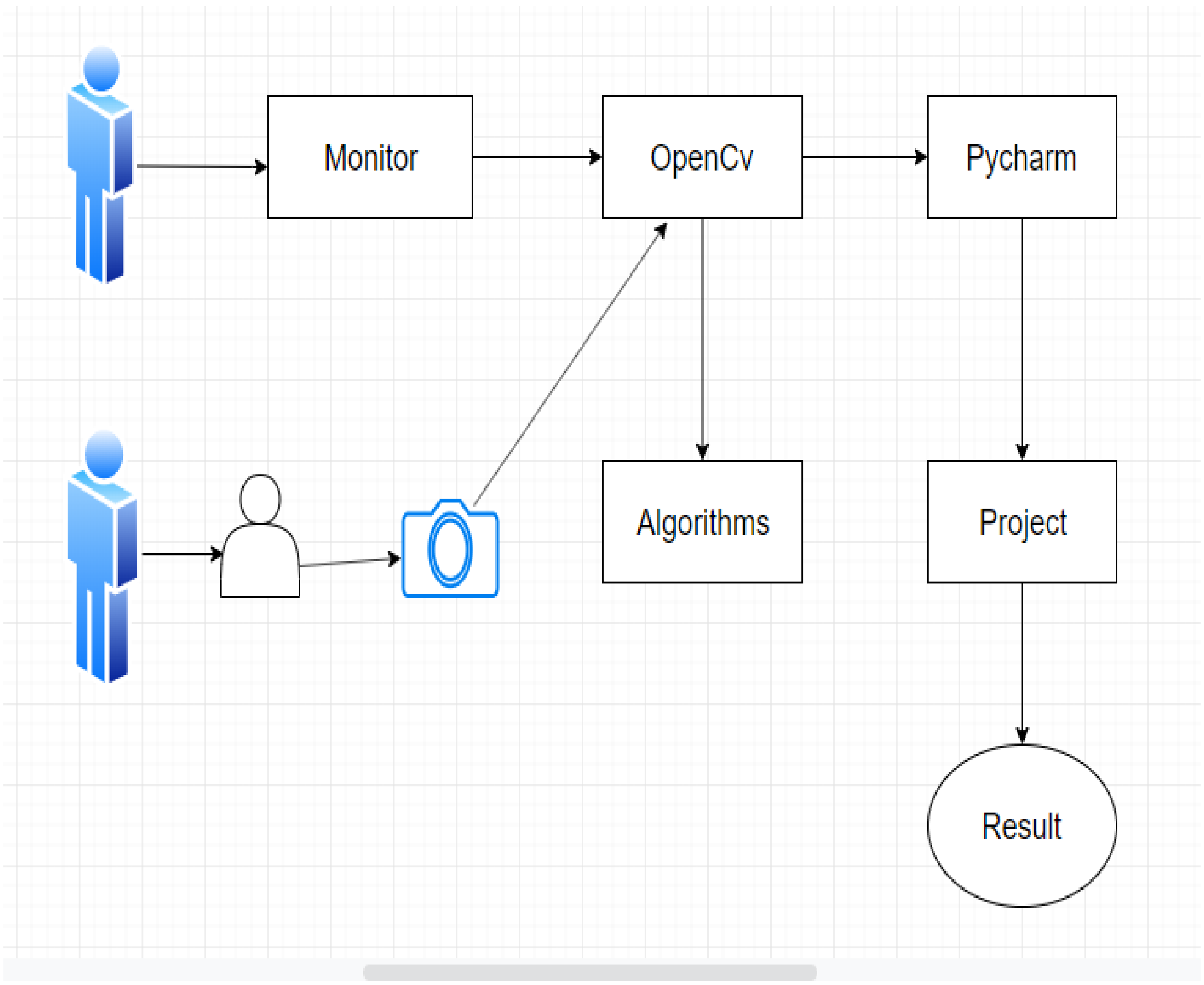

Figure: 4.1.3 Architecture design

4.2 ALGORITHM

HAAR classifier

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features proposed by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001.It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

The algorithm has fourstages:

1. HAAR Feature Selection
2. Creating Integral Images
3. Adaboost Training
4. Cascading Classifiers

It is well known for being able to detect faces and body parts in an image, but can be trained to identify almost any object. Lets take face detection as an example. Initially, the algorithm needs a lot of positive images of faces and negative images without faces to train the classifier. Then we need to extract features from it. First step is to collect the HAAR Features. A HAAR feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums.

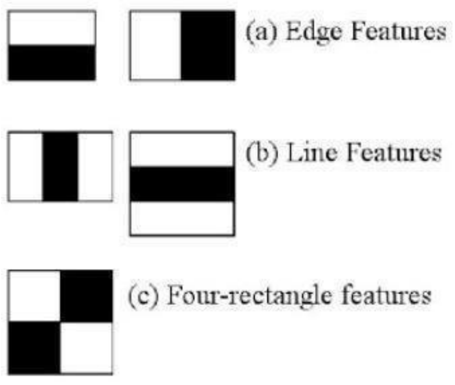

Fig. 4.3: Features of HAAR cascade classifier.

Integral Images are used to make this super fast.But among all these features we calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the propertythat the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant.

During the detection phase, a window of the target size is moved over the input image, and for each subsection of the image and Haar features are calculated. You can see this in action in the video below. This difference is then compared to a learned threshold that separates non-objects from objects. Because each Haar feature is only a "weak classifier" (its detection quality is slightly better than random guessing) a large number of Haar features are necessary to describe an object with sufficient accuracy and are therefore organized into cascade classifiers to form a strong classifier.

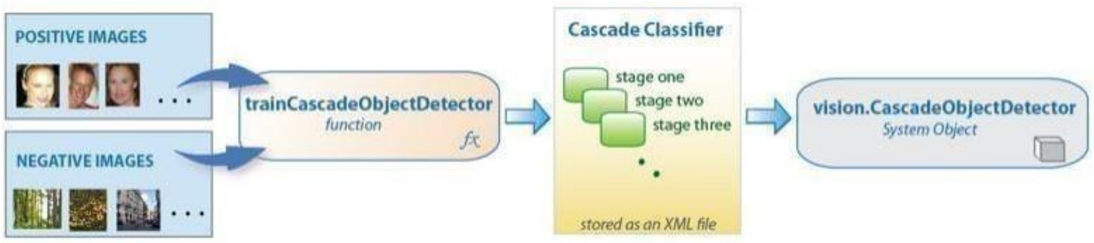

Fig. 4.4:Stages of Cascade classifier.

The cascade classifier consists of a collection of stages, where each stage is an ensemble of weak learners. The weak learners are simple classifiers called decision stumps. Each stage is trained using a technique called boosting. Boosting provides the ability to train a highly accurate classifier by taking a weighted average of the decisions made by the weak learners.

Each stage of the classifier labels the region defined by the current location of the sliding window as either positive or negative. Positive indicates that an object was found and negative indicates no objects were found. If the label is negative, the classification of this region is complete, and the detector slides the window to the next location. If the label is positive, the classifier passes the region to the next stage.

The detector reports an object found at the current window location when the final stage classifies the region as positive. The stages are designed to reject negative samples as fast as possible. The assumption is that the vast majority of windows do not contain the object of interest.

Machine Learning is a rapidly growing Haar Classifier is a supervised classifier, it has mainly been used for facial detection but it can also be trained to detect other objects. Computer vision is such a growing field that there are so many resources available for your own personal use. OpenCV provides a lot of functionality for machine learning techniques and the Haar Classifier is one of them.

Example rectangle features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the black rectangles. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature, and (D) a four-rectangle feature.

This technique harbors on the concept of using features rather than pixels directly. Which can then be easier to process and gain domain knowledge from correlating feature sets. It has also proven to be exceptionally faster.

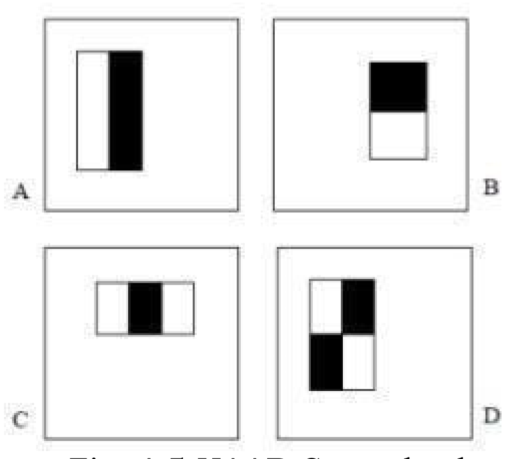

Fig. 4.5:HAAR Cascade classifier.

The two features are shown in the top row and then overlayed on a typical training cavity in the bottom row. The first feature measures the difference in intensity between the region of the top portion with the region right below. The feature capitalizes on the observation that the top region is often darker than lower parts of the cavity (a gradient). The second feature compares the intensities in the middle regions to the intensity across the outer edges of the cavity.

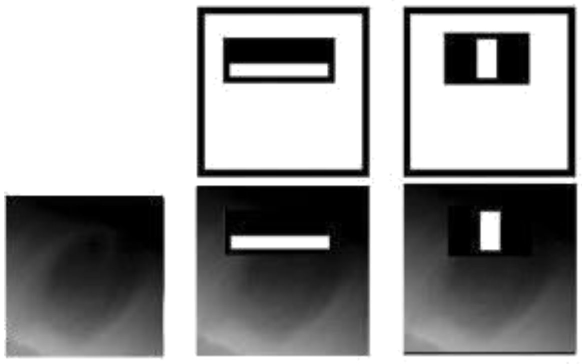

Fig. 4.6:HAAR Cascade

A Haar Classifier is really a cascade of boosted classifiers working with haar-like features. Haar-like features are specific adjacent rectangular regions at a specific location in a window (as shown in the first image above). The difference is then used to categorize subsections of an image and separates the non-objects from objects. Due to this distinction it is more of a weak learner, where you must use a large number positives to get as many Haar-like features as possible to accurately describe an object with some type of correlation and accuracy. So with all of these essentially weak classifiers we combine them into our classifier cascade to hopefully form a strong learner, by way of boosting.

Boosting is the concept of creating a set of weak learners and combining them to make a single strong learner. In the implementation of detecting solar cavities I used the AdaBoosting algorithm (aka Adaptive Boosting). This algorithm was initially proposed by Yoav Freund and Robert Schapire in A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting.

4.3 SAMPLE CODE

```
import numpy as np
import cv2
face_cascade                                                    =
cv2.CascadeClassifier('C:/Users/N/Desktop/haarcascade_fr
ontalface_default.xml')

image = cv2.imread('C:/Users/N/Desktop/test.jpg')
grayImage = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(grayImage)

print type(faces)

if len(faces) == 0:
    print "No faces found"

else:
    print faces
    print faces.shape
    print "Number of faces detected: " + str(faces.shape[0])

    for (x,y,w,h) in faces:
        cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),1)

    cv2.rectangle(image,      ((0,image.shape[0]      -25)),(270,
image.shape[0]), (255,255,255), -1)
    cv2.putText(image,   "Number   of   faces   detected:   "   +
str(faces.shape[0]),            (0,image.shape[0]            -10),
cv2.FONT_HERSHEY_TRIPLEX, 0.5,  (0,0,0), 1)

    cv2.imshow('Image with faces',image)
    cv2.waitKey(0)
     cv2.destroyAllWindows()
```

5.TEST CASES

Test Case Id	Test Case Name	Test Case Desc.	Test Steps			Test Case Priority	Status
			Input	Expected	Actual		
01	Output of the application	To test whether the correct Number of People are Detected or Not	10 People	10 People	10 People	Medium	Pass
			15 People	15 People	15 People	Medium	Pass
			1 People	1 People	1 People	Low	Pass
			9 People	9 People	9 People	Medium	Pass
		.	NO People	NO People	NO People	Medium	Pass
			30 People	30 People	17 People	High	Fail
			50 People And face objects	50 People	65 People	High	Fail

Table.5.1 Test Cases.

6.OUTPUT

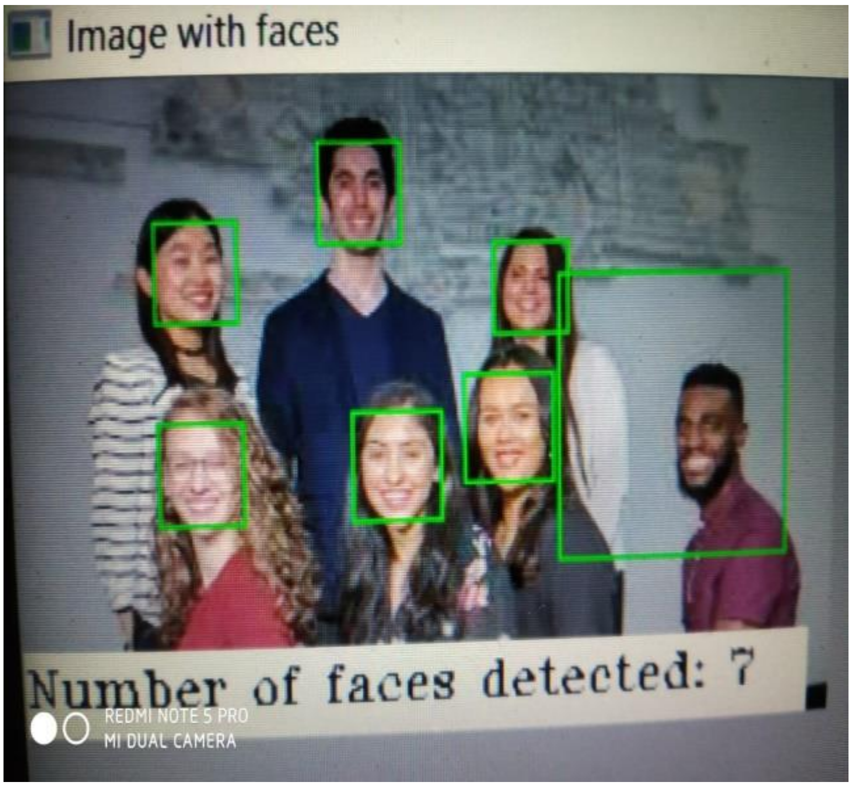

Fig.6.1:Output for 7 people.

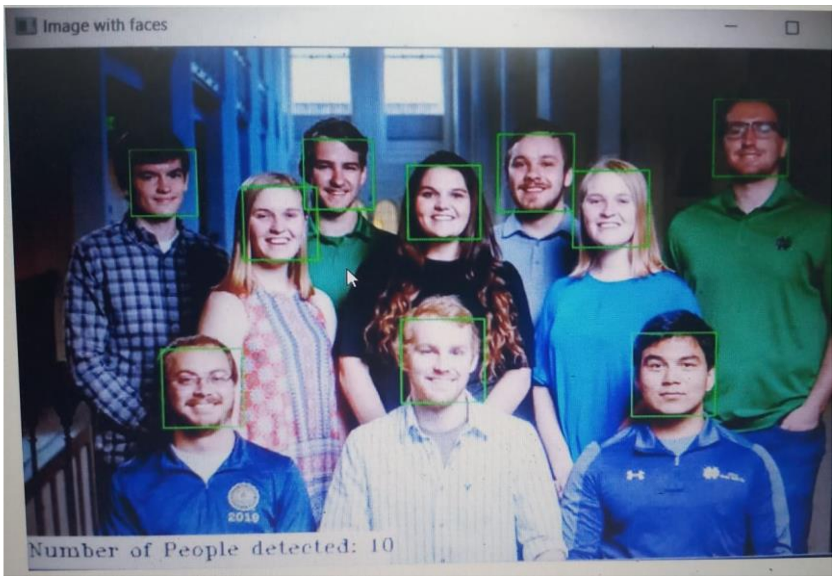

Fig.6.2:Output for 10 people.

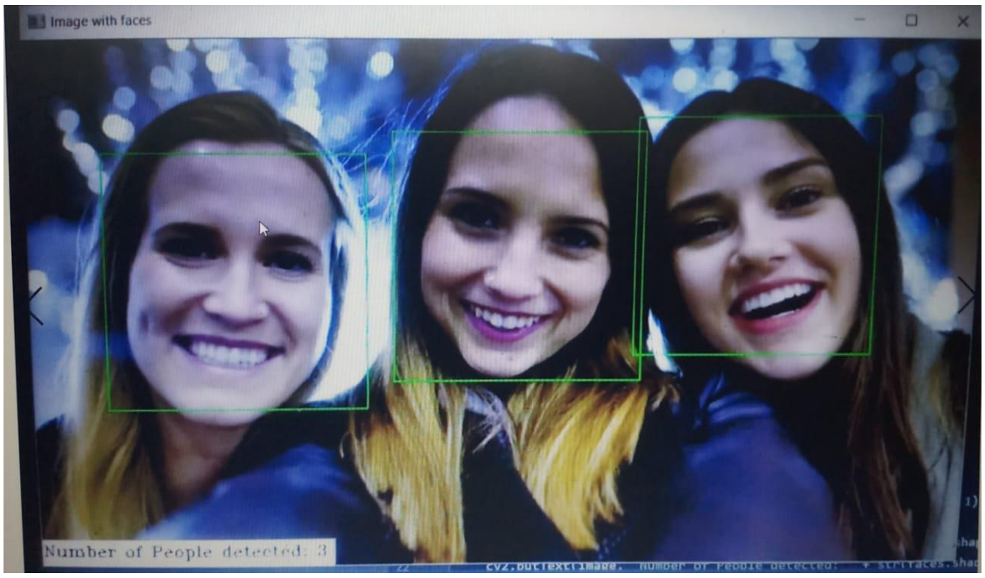

Fig.6.3:Output for 3 people.

In the above figure, i.e., Sample output is the window where the number of persons are being captured by the given image and the respective count is displayed. On detecting the face, respective count is shown on the left side of the screen. This count changes with the change in the image. Hence this real time application is very beneficial in various fields like Security and survilliance, computer science, linguistics, and related disciplines.

7.CONCLUSION

Proposed is a people counter using OpenCv, python to predict Number of the people and represent them using count. This real time application is very beneficial in various fields like Security and survialliancve, computer science, linguistics, and related disciplines. This research conducts an experimental study on recognizing facial detection and counting.. These include image acquisition, preprocessing of an image, face detection, feature extraction and then when the number of persons are classified the system assigns the user particular count according to his facial detection. Our system focuses on images taken from the user. The aim of this research is to develop automatic facial recognition system and count the number of persons over there.

8. FUTURE ENHANCEMENT

The fundamental extension is to add the dataset. Also the time taken to generate the result is very less. The accuracy of the project may be increased with the increase in the dataset. At present, the project is detecting Nearly 20-30 persons in an image. Robust spontaneous detection and counting can be increased with the help of machine learning and deep learning. This is going to have an impact on our day to day life by enhancing the way we interact with computers or in general, our surrounding living and work spaces. High correct recognition rate , significant performance improvements in our system. Promising results are obtained under face registration errors, fast processing time. System is fully automatic and has the capability to work with video feeds as well as images. It is able to recognize spontaneous Faces. Our system can be used in Digital Cameras where in the image is captured and provide the results for the number of people present in a particular region. In security systems which can identify a person and report the number of people .This project can be implemented in crowd analysis to detect the number of people coming to that region in a particular time. Police and security can use the system to understand the intensity of crowd in that region and analyize.

9. REFERENCES

[1]. About Pycharm IDE
https://www.jetbrains.com/pycharm/

[2]. OpenCV from
https://opencv.org/

[3]. People counter
https://www.pyimagesearch.com/2018/08/13/opencv-people-counter

[4]. Face detection and count
https://techtutorialsx.com/2017/05/02/python-opencv-face-detection-and-counting

[5] . HAAR Casscade feature from

https://docs.opencv.org/3.4/d7/d8b/tutorial_py_face_detection.html