

SMART SECURITY SYSTEM USING ARTIFICIAL INTELLIGENCE

Submitted in partial fulfilment of the requirements
for the award of
Bachelor of Engineering degree in Computer Science and
Engineering

by

VENKATA DURGA PHANIDHAR KURPATI(37110828)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119**

MARCH - 2021



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **VENKATA DURGA PHANIDHAR KURPATI (37110828)** who carried out the project entitled "**SMART SECURITY SYSTEM USING ARTIFICIAL INTELLIGENCE**" under my supervision from **August 2020 to March 2021**.

Internal Guide

Dr. Mercy Paul Selvan, M.E., Ph.D.,

Head of the Department

Dr. S. Vigneswari, M.E., Ph.D.,

Dr. L. Lakshmanan, M.E., Ph.D.,

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I **VENKATA DURGA PHANIDHAR KURAPTI** hereby declare that the Project Report for "**SMART SECURITY SYSTEM USING ARTIFICIAL INTELLIGENCE**"

is

done by me under the guidance of **Dr. Mercy Paul Selvan, M.E., Ph.D.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean, School of Computing, Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. Mercy Paul Selvan, M.E., Ph.D., Assistant Professor**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

A smart security system is an artificial intelligence-based system that is designed to avoid incidents caused by human error. Buildings with lower wall heights in balconies are becoming more popular, as seen in today's pattern. This may be dangerous, particularly for children, the mentally ill, or the elderly, depending on their conditions. These conditions can be avoided by implementing a home security device. This system includes various modules such as photo capture, face recognition, distance measurement, and speak modules, which allow us to detect when someone is approaching the balconies and avoid or mitigate accidents.

TABLE OF CONTENT

ABSTRACT

LIST OF FIGURES

S.NO.	TITLE	PAGE NO.
1	INTRODUCTION	
	1.1 INTRODUCTION	1-3
	1.2 INTRODUCTION TO ARTIFICIAL INTELLIGENCE	3-4
	1.3 ROLE OF AI IN SMART SECURITY SYSTEM	5
	1.4 INTRODUCTION TO CONVOLUTION	5-8
	NEURAL NETWORK	
	1.5 INTRODUCTION TO PHOTO CAPTURE	9-10
	1.6 INTRODUCTION TO DISTANCE CALCULATION	10
2	AIM AND SCOPE	
	2.1 AIM	11
	2.2 SCOPE	11
3	IMPLEMENTATION	
	3.1 EXISTING SYSTEM	12-13
	3.2 SYSTEM FUNCTIONALITIES	13-14
	3.3 PROJECT IMPLEMENTATION	15-16
	3.4 MODULE DESCRIPTION	16-19
4	SOFTWARE DEVELOPMENT METHODOLOGY	
	4.1 DESCRIPTION OF DIAGRAM	20-21
	4.2 ARCHITECTURE DIAGRAM	21
	4.3 SDLC USED	22-24
5	RESULTS AND DISCUSSION	25-28

REFERENCES

APPENDIX

SOURCE CODE

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
1.1	LAYERS OF CNN	6
1.2	POOLING	7
1.3	FULL CONNECTION	8
1.4	PHOTO CAPTURE	9
1.5	DISTANCE CALCULATION	10
4.1	FLOW CHART	20
4.2	ARCHITECTURE DIAGRAM	21
4.3	AGILE MODEL	22
5.1	PHOTO CAPTURE OUTPUT	25
5.2	CNN TRAINING	26
5.3	CNN PREDICTION	27
5.4	DISTANCE DATASET GRAPH	27
5.5	DISTANCE DATASET	27
5.6	SPEAK MODULE	28

CHAPTER 1

INTRODUCTION

N

INTRODUCTION

In the 21st century with the advancement in technology, there is tremendous improvement in the height of the building. This change can be considered as positive till certain extent but still there are certain issues which are being faced with the rise in the buildings. This improvement is positive only if they are constructed with accordance to the laws, but the buildings which are being raised are constructed as per people convenience and which is leading to the cause of accidents

These accidents are caused mainly due to the human negligence and also building the buildings without proper documented approval. This negligence may affect not only children but also it affects all the people. Generally, it will be affecting kids and mentally retarded people as they donot have the proper control over them.

Falls account for significant morbidity and mortality among young children, and are one of the most common reasons for emergency department visits. Since the early 1970s, falls from buildings have been recognized as an important cause of injuries and deaths, and a program in New York City that installed window guards in tall apartment buildings led to a dramatic decline in childhood deaths from building falls. However, in many other parts of the United States, falls from windows and balconies of apartments are not as likely to result in death, possibly because apartments are less likely to be as tall in other parts of the country.

Nevertheless, these fall related injuries might result in substantial morbidity, if not mortality. In addition, we could find no recent studies that included on-site visits and review of pertinent environmental aspects that may contribute to such a fall. We attempted to determine the circumstances around childhood falls from apartments in the Dallas County area, including inspection of the pertinent environment in order to identify whether interventions may be developed to

prevent these injuries.

Let us consider an scenario, a nuclear family consisting of father mother and baby. As we know kids gets attracted to the things easily, so they will be moving towards the places or handling the things which are open. As we know in cities most of the people lives in apartment culture where it has balcony even if it is in top floor. Kids

generally move towards the open areas, but in this situation if this kids goes towards the open area in the flats and they don't have any control over the actions done by them.

As it is an open area it cannot be close as if they close the balconies the total ventilation and light gets closed which is not a good environment to survive. So, in this situation people leave their children in one area and proceeds with their work. As children don't have the idea over the things they move towards the balconies and sometimes hangs to the grills which may lead to the falling down from the building which may even lead to the chance of losing life of a child or the child may get injured severely.

Consider the same family who lives in the same condition with a mentally retarded person in the family. As we know that the mentally retarded person will not have any control over the actions that are being done by the person. As the person is mentally retarded the person should be under another person supervision but there can be a situation where the person might be moving aside with some work and this might become a danger task that the mentally retarded person moves towards the open balcony which might cause accidents in the way the person might fall from the top of the building.

To overcome the accidents that occur due to negligence or some other issue we can install the house with a smart alerting system which is also called as Smart Navigation System which has a trained set of images of one person in the house who has to be restricted from the movement towards the balcony and this will also calculates the distance between the trained object and border line probably called as Line of Control which is referred as border in general communication and alerts the rest of the members in the house by announcing with prompt message set.

This security system is installed using a webcam and the system is to be connected to a speaker in such a way that when any object is moving towards the LOC it will start capturing the object and sends the image of the objet to the next step where it will be classifying the image as trained or not trained and if it is a trained object moving towards the LOC, the system calculates the distance

between the object and the LOC and when the object is getting too close then the system alerts with the message of through the speaker.

The above system can be implemented using Artificial Intelligence Technology. This system is implemented using four algorithms.

1. Photo capture
2. Convolution Neural Network
3. Regression
4. Speak

Introduction to Artificial Intelligence

In computer science, artificial intelligence (AI), sometimes called machine intelligence, is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans.

Colloquially, the term "artificial intelligence" is often used to describe machines (or computers) that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving".

As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect. A quip in Tesler's Theorem says "AI is whatever hasn't been done yet." For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Modern machine capabilities generally classified as AI include successfully understanding human speech, competing at the highest level in strategic game systems (such as chess and Go), autonomously operating cars, intelligent routing in content delivery networks, and military simulations. Artificial intelligence can be classified into three different types of systems: analytical, human-inspired, and humanized artificial intelligence.

Analytical AI has only characteristics consistent with cognitive intelligence; generating a cognitive representation of the world and using learning based on past experience to inform future decisions. Human-inspired AI has elements from cognitive and emotional intelligence; understanding human emotions, in addition to cognitive elements, and considering them in their decision making. Humanized AI shows characteristics of all types of competencies (i.e., cognitive, emotional, and social intelligence), is able to be self-conscious and is self-aware in

interactions.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. For most of its history, AI research has been divided into subfields that often fail to communicate with each other.

These sub-fields are based on technical considerations, such as particular goals or deep philosophical differences. Subfields have also been based on social factors (particular institutions or the work of particular researchers). The traditional problems of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence is among the field's long-term goals. Approaches include statistical methods, computational intelligence, and traditional symbolic AI.

Many tools are used in AI, including versions of search and mathematical optimization, artificial neural networks, and methods based on statistics, probability and economics.

The AI field draws upon computer science, information engineering, mathematics, psychology, linguistics, philosophy, and many other fields.

The field was founded on the claim that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the nature of the mind and the ethics of creating artificial beings endowed with human-like intelligence which are issues that have been explored by myth, fiction and philosophy since antiquity.

Some people also consider AI to be a danger to humanity if it progresses unabated. Others believe that AI, unlike previous technological revolutions, will create a risk of mass unemployment.

In the twenty-first century, AI techniques have experienced a resurgence following concurrent advances in computer power, large amounts of data, and theoretical understanding; and AI techniques have become an essential part of the

technology industry, helping to solve many challenging problems in computer science, software engineering and operations research.

ROLE OF AI IN SMART SECURITY SYSTEM

The Smart security system is an alerting system which is can be implemented using artificial intelligence. This can be implemented using the following modules in the following method.

Firstly, we need to implement the photo capture module. This module is implemented using the opencv library.

Next to the photo capture module there is a module called classification module where the classification is done using the CNN algorithm which implements the libraries called tensorflow, keras. This module is used to classify whether the object is known or unknown.

Immediate to this there comes the distance calculation module which implements the opencv module and finds the distance between the object and the boundary by implementing the values obtained here in a regression part.

Last part of the system is the alerting system which is used to alert people with the message given through the speaker.

INTRODUCTION TO CONVOLUTION NEURAL NETWORK

Convolutional Neural Network (CNN) is a deep learning network used for classifying images. The basic premise behind CNN is using predefined convolving filters to identify patterns in image edges, parts of objects and the build on to this knowledge to detect complete objects like animals, human being, automobiles etc.

Input Layer:

This layer is responsible for resizing input image to a fixed size and normalize pixel intensity values.

Convolution Layer: Image convolution is process of convolving a small 3x3, 5x5 matrix called kernel with image and subsample the image. This is used for extracting a specific feature like edges in the image. There are several such filters used for specific purpose.

Pooling Layer: Pooling layers perform max or average operation on output of convolution layers to retain most significant information about patterns and forget insignificant ones

Fully Connected Layer: These are final logistic regression layers which maps visual features to desired output functions

Output Layer: This layer contains probabilities of classes of for each image input. For instance, if the image is cat. Output would be [0.85, 0.15] where 0.85 represents probability of image being cat and 0.15 represents probability of being a dog. Convolutional Neural Networks(CNN) are one of the popular Deep Artificial Neural Networks. CNNs are made up of learnable weights and biases.

CNNs are very similar to ordinary neural networks but not exactly same. CNNs are primarily used in image recognition, image clustering and classification, object detection

etc..

CNNs use relatively less pre-processing when compared with other algorithms of image processing. The connectivity pattern of the CNN resembles the structure of animal visual cortex.

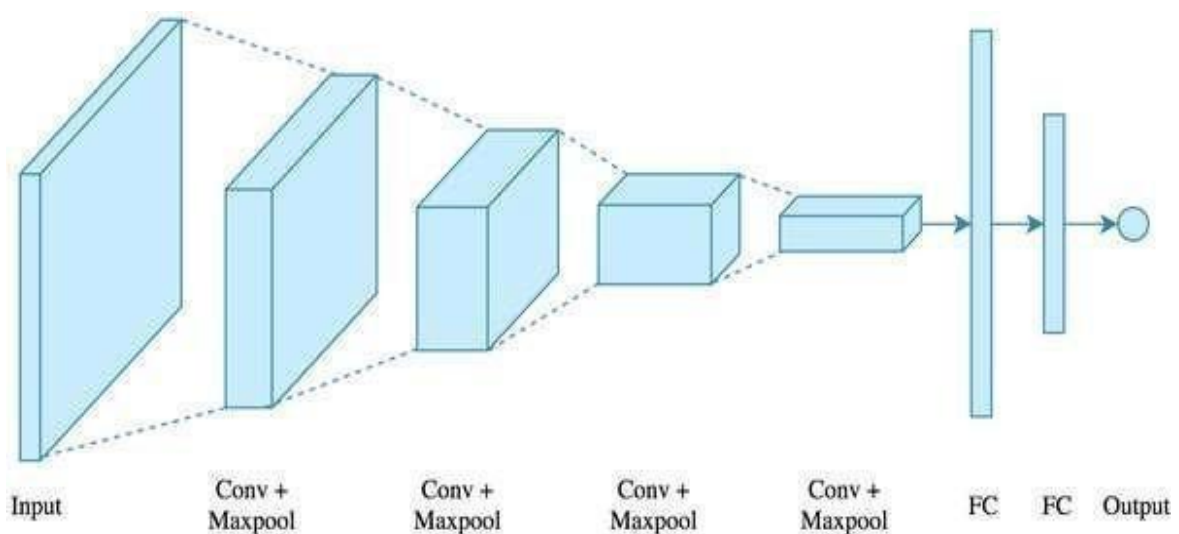


Fig 1.1

Cnn consist of different layers. They are input layer and output layer. Between these layers there are some multiple hidden layers. There is no limitation for hidden layers

present in the network. Input layer takes the input and train specifically and gives output from the output layer. With the help of CNN, we can use the large amount of data with more effectively and accurately.

Layers of CNN

Convolution:

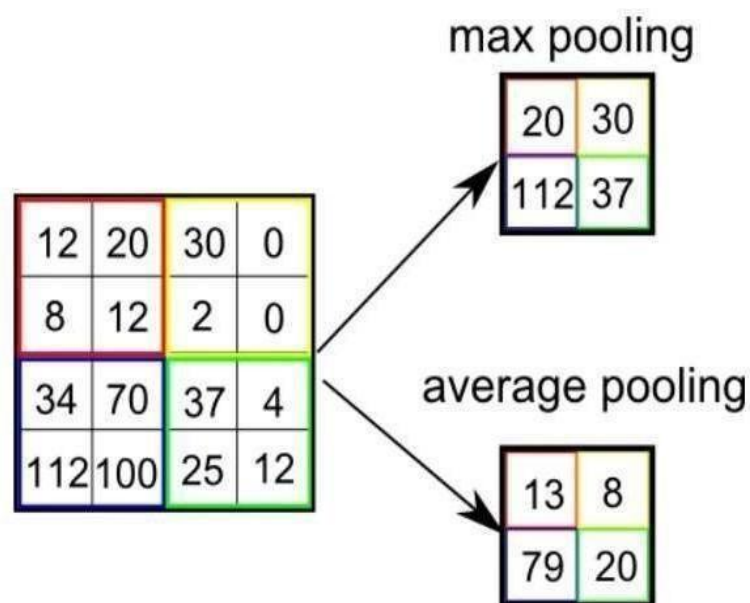
Convolution means rolling together. Convolution layer applies a convolution operation to the input for passing the result to the next layer. Each convolution processes the data only for its respective field.

Activation Function

It is used to determine the output of a neural network (like yes/no). It can also be attached between two different neural networks. The popular activation functions are RELU, softmax etc...

Pooling:

It is responsible for reducing the spatial size of the convoluted feature. Pooling combines the output of neuron cluster at one layer into a single neuron in the next layer.

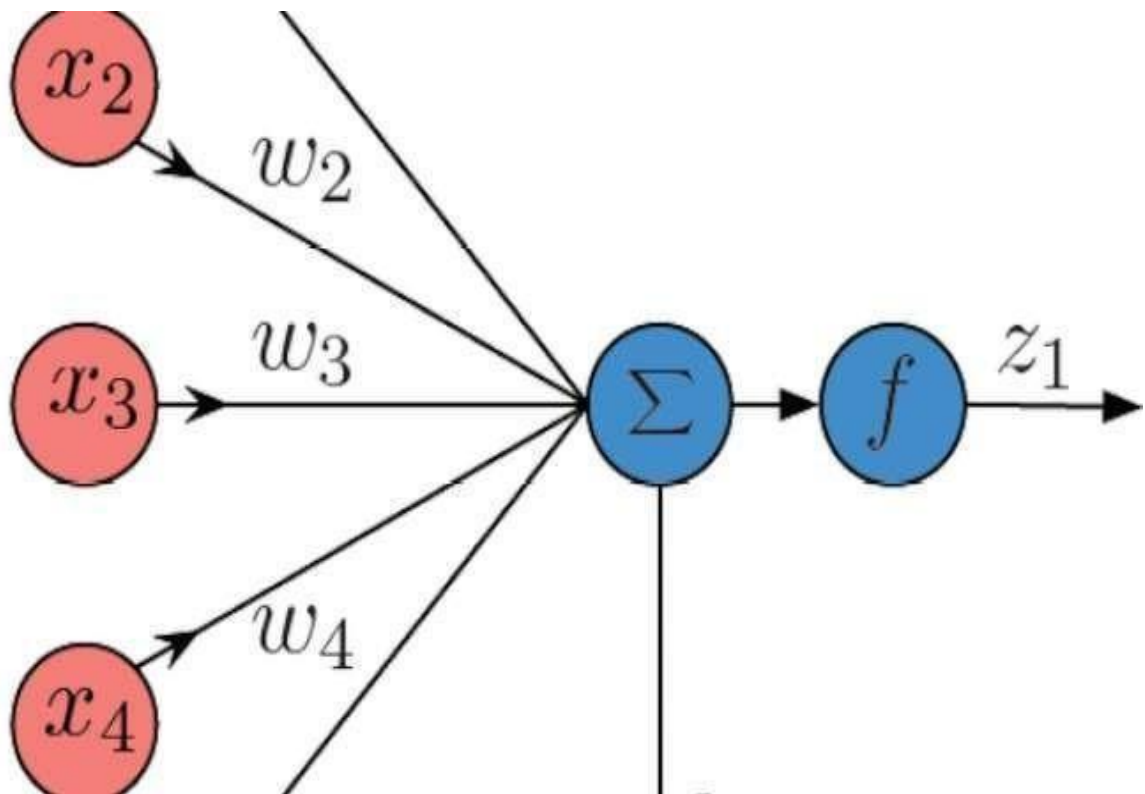


Max pooling uses the maximum value from each cluster of neurons at the prior layer. Average pooling uses the average value from clusters of neurons at the prior layer. Fully Connected Layer:

It connects every neuron in one layer to every neuron in another layer. It is similar to Multi-Layer-Perceptron Neural Network(MLP).

Applications of CNN

- Image Classification
- Object Detection
- Handwritten digit recognition
- Optical Character Recognition
- Emotion Detection
- Feature extraction



Introduction to Photo capture

Python provides various libraries for image and video processing. One of them is OpenCV. OpenCV is a vast library that helps in providing various functions for image and video operations. With OpenCV, we can capture a video from the camera. It lets you create a video capture object which is helpful to capture videos through webcam and then you may perform desired operations on that video.

Computer vision is a process by which we can understand the images and videos how they are stored and how we can manipulate and retrieve data from them. Computer Vision is the base or mostly used for Artificial Intelligence. Computer-Vision is playing a major role in self-driving cars, robotics as well as in photo correction apps

OpenCV is the huge open-source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human. When it integrated with various libraries, such as Numpy, python is capable of processing the OpenCV array structure for analysis. To Identify image patterns and its various features we use vector space and perform mathematical operations on these features.



Fig 1.4

Applications of OpenCV:

There are lots of applications which are solved using OpenCV, some of them are listed below

face recognition

Automated inspection and surveillance

number of people – count (foot traffic in a mall, etc)

Vehicle counting on highways along with their speeds

OpenCV Functionalities:

Image/video I/O, processing, display

Object/feature detection

Introduction to Distance Calculation

You have to know the approximate (x, y) dimensions of the object that you want to detect. For example, if it is a frontal face, you have an idea of the approximate dimensions it has. If it is a full body view of a person, there are no "approximate" dimensions possible because the (x, y) size will change drastically when the person is standing vs sitting. So you should first determine if your object is deformable or rigid. If it is rigid, then proceed.

You now need to perform camera calibration, to determine the focal length 'f' of the camera. The focal length is in terms of pixel distances here.

After detecting an instance of the object using cascade classifier, measure the length (or breath) of the bounding box (in pixels).

Use the pinhole camera equation to determine depth .

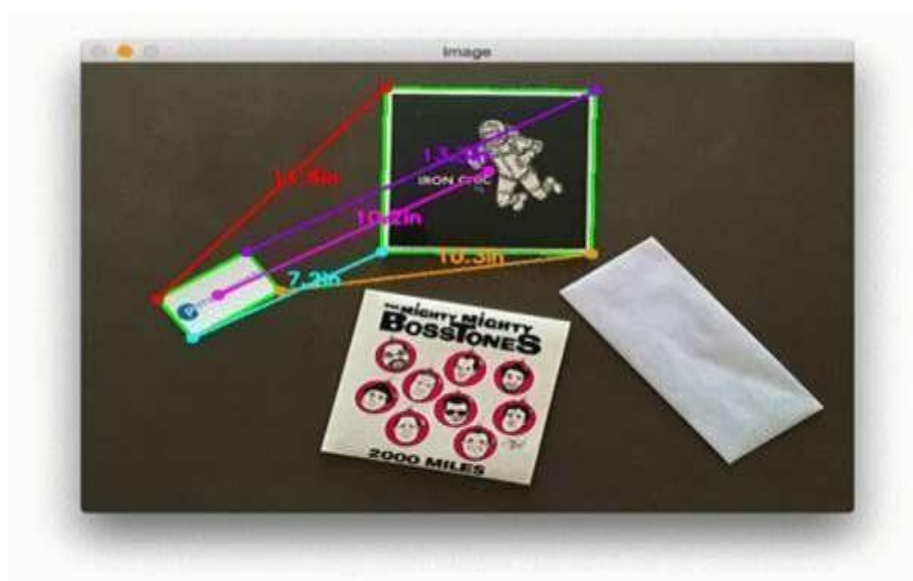


Fig 1.5

Chapter 2

Aim and Scope

Aim

The aim of this proposed smart security camera is to minimise the number of injuries and save the lives of schoolkids and mentally disturbed persons in situations such as when they approach high-rise building balconies.

Scope

This model will aid in reducing manual emphasis on guarding a specific topic, such as an individual or a kind. so that time may be spent on more critical activities rather than focusing on the wellbeing of a certain individual It would make it easier for the guardian to act in such a manner that his intervention is only needed when the warning system signals that the subject has entered a dangerous environment.

CHAPTER 3

IMPLEMENTATION

Existing System:

The existing system is something that deals with independent implementation of the each, module. The existing system works in the following way:

Photo capture system is a system that is used to keep track of the number of people coming towards the in gate or moving away from the out gate. As we see there is a buffer for the storage of memory. There is no option called memory refresh in the existing method. The existing system runs continuously as the code runs in an infinite loop. But the system is updated in such a way that it gets terminated when an it crosses the certain time bound. The existing model doesn't have the feature of saving the data using file concept and taking last image as record.

In the classification system though it uses the system that deals with compare and contrast of features, it has to get input manually from user hardcoded. The input is a static input i.e., the user has to change the image continuously in code. It cannot take the input from the previous module as it has to get input manually. It doesn't have an option to speak out the classified data or compute distance between the classified an unclassified data.

Distance computation module deals with the calculating distance between two objects. But this system doesn't have an option to set line of control i.e, the boundary line between the object and the restricted area. This system cannot implement the alerting system to prompt the messages given by the user. The existing system is made ready using a dataset with combination of two labels that consist of gap and w length. Based on the value of w the gap is calculated using measuring scale. This values are recorded and stored as data set and this data set is used in regression code to compute the distance.

Alerting module is used to express or alert the people in the house by connecting to the speaker. This system generally used to prompt according to message given by the user. But the system doesn't have any auto trigger option.

As we see in our day to day life, we use all the modules independently and we find it difficult to use the modules indepently for the required purpose.

Also as we there is no buffer set refresh the memory in phot capture which leads to excessive use of space for the given system.

The existing system does not have for photo capture does not terminates itself which will lead to capture n number of images and this will cause a difficulty in storing the images in the storage.

SYSTEM FUNCTIONALITIES

This module is the first and foremost step of this project. It deals with capturing the object when the object comes in to the range of the camera. This module is carried out using openCV library. When an object comes in to the focal range the camera starts clicking the images of the object continuously with a certain interval of time. The images clicked are stored in buffer memory. The image that is clicked at the last will be sent as input to the next module i.e, Object Classification. In this module as there are mare number of pictures being clicked, the buffer is set with a capacity. If the camera captures more than the limit of the buffer capacity, it clears the memory in the next upcoming operation by checking the number images present in the memory.

This module is the second module. This module is based on an algorithm called Convolution Neural Network, here the module deals with the classification of objects from images. So this module consists of two parts, first part deals with the training of the images from the given dataset. As we are training the system with the images the dataset must consist of two parts one is the training set and the other is the testing set, and the images must divide in such a way that training part gets more number of images. The first part is operated only when a dataset is added or changed. In this part of the module the system extracts the various features of the object and neglects the useless content from the image by undergoing four steps namely convolution, maxpooling, flattening and full connection, As it extracts various features in the process it stores all the extracted features in a file called model.h5. In the second part, it takes the last captures image as the input from the previous module and undergoes the above four steps to extract the features. The extracted featured are crosschecked with features saved as model. If the features match with the extracted feature, then it will enable the access to the next module. If the feature doesn't match with the

extracted features, then no action will be taken.

The next module is distance calculation module. This module is enabled when the features of the object gets matched with model feature. So this module waits for the instruction from the previous module. This module is carried out using the regression concept, as we need to calculate the distance we need to prepare a dataset in .csv format. The dataset will be implemented in the regression code and it is linked to a camera code to find the distance between the object and the camera. In this module there is an option enabled to set the Line of Control(LOC). LOC can be expressed as the digital boundary to a live area. LOC is fixed at point set by the user. When an object comes close to the LOC then it invokes the next module called speak module.

This is the last module of this project. This code triggers when the object moves close to the LOC. When a classified object moves close to the LOC then the alert system will announce the prompt message given by the user. This module is developed using pyttsx3 library. This module requires a speaker to connected to prompt the message given by the user.

The whole system is integrated in a way such that when an object enters the room it follows certain steps to prevent the object from accidents.

First, the object capture module is enabled when an object comes in to the focal range of a camera.

Once the object gets captured in the object capture module, it is sent as an input to the second module called object classification.

As object classification module is a pre-trained model with the dataset it extracts set of features from the input image and compares with the model created from trained data.

If the input image of the object matches with any of the trained data, then it will trigger the next module to compute distance between the object and line of control.

As the object is identified, this module calculates the distance between the object and the LOC.

If the identified object is getting close to the boundary or line of control, then it triggers the next module called speak.

This is the last step where the system prompts the certain message given by the user to alert the rest of the people in the house.

Project Implementation:

The project smart security system can be implemented by using the following steps:

1. The system has to be setup in prior in such a way that when an object comes in to the focal range of the camera it must start capturing the images.
2. When an object enters the focal range of the camera the first module i.e., photo capture enables.
3. In this step, the system first checks the buffer memory where the images are going to be stored. If the number of images in buffer are more than the assigned capacity, then the buffer gets refreshed by deleting the data.
4. After deleting the images, now camera starts detecting the objects. As the object gets detected, it starts capturing the images till certain interval of time.
5. The last image captured before finishing the time interval is sent as the input to the next step.
6. Once the image gets captured, it will be used as an input in the classification phase.
7. The classification phase consists of two methods. One is testing and the other is training.
8. The classification of the image is carried out by a neural network algorithm known as CNN (Convolution Neural Network).
9. The CNN algorithm has to be trained with the set of images which is known as the dataset.
10. Dataset is a collection of images that are required by the system to train the model.
11. The entire dataset of images is to be divided in to two parts namely training set and testing set.
12. The images are distributed among the both sets in the ratio 7:3 i.e., out of every 100 images 70 images will be allotted for training purpose while the rest 30 images will be allotted for the testing purpose to create an efficient model for classification.
13. In execution of CNN algorithm, the system has to undergo training phase.
14. In training step, the images are to initialised to a model.
15. After initialising the model, the system will be added with the four major layers

of CNN algorithm i.e., convolution, maxpooling, flattening, and full connection.

16. After adding the layers, it will be sent to the full connection layer.

17. Now the user has to define the number of epoch i.e., the number of phases in which the dataset of the images has to get trained.
18. After declaring the epochs, the user has to define the number of steps to be involved in each epoch.
19. Once the epochs are set all the images in the dataset undergoes all the four layers for the creation of the model.
20. This feature of creating model is called training.
21. Next comes testing where it takes input from the previous module and undergoes all the four layers for feature extraction.
22. All the extracted features are compared with the featured of model.
23. If all the features of input model features matches with the trained features, it shows the prompt message as prediction.
24. Else it shows the message as unpredicted.
25. Once the object is detected, then it starts calculating distance where it is done using the regression code.
26. The regression code is trained with the set of values to compute the distance between the object and line of control.
27. Once the object starts moving close to the LOC, it starts calculating the distance.
28. If the object is at LOC, it will start alerting unless or until the object moves back.
29. If the object moves back, it will stop alerting.
30. Alerting system module is developed using the pyttsx library where we can set an alert message and integrate it with the speaker.
31. In this way the system gets implemented.

Module Description:

The Smart Security System can be implemented by the following modules:

1. Photo Capture
 2. Object Classification
 3. Distance Calculation
 4. Alerting System
1. Photo Capture:

This is the first and foremost section of the smart security system. This module deals with capturing the images of the object when an object enters the focal

range camera. As this module is used to capture the image of an object, this module involves the implementation of a camera. The camera needs to be integrated in such a

way that when an object enters the focal range of the camera it continuously captures the image of the object.

This module imports the libraries called opencv, time and os. This module detects the objects using the predefined file called haarcascade which is a file saved with .xml extension. This file is written in such a manner that it has features of the object to be detected.

To run this module, the following steps are to be followed:

1. First, implement the required libraries.
2. Load the cascades required to detect the faces.
3. Define the detect function where the function allows detect the face using the stored cascade.
4. Enable the camera to perform the face recognition.
5. Set the time to run the code.

When an object enters the focal range of the camera it starts capturing the images of an object. As it is observed that the code captures n number of images when it works, it is enabled with a feature to delete the old entries when the code starts its cycle.

This code is set to run to a particular time in such a way that the image captured at the end is stored in such a way that the last image captured gets stored in a notepad using the file concept, and this image is sent as input to the next module. This module is enabled with buffer refresh option i.e, the camera will be refreshing its buffer by checking the number of items present in the buffer at the time of code starting. If the buffer has more than the limit it removes all the images in the buffer and goes for capturing the new images.

2. Object classification:

This is the next step in the execution of the system. This module is carried out by using the algorithm called Convolution Neural Network, where it involves two phases one is the training phase and the other is the testing phase.

In this module the user has to train the system with images of the object to be under supervision. The training is to be done in such a way that the entire dataset is to be divided in the ratio 7:3, where the 70% of the data is set for training purpose and the rest 30% of data is set to testing purpose.

In training phase the system extracts various features from the given image data set and store them in a single model and the model has to be stored with the extension

called .h5. As we use various folders to run the code all the files and folders are to be stored in a single location from the place where the code is getting executed.

Here in this module, the code takes input from the previous module i.e, the image captured at the end and the captured image undergoes testing where it extracts the feature from the input image and compares with the model and finally classifies whether the given image is known or unknown.

As CNN involves extracting of features the code undergoes the following four steps in execution of the code. They are:

1. Convolution
2. Maxpooling
3. Flattening
4. Full Connection

1. Convolution:

In this step the system extracts the various features from the given image dataset i.e., it convolutes the various features of the image.

2. Maxpooling:

Pooling can be explained as the removal of the unwanted data from the image. Here maxpooling is something that it removes around 75% of the unwanted data from the image. Generally, the unwanted data in the image is referred as the background of the object in the image.

3. Flattening:

After the image gets maxpooled there will be excess cells in the image matrix. So by using the flattening technique, image matrix is converted to the size required as per the data. Simply it can be said as conversion of multi-dimensional matrix to a single dimensional matrix.

4. Full connection:

After the image gets flattened, there are various single dimensional matrices which are being connected to form a model which will be used to compare with the input. The steps involved in executing cnn algorithm are as follows:

Intermediate Code:

1. Import the required libraries such as keras and the properties required in keras.
2. Initialise a classifier to classify them in the order.

3. Add the convolution layer.
4. Add the pooling layer.

5. Add the flattening layer.
6. Add the full connection layer.
7. After adding the layer compile the layers in to a model.
8. Fit the images using the train and test dataset.
9. Set the number of epochs and number of steps per each epoch. Prediction Code:

1. Import the required libraries.
2. Load the model.
3. Define detect function.
4. Check with the prediction and classify the object.
3. Distance Calculation:

This the third step of this system, where the system computes the distance of the classified object. The distance is computed between the object and the boundary which is technically termed as Line of Control. In this module the system enables camera to calculate the identified object distance and also this module enables us to set the boundary to the object.

This module implements using two functions. They are:

1. Object detection
2. Regression

Object detection using open cv has special feature to calculate distance, here in this module there is a parameter which has to be implemented in dataset with distance so that it will be easy to calculate the distance. The parameter 'w' is used to calculate the distance.

Regression is the code that gets trained with the dataset and calculates the distance.

4. Alerting system:

This code helps to speak out the prompt message given and this is to be integrated with the above distance computation module.

CHAPTER 4

SOFTWARE DEVELOPMENT METHODOLOGY

DESCRIPTION OF DIAGRAM

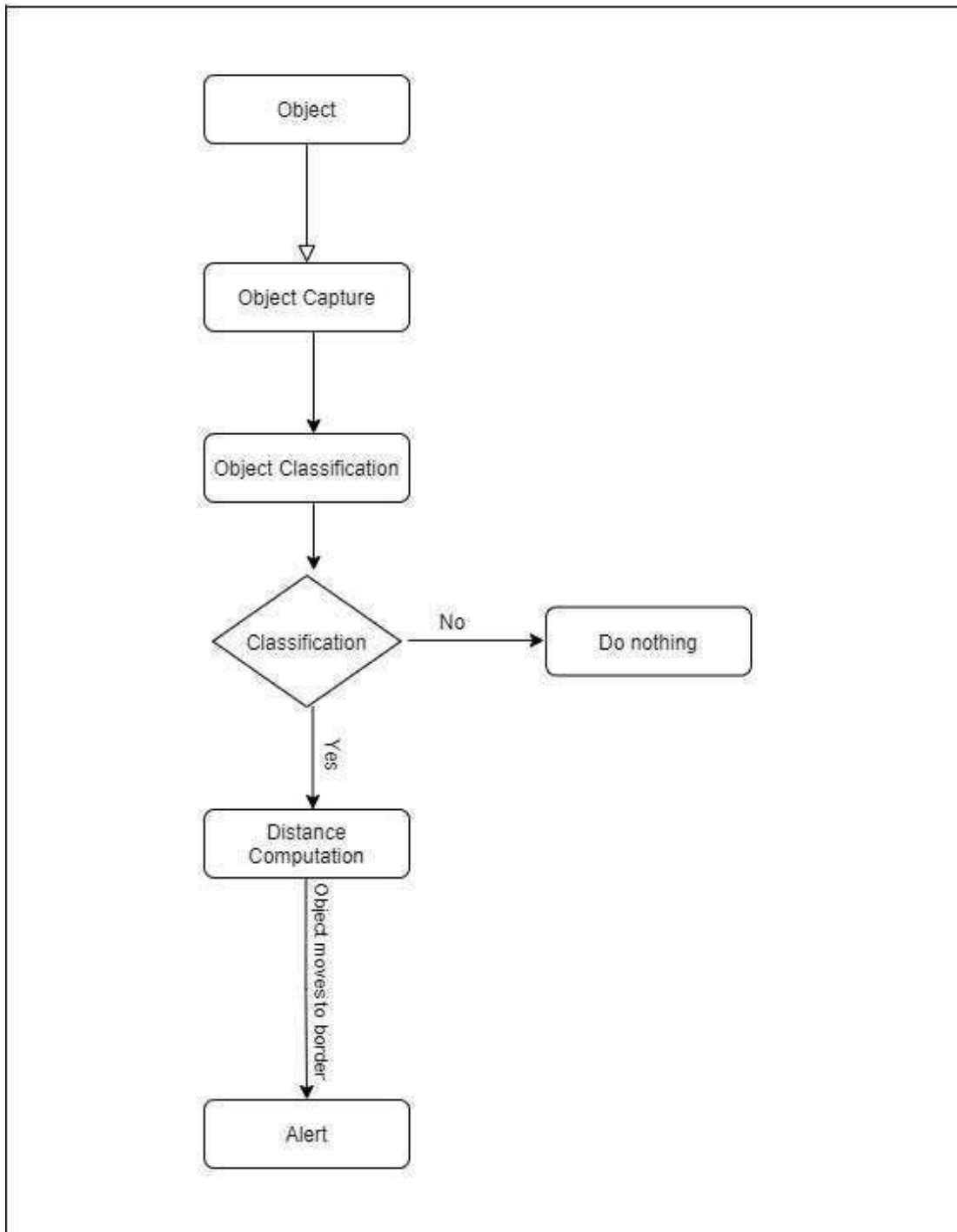


Fig 4.1

The above block diagram shows flow of work in the system. Arrows mentioned determines the direction of work flow as we here the work flows from top to bottom all the arrows are directed towards down.

As we see there are ovals which represent the number of modules present in the flow of work.

The whole system is integrated in a way such that when an object enters the room it follows certain steps to prevent the object from accidents. First, the object capture module is enabled when an object comes in to the focal range of a camera. Once the object gets captured in the object capture module, it is sent as an input to the second module called object classification.

As object classification module is a pre-trained model with the dataset it extracts set of features from the input image and compares with the model created from trained data. If the input image of the object matches with any of the trained data, then it will trigger the next module to compute distance between the object and line of control.

As the object is identified, this module calculates the distance between the object and the LOC. If the identified object is getting close to the boundary or line of control, then it triggers the next module called speak. This is the last step where the system prompts the certain message given by the user to alert the rest of the people in the house

ARCHITECTURE DIAGRAM

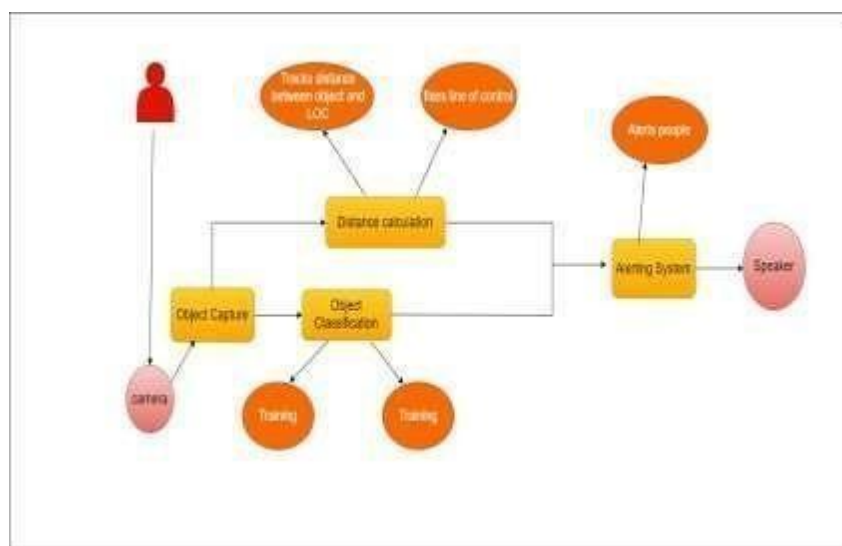


Fig 4.2

SDLC USED

The entire system is developed using agile model. Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like –

Planning

Requirements Analysis

Design

Coding

Unit Testing and

Acceptance Testing

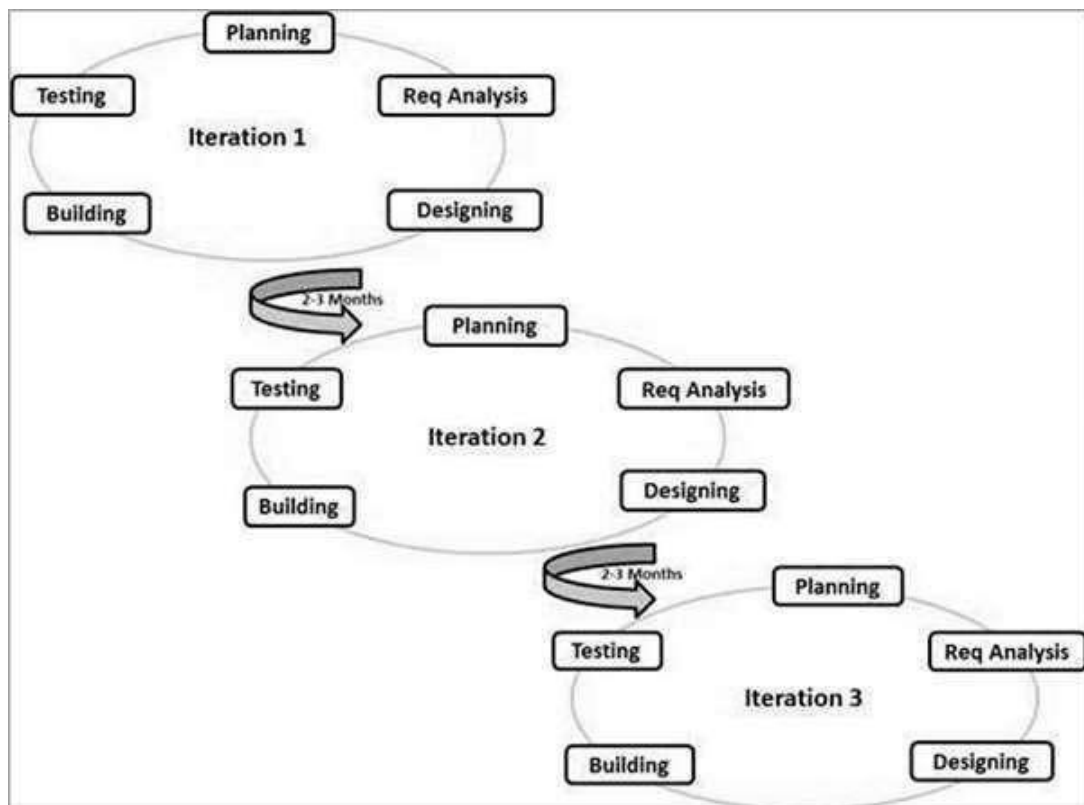


Fig 4.3

At the end of the iteration, a working product is displayed to the customer and important stakeholders.

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

The Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

The most popular Agile methods include Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995). These are now collectively referred to as Agile Methodologies, after the Agile Manifesto was published in 2001.

Following are the Agile Manifesto principles –

Individuals and interactions – In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.

Working software – Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.

Customer collaboration – As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.

Responding to change – Agile Development is focused on quick responses to change and continuous development.

By using the above methodology, the required system is made as the following parts. As we have mentioned that there were four modules involved in the

development of the system.

Each module has undergone cycles that were present in each iteration. First iteration has dealt with the development of the photo capture module using the opencv libraries, where the following changes are made to achieve the requirements. As we know that the module stores the images directly in the location where the code is store but the changes are made to system in such a way that all the captured images are stored in the buffer memory given by the user. Also auto refresh option is enabled where the photos gets deleted once the capacity reaches its limit.

After the development of the module, the module has undergone the testing process to check whether the changes made are working and to check the ability of the code.

This process is continued for all the rest modules where each module is integrated to the previous developed module once the development is done in the iteration.

Thus, it took around four iterations to develop the system as it involves four various modules.

RESULTS AND DISCUSSION

[illegible]

This is the output of the first module that is photo capture, where the camera captures the faces of the objects. Also the captured images are stored in a location set by the user in the code, the images captured are stored in the location. Also the buffer memory is set in such a way that it accepts the any number of images before the iteration, once the iteration starts the it captures all the images, but the images gets refreshed and deleted if the images are crosses the limit of the images. Also, the last image captured before the camera gets closed is sent to a notepad called s.txt where this is sent as input to the next

module.

Next comes the classification module where it has two part one is the testing phase and training phase. The following image is the training image where the images are given through a dataset in two folders known as training and testing folders. The images are divided in such a way that the images are separated in the ratio 7:3 where 70% photos are stored as training and 30% are tested for the model generation. All the images undergo the four steps of CNN and creates a model with the extension called .h5 where the it collects all the features from given images.

```
Epoch 1/25
250/250 [=====] - 281s 1s/step - loss: 0.7097 - accuracy: 0.6024 - val_loss:
0.5902 - val_accuracy: 0.7002
Epoch 2/25
250/250 [=====] - 78s 311ms/step - loss: 0.5975 - accuracy: 0.6820 - val_loss:
0.6361 - val_accuracy: 0.6592
Epoch 3/25
250/250 [=====] - 78s 310ms/step - loss: 0.5703 - accuracy: 0.7053 - val_loss:
0.6176 - val_accuracy: 0.6952
Epoch 4/25
250/250 [=====] - 83s 330ms/step - loss: 0.5479 - accuracy: 0.7206 - val_loss:
0.5446 - val_accuracy: 0.7267
Epoch 5/25
250/250 [=====] - 93s 370ms/step - loss: 0.5375 - accuracy: 0.7261 - val_loss:
0.5541 - val_accuracy: 0.7307
Epoch 6/25
250/250 [=====] - 101s 402ms/step - loss: 0.5217 - accuracy: 0.7409 - val_loss:
0.5667 - val_accuracy: 0.7317
Epoch 7/25
250/250 [=====] - 97s 388ms/step - loss: 0.5022 - accuracy: 0.7505 - val_loss:
0.5234 - val_accuracy: 0.7462
Epoch 8/25
250/250 [=====] - 82s 326ms/step - loss: 0.4859 - accuracy: 0.7620 - val_loss:
0.5598 - val_accuracy: 0.7387
Epoch 9/25
250/250 [=====] - 86s 344ms/step - loss: 0.4732 - accuracy: 0.7688 - val_loss:
0.5251 - val_accuracy: 0.7477
Epoch 10/25
250/250 [=====] - 84s 336ms/step - loss: 0.4599 - accuracy: 0.7772 - val_loss:
0.5466 - val_accuracy: 0.7407
Epoch 11/25
250/250 [=====] - 87s 348ms/step - loss: 0.4437 - accuracy: 0.7940 - val_loss:
0.5325 - val_accuracy: 0.7487
Epoch 12/25
250/250 [=====] - 90s 361ms/step - loss: 0.4330 - accuracy: 0.7991 - val_loss:
0.5058 - val_accuracy: 0.7663
Epoch 13/25
250/250 [=====] - 91s 362ms/step - loss: 0.4250 - accuracy: 0.8044 - val_loss:
```

Fig 5.2

Once the images are trained then comes the second phase called testing phase where the input is taken from the previous module. Now, the input taken from the previous module undergoes the four steps which were followed by the remaining images and the features. Now the extracted features are compared with features stored in the file called model.h5 and predicts the input as identified or not. This process is called classification of images based on the training set given by the

user.

```

In [6]: runfile('F:/Phani Lappy/AI INTREN/cnn/dataset/prediction.py', wdir='F:/Phani Lappy/AI INTREN/cnn/
dataset')
WARNING:tensorflow:From F:\Phani lappy\AI INTREN\cnn\dataset\prediction.py:25: Sequential.predict_classes
(from tensorflow.python.keras.engine.sequential) is deprecated and will be removed after 2021-01-01.
Instructions for updating:
Please use instead: * `np.argmax(model.predict(x), axis=-1)`, if your model does multi-class
classification (e.g. if it uses a `softmax` last-layer activation). * `(model.predict(x) >
0.5).astype("int32")`, if your model does binary classification (e.g. if it uses a `sigmoid` last-
layer activation).
[[1]]
00G

```

Fig 5.3

Once object gets identified, the next step is to find the distance between the object and the Line of Control. The calculation of code is done by using the regression code which implements the data set using .csv format. The file consists of predictions of the distance and also this module uses the photo capture module where a parameter called w is used to calculate distance.



Fig 5.4

gap	est.dst
331	0.3
290	0.4
133	0.83
125	0.87
117	1
106	1.1
101	1.2
92	1.3
79	1.4
66	1.5

Fig 5.5

Last module is the alerting system which is implemented in the distance computation module which prompts the message when an objects is close to the Line of Control. The system stops prompting once the objects moves away from the line of control.

```
import pytsx3 as pytsx
engine = pytsx.init()
rate = engine.getProperty('rate')
engine.setProperty('rate',rate)
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)
engine.say("alert!someone is crossing the line!hurry up!")
engine.runAndWait()
```

Fig 5.6

CHAPTER 6

CONCLUSION AND FUTURE WORK

CONCLUSION

Thus, by implementing the above system we can reduce the accidents that are caused due to the human negligence. Also, by implementing this system we can also restrict the movement of the certain people in the house and prevent them in most of the worst situations and getting affected. As this system involves detection features so there will be a set of images of person and when the particular person is about to cross the line of control then the system starts alerting by the speaker and the alerting gets reduced when the detected person moves away from the camera. As this involves various modules such as photo capture, prediction, distance calculation and speak module. When an object enters the room and starts moving towards the camera then the object gets captured by the photo capture module. This image is sent as input to the prediction module where classification is done. If the object is classified, then the distance computation module starts calculating distance and if distance gets close to the LOC then the system starts alerting unless the object moves back.

FUTURE WORK

The work in the system can be extended in future in such a way that, if the person crosses the Line of Control then the system activates with a physical instrument which can gently push the object without harming the object. This can also be installed with a grill system where when a person crosses the LOC then it raises the grills and prevents the movement of the object unless another object comes and releases him from the grill.

References

- [1] D. F. Warring, "Teacher evaluations: Use or misuse," *Universal J. Educ. Res.*, vol. 3, no. 10, pp. 703_709, 2015.
- [2] B. P. Smith and B. Hawkins, "Examining student evaluations of black college faculty: Does race matter," *J. Negro Edu.*, vol. 80, no. 2, pp. 149_162, 2011.
- [3] S. L. Sohr-Preston, S. S. Boswell, K. McCaleb, and D. Robertson, "Professor gender, age, and 'hotness' in influencing college students' generation and interpretation of professor ratings," *Higher Learn. Res. Commun.*, vol. 6, no. 3, pp. 1_21, 2016.
- [4] P. A. Schutz, "Inquiry on teachers' emotion," *Educ. Psychologist*, vol. 49, no. 1, pp. 1_12, 2014.
- [5] P. Ekman and W. V. Friesen, *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Palo Alto, CA, USA: Consulting Psychologists Press, 1978.
- [6] S. Prosen, H. S. Vitulic, And O. P. Skraban, "Teachers' emotional expression in the classroom," *Hacettepe Üniversitesi Eğitim Fakültesi Dergisi*, vol. 29, no. 1, pp. 226_237, 2014.
- [7] E. W. Lindsey, "Frequency and intensity of emotional expressiveness and preschool children's peer competence," *J. Genetic Psychol.*, vol. 180, no. 1, pp. 45_61, 2019.
- [8] Y. Sun and Q. Li, "The application of deep learning in mathematical education," in *Proc. 1st IEEE Int. Conf. Knowl. Innov. Invention (ICKII)*, Jul. 2018, pp. 130_133.

SOURCE CODE

```
# Face Recognitio

# Importing the
libraries import cv2

import
time

import os

import sys

#photo directory+storage+removal

d=os.chdir(os.getcwd()+"\\input") # d for
directory l=os.listdir(d) # l for list of filenames

if len(l)>=10:

    print(os.getcwd())

    for i in l:

        os.remove(i)

        #print(i)#showa deleted items

os.chdir("F:\\Phani_lappy\\AI_INTREN\\Project\\dataset"

) # Loading the cascades

face_cascade= cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

# Defining a function that will do the
detections def detect(gray, frame):

    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
```

```
for (x, y, w, h) in faces:  
    print (faces.shape)
```

```

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

print("Data Found")

cv2.putText(frame, "Number of faces detected: " + str(faces.shape[0]),
(10,30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

roi_gray = gray[y:y+h, x:x+w]

roi_color = frame[y:y+h, x:x+w]

FaceFileName = "img_" + str(y) +

".jpg" a = open("s.txt",'w')

a.write(FaceFileName)

a.close()

cv2.imwrite("input\\"+FaceFileName, roi_color)

eyes = eye_cascade.detectMultiScale(roi_gray, 1.1, 3)

for (ex, ey, ew, eh) in eyes:

    cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)

return frame

# Doing some Face Recognition with the

webcam start = time.time()

Total_Execution_Time = 10 #sec

photo_capture = cv2.VideoCapture(0)

while True:

    _,frame = photo_capture.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

```

```
canvas = detect(gray, frame)  
cv2.imshow('Video', canvas)
```

```

time.sleep(1)#comment

"""if cv2.waitKey(1) & 0xFF ==

    ord('q'): break"""

if time.time() > start + Total_Execution_Time:

    break

photo_capture.release()

cv2.destroyAllWindows()

#Prediction with distance calculation

from keras.models import load_model

from keras.preprocessing import

image import numpy as np

import

cv2

import os

model = load_model('mymodel.h5')

model.compile(loss='binary_crossentropy',

              optimizer='adam',

              metrics=['accuracy']

              )

from skimage.io import imread

from skimage.transform import

resize import pyttsx3

from sklearn import

```

```
linear_model import pandas as
```

```
pd
```

```
def lin_reg(input):
```

```
    data_set
```

```
=
```

```
pd.read_csv("F:/Phani_lappy/AI_INTREN/Project/dataset/distance/testfile.csv")
```

```

y =
data_set["est.dst"] x
= data_set["gap"]
x = x .values
x = x.reshape(-1,1)
model = linear_model.LinearRegression()
model.fit(x,y)
dst =
model.predict([[input]])
return dst[0]

def speak():
    engine = pyttsx3.init()
    voices = engine.getProperty('voices')
    engine.setProperty('voice', voices[1].id)
    pyttsx3.speak("the subject is crossed his limit")

def detect(gray,frame):
    global input
    face_cascade=cv2.CascadeClassifier("F:/Phani_lappy/AI_INTREN/Project/dataset
/distance/haarcascade_frontalface_default.xml")
    faces = face_cascade.detectMultiScale(gray,1.3,5)
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
        roi_gray = gray[y:y+h, x:x+w]

```

```
roi_color = frame[y:y+h,
```

```
x:x+w] input = w
```

```
dst = lin_reg(input)
```

```
if dst < 1:
```

```

        speak()

    return frame

photo_capture = cv2.VideoCapture(0)

cv2.namedWindow("winL")

cv2.resizeWindow("winL",400,300)

while True:

    _,frame = photo_capture.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    canvas = detect(gray,frame)

    cv2.imshow("winL", canvas)

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

photo_capture.release()

cv2.destroyAllWindows()


def detect(frame):

    #try:

        img = resize(frame,(64,64))

        img = np.expand_dims(img,axis=0)

        if(np.max(img)>1):

            img = img/255.0

        prediction =

        model.predict_classes(img)

```

```
print(prediction)
```

```
if(prediction):
```

```

    print ("CAT")

    detect()

    return "not
harmfull" else:

    #print ("DOG")

    return "harmfull"

"""except AttributeError:

print("shape not
found") engine =
pytttsx3.init()

rate = engine.getProperty('rate')

engine.setProperty('rate',rate)

voices = engine.getProperty('voices')

engine.setProperty('voice', voices[1].id)

engine.say("alert!unknown crossing the line!hurry
up!") engine.runAndWait()"""

a = open("s.txt",'r')

w = a.read().strip()

a.close()

print(w)

frame=cv2.imread("input\\"+w
)

```

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
data = detect(frame)
```

```
#Dataset Generation
```

```
from keras.models import load_model
```

```
from keras.preprocessing import
```

```
image import numpy as np
```

```
import
```

```
cv2
```

```
import os
```

```
model = load_model('mymodel.h5')
```

```
model.compile(loss='binary_crossentropy',
```

```
              optimizer='adam',
```

```
              metrics=['accuracy']
```

```
    )
```

```
from skimage.io import imread
```

```
from skimage.transform import
```

```
resize import pyttsx3
```

```
from sklearn import
```

```
linear_model import pandas as
```

```
pd
```

```
def lin_reg(input):
```

```
    data_set
```

```
=
```

```
pd.read_csv("F:/Phani_lappy/AI_INTREN/Project/dataset/distance/testfile.csv")
```

```
) y = data_set["est.dst"]  
  
x =  
  
data_set["gap"] x  
  
= x .values  
  
x = x.reshape(-1,1)  
  
model = linear_model.LinearRegression()
```

```
model.fit(x,y)

dst =

model.predict([[input]])

return dst[0]
```

```
def speak():

    engine = pyttsx3.init()

    voices = engine.getProperty('voices')

    engine.setProperty('voice', voices[1].id)

    pyttsx3.speak("the subject is crossed his limit")
```

```
def detect(gray,frame):

    global input

    face_cascade = cv2.CascadeClassifier("F:/Phani_lappy/AI_INTREN/Project/dataset/distance/haarcascade_frontalface_default.xml")

    faces = face_cascade.detectMultiScale(gray,1.3,5)

    for (x, y, w, h) in faces:

        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

        roi_gray = gray[y:y+h, x:x+w]

        roi_color = frame[y:y+h,

        x:x+w] input = w
```

```
dst = lin_reg(input)
```

```
if dst < 1:
```

```
    speak()
```

```

    return frame

photo_capture = cv2.VideoCapture(0)

cv2.namedWindow("winL")

cv2.resizeWindow("winL",400,300)

while True:

    _,frame = photo_capture.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    canvas = detect(gray,frame)

    cv2.imshow("winL", canvas)

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

photo_capture.release()

cv2.destroyAllWindows()


def detect(frame):

    #try:

        img = resize(frame,(64,64))

        img = np.expand_dims(img,axis=0)

        if(np.max(img)>1):

            img = img/255.0

        prediction =

        model.predict_classes(img)

        print(prediction)

```

```
if(prediction):  
    print ("CAT")
```

```

        detect()

        return "not

harmfull" else:

        #print ("DOG")

        return "harmfull"

    """except AttributeError:

    print("shape not

found") engine =

pytsx3.init()

rate = engine.getProperty('rate')

engine.setProperty('rate',rate)

voices = engine.getProperty('voices')

engine.setProperty('voice', voices[1].id)

engine.say("alert!unknown crossing the line!hurry

up!") engine.runAndWait()"""

a = open("s.txt",'r')

w = a.read().strip()

a.close()

print(w)

frame=cv2.imread("input\\"+w

)

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

```

```
data = detect(frame)
```