IN ORGANIZATION EMPLOYS TRACKER USING MACHINE LEARNING

Vaibhav Sharma[1], Faran Siddiqui[2], Amod Choubey[3], Prof. Ruchi Rahi[4]

Department of Computer Engineering
Theem College of Engineering

Abstract: Security and surveillance are important issues in today's world. The recent acts have highlighted the urgent need for eficient surveillance. Contemporary surveillance systems use digital video (DVR) cameras which play host to multiple channels. The major drawback with this model is that it requires continuous manual monitoring which is infeasible due to factors like human fatigue and the price of manual labor. Moreover, it's virtually impossible to look through recordings for important events within the past since that might require the playback of the whole duration of video footage. Hence, there's indeed a requirement for an automatic system for video surveillance that may detect unusual activities on its own.

Keywords: Video Surveillance System, Face Identity, Computational Intelligence.

1. INTRODUCTION

The practical needs of the surveillance system have been growing rapidly in the past decade. These growth rates are also supported by the development of high-definition technology. Surveillance systems have earned huge attention as application-oriented researchers are integrating the ability of computer vision, machine learning, and image processing. The major purpose of this approach is to automatically interpret the scene in a video, to observe, and to predict the interaction of an object within the scene supported information that has been gathered from camera sensors. The first utilization of the surveillance system was constructed from tube cameras which were distributed to monitor or to relay the factory activities in the 1940s. The conventional video surveillance system was often named Close-Circuit Television (CCTV), which was fragile and expensive since the cameras were provided and installed by a security team to examine the occurrence within the scene from a video display. The appearance of digitalized CCTV and advanced computation systems has increased the expansion of a (semi-) self-loading system which is known as the 2nd generation of the surveillance system. This new system has received advantages from the previous advancement in digital video communication, such as effective data compression, steady transmission, and bandwidth decrement. The progress in the new generation to enhance the efficiency in CCTV monitoring. The issues that often come up in the surveillance system are the detection and tracking algorithms that are required for behavioral analysis. The automatic video surveillance system integrates real-time live computer vision and an intelligent approach to solve these issues. The implementation of this approach can, therefore, help the system to give a real-time alert and forensic examination results for the security staff because it's supported by a complicated video analysis technique. The implementation of advanced use of video closed-circuit television in recent decades has led to varied domains, like crime prevention; elder treatment; accident detection; traffic oversee and control; counting moving object such as pedestrians or vehicles; human activity understanding; motion detection; activity analysis; identifying, tracking, and classifying human. The availability, development, and price of processors and sensors have also led to the use of this technique in terms of supporting indoor and outdoor neighborhoods such as shopping malls, airports, train stations, and parking lots. The study of video surveillance systems is often associated with another multidisciplinary area, such as image or pattern analysis and recognition, signal processing, distributed system, and communication. Currently, lots of studies have proposed the implementation of network infrastructure, and hardware and software components that support and provide efficient and effective video surveillance services to customers. There are several datasets, processing methods, and proposed frameworks that have been published. However, the currently proposed

studies of intelligence systems of video surveillance have provided an unclear explanation due to their complex implementation. Due to various publications in numerous journals, conducting a direct comparison may face difficulties. The main purposes of this study are, therefore, as follows: • To colligate and classify the studies related to the surveillance system. • to supply and propose an idea of the framework which will be integrated and classified based on the corresponding articles. • To present recommendations for the development of surveillance system researches upon the completion of this review. This study is conducted as follows: the methodology of the study is presented in Section 2. The outcomes and answers to the research questions are then discussed in Section 3.

2. RELATED WORK

The related work done by the MobiDev for their employees where they are using camera module for face detection and identification by using cross-platform 128vector dlib library. They using it for secure access that prevents unidentified visitors from entering the restricted areas and provides access to the authorized persons.

3. PROPOSED SYSTEM

For this, we required the minimum system configuration that is:

Hardware Requirements

Central Processing Unit (CPU) — Intel i5 7th Gen processor or higher. An AMD equivalent processor.
RAM — 8 GB min, if possible 16 GB or higher.
Graphics Processing Unit (GPU) — NVIDIA GTX 960 or higher. AMD GPUs will not work for DL, ML Or AI.
Operating System — Microsoft Windows 10.

Software Requirements

Anaconda
Install Python latest Version
Update Anaconda
Install CUDA & cuDNN
Create an Anaconda Environment
Install Deep Learning API's
Install Dlib using anaconda Prompt
Install face_recognition Classifier using the same anaconda Prompt.

4. Data Collection & Pre-processing

For DATA Collection we are using a camera module that collects the reference images from different angles and identifies the faces and provides the id to the faces which are identified.
after that, the collected images are again used by the face recognition classifier to identify the face in real-time. Once the faces are detected it starts creating the log by using the date-time classifier. And we can use that logs to calculate the total avail time to the employees in different areas in the organization.

5. How the Face Recognition works

face_recognition package

Module contents

`face_recognition.api.batch_face_locations(images, a number of times to upsample=1, batch size=128)[source]`

Returns the 2d arrays bounding boxes of the human face in the pictures using the CNN algorithm for face detection If there is a GPU, this can produce a much faster result since the GPU can process multiple images at once. If there is no GPU, then no need for this function.

Parameters:

- **image** – A collection of images (each as the NumPy array)
- **a number of times to upsample** – Number of times to upsample the images looking for faces. High numbers detect the smaller faces.
- **batch size** – Number of images to include in GPU processing.

A data of tuples that found faces locations in CSS (top, right, bottom, left) order

`face_recognition.api.compare_faces(known face encodings, face encoding to check, tolerance=0.6)[source]`

Check the list of face encoding against a candidate encoding to know if they match.

Parameters:

- **known face encodings** – known face encodings
- **face encoding to check** – An single face to encode to compared the list
- **tolerance** – The distance between faces to check the match. The lower is the more strict. 0.6 is the typical

best performance.

s: list of True or False values indicates which known face encodings match the face encoding

`face_recognition.api.face_distance(face encodings, face to compare)[source]`

Given the list of faces to encodings, check them to known faces encoding and get a Euclidean distance for each face. The distance tells how similar the faces are.

Paramet ers:

- **face encodings** – face encodings to check
- **face to compare** – face encoding to check against

s: A NumPy array with a distance of each faces in the same as the 'faces' array

`face_recognition.api.face_encodings(face image, known face locations=None, num jitters=1, model='small')[source]`

Gives an image that returns 128vector of faces encoding each face in the image.

Parameters:
- **face image** –an image that contains one or more face
- **known face locations** – the bounding boxes of each faces as known
- **num jitters** – Number of times to re-sample the faces check to encode. the Higher is more accurate, but slower result (i.e. 100 is 100x slower)
- **model** – which model to use. large or small which only returns the 5 points but is fast.

Returns: list of 128-dimensional face encodings in the image.

`face_recognition.api.face_landmarks(face image , face location =None, model='large')`

Given an image, returns a distance of faces feature locations of (eyes, nose, etc) in each face.

Parameters:
- **face image** – image to search
- **face location** – provide a list of face locations to compare.
- **model** – Optional - which model to use. large or small which gives 5 points but more faster.

Returns: list of dots of face feature(eye, nose, and much more)

`face_recognition.api.face_locations(img, number of time to upsample = 1, model = 'hog')='hog')[source]`

Returns the array of a bounded box of human face in an image

Paramet ers:
- **img** – image
- **number of times to upsample** – Number of times to upsample the image checking for faces. A higher number find the smaller faces.
- **model** – Which face detection model to use. “hog” having less accuracy but faster on CPUs. “CNN” is a more accurate model which is GPU/CUDA accelerated. The default is “hog”.

s: list of tuples that found face locations in CSS (top, right, bottom, left) order

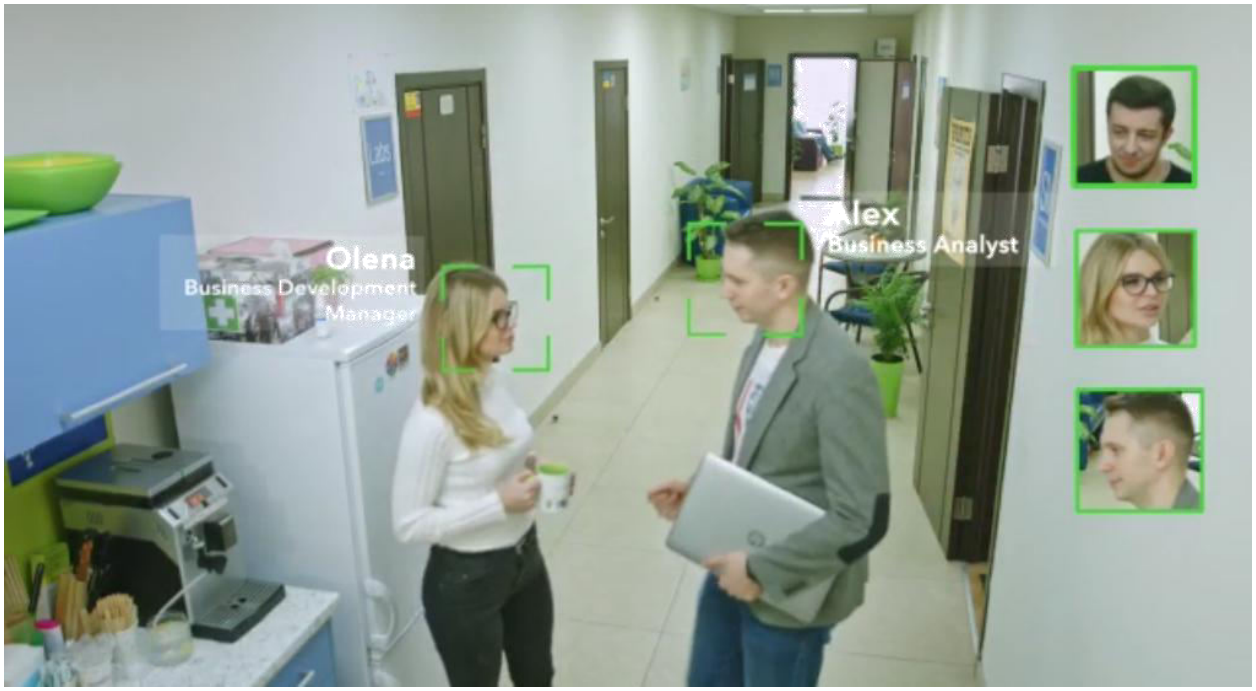
`face_recognition.api.load_image_file(file, mode='RGB')[source]`

Loads the image(.jpg,.jpeg, .png, etc) into a NumPy array

Paramet ers:
- **file** – image file
- **mode** – convert the image to ‘RGB’ (8-bit RGB, 3 channels) as well as ‘L’ (black and white)

Returns: images contents the NumPy array

6. EXPECTED RESULT:



7. CONCLUSION AND FUTURE WORK

Face identity and recognized by a face recognition classifier and it is easily able to create logs for the identify faces which we can further use to track the employee activity by using In Organization Employs Tracker Using Machine Learning.

REFERENCES

1.Smart video surveillance system DOI: 10.1109/ICIT.2010.5472694

2.Design of smart video surveillance system for indoor and outdoor scenes DOI: 10.1109/ICDSP.2017.8096120

3. *VIDEO SURVEILLANCE SYSTEMS https://www.researchgate.net/publication/228708805_VIDEO_SURVEILLANCE_SYSTEMS*