CROWD CONTROL

PROJECT REPORT

submitted by

KRUPA BOSE(CML17CS032)
STEFFI JOHNSON(CML17CS049)
K H NIRMAL DAS(MBI17CS022)
LIJO JOY(LMBI17CS033)

to

The APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology
In
Computer Science and Engineering

Department of Computer Science and Engineering

Mar Baselios Institute of Technology and Science
Nellimattom

DECEMBER 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAR BASELIOS INSTITUTE OF TECHNOLOGY & SCIENCE
Nellimattom

CERTIFICATE

This is to certify that the project report entitled **"Crowd Control"** Submitted by ***KRUPA BOSE*** with Reg no: CML17CS032, ***STEFFI JOHNSON*** with Reg no: CML17CS049***, K H NIRMAL DAS*** with Reg no: MBI17CS022 and ***LIJO JOY*** with Reg no: LMBI17CS033 during the academic year 2020-2021 towards the partial fulfillment of the requirement for the award of Bachelor of Technology in Computer Science & Engineering of APJ Abdul Kalam Technological University Trivandrum, Kerala is a bonafide work carried out by him in this department under our guidance and supervision.

Tinku Soman Jacob Assistant Professor Project Guide	Bonia Jose Assistant Professor Project Coordinator	Midhun Mathew Head of Department Computer Science and Engineering

ACKNOWLEDGEMENT

First and the foremost, I thank **ALMIGHTY GOD** who gave me the inner strength, resource and ability to complete my work successfully, without which all my efforts would have been in vain.
I express my sincere gratitude to the principal, **Dr. P SOJAN LAL** for providing me the provision to do the project in the successful way.
I stand grateful to **Asst. Prof. Midhun Mathew**, Head of the Department of Computer Science and Engineering, for his whole hearted support.
I express my sincere gratitude to my project co-ordinator, **Asst. Prof. Bonia Jose** for her whole-hearted support.
I extend my heartfelt gratitude to my project guide, **Asst. Prof. Tinku Soman Jacob** for his guidance and encouragement which where indispensable for the fulfillment of this seminar presentation.
I use this opportunity to thank all faculties in the department and the friends who have helped us to complete the project preliminary, with their inspiration and co-operation. Last but not the least I am thankful to my parents without their prayers and blessings my project presentation would not have been a success.

ABSTRACT

Video surveillance is gaining popularity in numerous applications, including facility management, traffic monitoring, crowd analysis, and urban security. Despite the increasing demand for closed circuit television (CCTV) and related infrastructure in public spaces, there remains a notable lack of readily deployable automated surveillance systems. In this study, we present a low-cost and efficient approach that integrates the use of computational object recognition to perform fully-automated identification, tracking, and counting of human traffic on camera video streams. Two software implementations are explored and the performance of these schemes is compared. Validation against controlled and non-controlled real-world environments is also demonstrated. The implementation provides automated video analytics for medium crowd density monitoring and tracking, eliminating labor-intensive tasks traditionally requiring human operation, with results indicating great reliability. Crowd Control (CC) is a low cost and efficient approach which integrates computations to overcome this situation by counting and identifying individuals from a crowd of a monitored video surveillance sequence. YOLO V3 algorithm is used to group objects and extract features from the video sequence and simultaneously count the pedestrian moving from the sequence from one frame to another.

CONTENTS

CONTENTS

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATION

CNN	CONVOLUTIONAL NEURAL NETWORK
BGS	BACKGROUND SUBTRACTION
SSD	SINGLE SHOT DETECTOR
DNN	DEEP NEURAL NETWORK
IDE	INTEGRATED DEVELOPMENT ENVIRONMENT

CHAPTER 1
INTRODUCTION

1.1 PROBLEM STATEMENT

Video surveillance is an integral component of modern urban security, and when coupled with computational analytics, can have greatly expanded functionality including facial recognition, motion detection, traffic and crowd monitoring, and automated hazard alarms. The continued advancement in computational tools and machine learning has in principle enabled automation of a wide variety of practical analyses on image and video inputs; more advanced machine intelligence systems are also increasingly capable of fulfilling traditionally human-controlled tasks that require real-time complex decisions, for instance initiating mitigation measures for severe traffic congestion or the dispatching of emergency services. In general, automated surveillance eliminates the need for round-the-clock manual monitoring, thereby reducing manpower requirements. This stands to yield operational cost reductions and productivity improvements. Nonetheless, amidst the progressing state-of-the-art, integration of automated analytics in commercial video surveillance for crowd monitoring and counting is an area that can be further explored and there is at present limited literature on demonstrated effective low-cost systems for deployment. In security and management sectors, there remains a great reliance on traditional manual monitoring of CCTV footage, and human patrols to conduct crowd monitoring and tracking.

Utilizing computer vision and real-time automated analytics in replacement of manual labour not only reduces operational costs but also eliminates human errors and lapses we seek to develop a viable deployment-ready implementation in this study. In this paper, we examine several viable approaches to automated crowd monitoring and tracking in indoor and outdoor scenarios, ultimately selecting a statistical background subtraction (BGS) scheme and a convolutional neural network-based single shot detector (SSD). These methods are easily deployable in the real world. A software solution is developed for use in general public spaces, and we validate the performance of the platform through indoor controlled tests in a shopping mall, and outdoor non-controlled tests in a public transport hub with considerable human traffic.

It is noted that the implementation of this video analytics system can also be applied to a broader range of scenarios— for instance, in factories to detect personnel in restricted places or in dangerous proximity to machinery, or in high-rise buildings to detect crowd densities that exceed safe thresholds for timely evacuation in case of emergencies. Time-oriented data collected from such deployments can also be logged and transmitted to a dashboard for data-informed planning and predictive analytics.

CHAPTER 2

LITERATURE SURVEY

We first provide an overview of object recognition frameworks, the challenges associated with achieving satisfactory performance, and the application of these frameworks in automated video analytics. A fundamental operational requirement of automated video surveillance analytics is the ability to identify and track different objects within the recorded footage, hence the need for object recognition; in the current context of crowd analysis, recognition of human subjects is critically relevant.

2.1 RELATED WORK

While visual recognition and classification of objects is intuitive to human perception, robust computational implementation is challenging. Varying exposure to outdoor conditions, changing illumination levels and direction, intermittent and sustained visual obstruction, and unpredictable movement of tracked subjects must all be overcome for reliable operation of recognition systems, oftentimes with limitations on available computational power. The resolution and clarity of available video footage is also typically non-ideal, limiting the effectiveness of pre-processing techniques aimed at compensating for variance in image conditions. In our context of recognition and tracking of human subjects, additional complexities arise from the wide range of possible dynamical behaviour—for instance, two persons in physical contact may be detected as a single entity, and the shape profile of a person may change drastically because of carried items or differing attire. Advanced machine vision systems are already being developed for security- and safety-critical applications, such as driverless vehicles and autonomous drones. These systems typically employ convolutional neural network (CNN)-based solutions that are trained on massive datasets of numerous modalities, including infrared and visible video input, lidar data, sound pick-ups from microphones, and navigational data from GPS or inertial guidance. Existing studies have shown excellent performance in the identification of key markers, such as lane boundaries, traffic signs, and pedestrians on systems intended for driverless vehicles under a wide range of lighting and driving conditions

CNN-based image recognition has also been applied very successfully to facial identification tasks, achieving large reductions in error rate when compared to non-CNN methods. These types of CNN-based methods are presently employed for automated user identification and tagging systems in prominent social media platforms. In general, CNN-based systems are hugely robust to changing background conditions and object appearances, but are typically computational expensive to train and run. In comparison to explicit rule-based or statistical methods, the employed CNN architectures are also more akin to black boxes and offer limited tractability—troubleshooting and tuning the systems for specific environments can therefore be challenging. There is also recent evidence attributing the efficacy of neural network deep-learning solutions to fine-tuning rather than a fundamental architectural advantage, suggesting that a properly tuned classical method may be able to achieve similar performance in certain scenarios. Indeed, non-CNN methods have been deployed to perform similar tasks. A real-time system for pedestrian tracking using gray-scale images from stationary cameras has been demonstrated, with satisfactory robustness to visual occlusions and ambiguities in perceived subject shape profiles. Numerous studies on pedestrian and traffic tracking, motion detection and analysis, and object classification using non-CNN methods have also been presented to date, suggesting good viability in these approaches. Non-CNN methods may hence be preferred in scenarios.

Traditional counting systems are generally based on infrared or pressure sensors. They are low cost but not easy to integrate with video surveillance system. Vision-based people counting systems become more popular these years for different scenes, like in buildings, streets and hot spot. In the literature, authors developed approaches relying on two main strategies: non-tracking based approaches and tracking based approaches. In the first case, authors try to discriminate foreground from background and count interesting targets. Mehta et al made use of classifiers such as neural networks trained to recognize the background in order to facilitate the location and counting of objects in the scene. Moreover, Schl¨ogl et al used motion features to classify each pixel as moving, stationary or background, and then grouped similar pixels together into blobs. They were compared later with the average human size varying with positions in the scene to estimate people number inside. Chao et al segmented crowd by motion model and extracted features from each segmentation. The correspondence between features and number of people were learned by Gaussian Process regression. In the second case, authors either count tracked people at a defined counting line or count people trajectories from tracking.

In, a tracking region was partitioned off from the scene with counting line on the edge. people were tracked by motion prediction combined with background subtraction and counted at the line. Another approach consists in getting feature trajectories in the scene by Kanade-Lucas-Tomasi (KLT) tracker, and then clustered trajectories with similar movement together for representing one moving object. This kind of methods are generally able to count a large number of people in a homogeneous crowd. From the state of the art, it appears that most of people counting approaches rely on the assumption that any moving objects in scenes are humans and suffer the miscount of other moving objects. In, a model of humans is defined based on average people size. In, skin color model is used to detect human. These are among the first tentative to elaborate more accurate people counting systems but still lack accuracy. In order to avoid this kind of miscounting, the basic idea of our approach is to use the most discriminant human feature: their face.

The problem of counting people moving toward the camera in a close space such as the entrance of a supermarket, bank or bus, where lightning conditions are relatively stable and people are generally facing the camera. Based on these scenes, we propose an approach that presents several improvements compared to the literature. The first improvement is the use of the face

detector to ensure that counted objects are people. Second, in order to deal with drastic changes of face scales in our scene, a scale-invariant Kalman filter is proposed. It is further combined with a kernel-based object tracking algorithm to handle face occlusions. Finally, we propose a strategy to count people by automatically classify face trajectories, which are characterized by an angle histogram of neighboring points. Two Earth Mover's Distance based classifier is used to discriminate true trajectories and false trajectories. The advantages are twofold. On the one hand, a filtering of the trajectories can be realized in order to reject false trajectories caused by false face detection and thus to improve counting accuracy. On the other hand, the automatic classification of the trajectories allows to avoid the manual and empirical elaboration of rules for counting people in a given scene.

Microsoft Kinect Sensor is used for detecting faces. Kinect has a RGB sensor and a IR sensor. CCTVs have a RGB sensor with IR LEDs for night vision. The device detecting the faces was Kinect with its vast hardware and sensors. These faces were passed on in a cropped manner to the system. Used a Ann to train the system using 4 face. The system was overall very fast with speeds sometimes around 200ms. The method used several sub methods like skin detection although he most prominent was face detection.

The Algorithm used was the Haar Cascade Classifier. HAAR Cascade classifier mainly works by finding patterns with reference to black and white rectangles. This particular paper used a advanced implementation which also involved diagonals. Every Face is modeled as an ellipse. Skin detection is used to speed up haar face detection .This is only detection not recognition. It is a multistep process where every frame was analyzed [3]. First step was detection using Haar classifier. Second step was detecting face using both Eigen Faces and Gabor algorithm, third step was decision making and selecting. The accuracy was around 50 percent because of changes in illumination pose…etc. The authors used method which specifically used Microsoft Azure to compute data [4]. This project main aim was to find objects in the environment for robot to navigate.It finds objects by identifying blobs .Another technique is to try a direct match .The user interface was a windows from C application. Authors used the method which used a technique to determine the consistent background [5]. This paper made use of 2 cameras to prevent any disturbance from lighting. The one particular method they used was background subtraction. The specific algorithm was the MOG background subtraction. Long term and shorter analysis is done in case someone come back to pick up his luggage.

Algorithm used was the background subtraction. Long term and shorter analysis is done in case someone come back to pick up his luggage. With increasing terrorist activities there was augmenting demand for video surveillance. Mostly images are generally classified based on the value of simple features [6]. It is always better to use features rather than using pixels as feature based systems always operates much quicker than pixel based systems. In this approach the algorithm consists of 3 intermediate steps : A) By using an intermediate representation for the image, Integral rectangle features can be computed very quickly. B) Adaboost technique can be used for the construction of classifiers that helps us to separate desired features from the collection of vast no of features. It utilizes a set of positive and negative images to train. C) Cascading of different classifiers A full advantage of Pattern Recognition and Image Processing model can be taken with the help of Open Computer Vision (OpenCV). To detect human face and achieve fast face detection of the video, mosaic gray rules were adopted. Three types of main algorithm based on mosaic model are: Gray Rules: The trisection image model according to the organ segmentation of face is established. Integral Image: Rectangle feature can be computed very quickly using an intermediate representation for the image.

Frequency Histogram: It removes the non-face region from the image and merge the overlapping face region. Changes in the facial appearance occurs due to natural ageing of human [8]. It consists of at least 5 face images of a subject that were collected over at least a 5year time spa. As elapsed time increases between two face images, population mean trends in genuine scores is estimated by using multilevel statistical models. Haar classifier is used to detect faces in the frames that are come as input [9]. A face detection method based on AdaBoost was adopted in the paper. Improved AdaBoost algorithm achieves more robust performance.

And high speed over conventional AdaBoost based methods. Study of locality preserving projection is essential and analyzed the impact of over existing face recognition techniques. Proposed a method using Bilinear CNN [10]. At each location convolution layer outputs of two CNNs of the image are multiplied. Image labels are used to train bilinear CNN model & it requires training network. Authors proposed a method using deep learning. Automatically collecting and labelling the data from CCTC videos is done in order to construct a dataset. Face were recognized using Face Recognition Algorithm. Haar Classifier is used for detecting faces. The paper reported accuracy of 99.2%. Training is an essential part and in this case required 2.6M images of 2.6 lakh people.

This paper classify object detection into many categories like model based system, image invariance method, example based method , static object detection , moving object detection .Recursive and non recursive algorithm isused for removing background. Object can be tracked using point tracking ,kernel tracking, edge detection and color. This paper classify object detection into many categories like model based system, image invariance method, example based method , static object detection , moving object detection.Recursive and non recursive algorithm is used for removing background .Object can be tracked using point tracking ,kernel tracking, edge detection and color. This paper proposes three steps. Steps include face detection, face feature extraction and finally face recognition . Face was detected and background removed, face features like face cuts and angles were formatted and styled, while recognition goes ahead and identifies it 3 detection methods were experimented with namely camshaft algorithm, Haar classifier, vend finding via motion. Camshift and finding via motion were fast but the most accurate and reliable was Haar classifier.

Authors introduces the face images are manually localized by publicly available media in the wild dataset (500 subjects). IJB-A protocol focuses on: the ability to search for a person's face in a set of images (search), and (ii) the ability to compare facial imagery from two persons, and verify whether or not they are the same person (compare). All faces have been manually localized and have not been filtered by a commodity face detector is a key distinction between this dataset and previous datasets. In the IJB-A dataset amount of variation in pose, occlusion and illumination is unprecedented. Based on conditional random field which is combined with the saliency measure which introduces a new salient object segmentation method. Statistical framework formulate saliency measure and contrast local features in illumination, color and motion information. Method is efficiently implemented by using statistical frame work.

The integral histogram approach and graph cut solvers effectively implements the method. The feature includes lab color values and optical formation which are obtainable in real time.

CHAPTER 3

METHODOLOGY

3.1 PROPOSED APPROACH

The proposed approach is based on object identification, crowd density monitoring and face recognition using various algorithms. This approach can also be used for monitoring crowd using a computer based security system.

3.1.1 : System Overview and Design

In this system, we have three modules which include object identification, crowd density monitoring and face recognition. We further describe key system components as depicted in Figure 3.1 as follows:

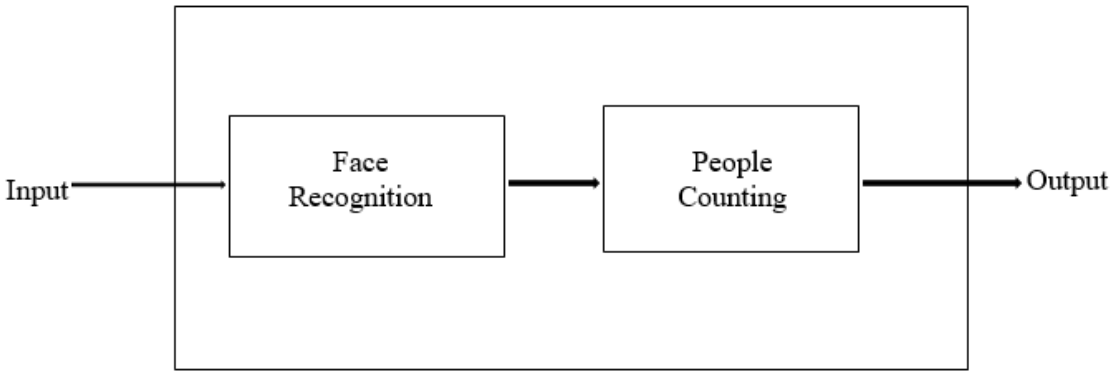


Fig. 3.1.1: System overview highlighting key components of proposed solution.

3.2 DATA FLOW DIAGRAMS

- **FACE DETECTION**

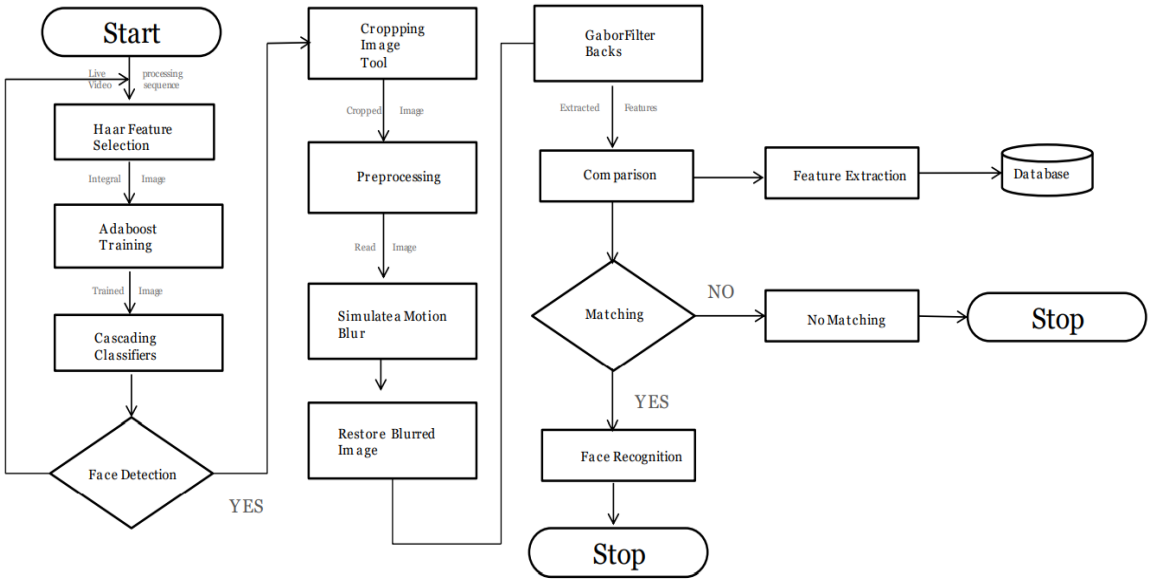


Fig 3.2.1: Face Recognition

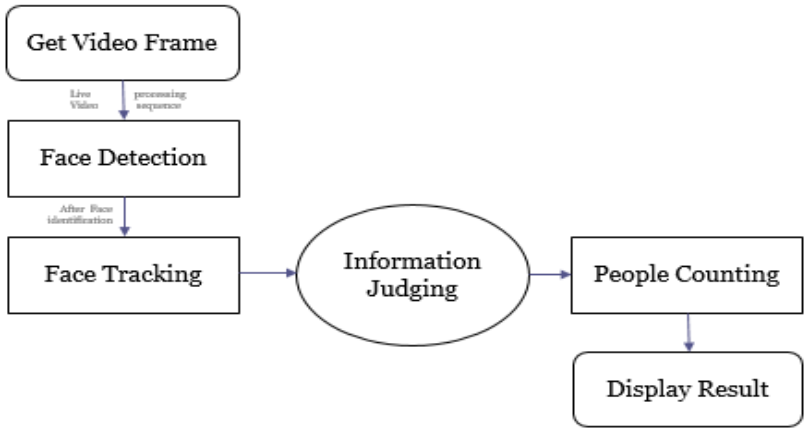


Fig 3.2.2: People counting

3.3 MAIN CONCEPTS

- Haar Feature Selection
- Adaboost Training
- Gabor Filter Banks
- Convolutional Neural Network
- HOG Algorithm
- People Counting

3.3.1 Haar Feature Selection

Haar-like features are digital image features used in object recognition. They owe their name to their intuitive similarity with Haar wavelets and were used in the first real-time face detector.

Historically, working with only image intensities (i.e., the RGB pixel values at each and every pixel of image) made the task of feature calculation computationally expensive. Paul Viola and Michael Jones adapted the idea of using Haar wavelets and developed the so-called Haar-like features. A Haar-like 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. This difference is then used to categorize subsections of an image. For example, with a human face, it is a common observation that among all faces the region of the eyes is darker than the region of the cheeks. Therefore, a common Haar feature for face detection is a set of two adjacent rectangles that lie above the eye and the cheek region. The position of these rectangles is defined relative to a detection window that acts like a bounding box to the target object (the face in this case).

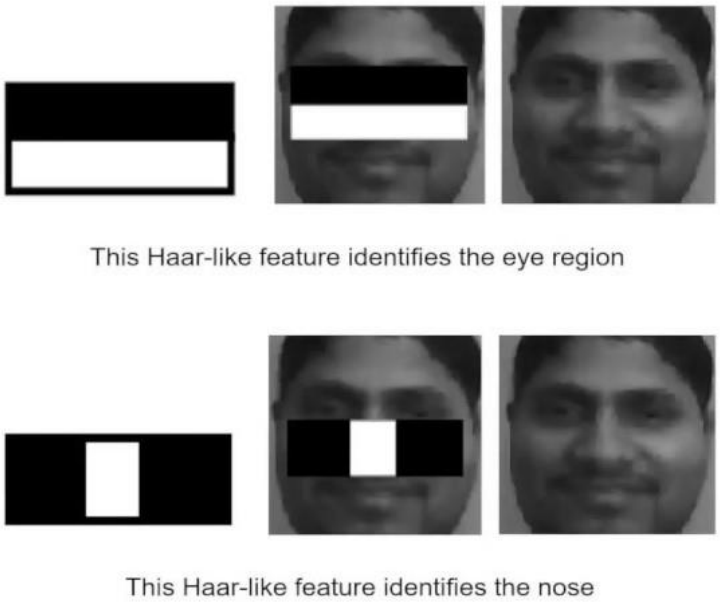


Fig 3.3.1: Haar Feature Selection

3.3.2 Adaboost Training

AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire. It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner.

Every learning algorithm tends to suit some problem types better than others, and typically has many different parameters and configurations to adjust before it achieves optimal performance on a dataset. AdaBoost (with decision trees as the weak learners) is often referred to as the best out-of-the-box classifier. When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the tree growing algorithm such that later trees tend to focus on harder-to-classify examples.

3.3.3 Gabor Filter Banks

In image processing, a Gabor filter, named after Dennis Gabor, is a linear filter used for texture analysis, which essentially means that it analyzes whether there is any specific frequency content in the image in specific directions in a localized region around the point or region of analysis. Frequency and orientation representations of Gabor filters are claimed by many contemporary vision scientists to be similar to those of the human visual system. They have been found to be particularly appropriate for texture representation and discrimination. In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave (see Gabor transform).

3.3.4 Convolutional Neural Network

A **Convolutional Neural Network (CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The agenda for this field is to enable machines to view the world as humans do, perceive it in a similar manner and even use the knowledge for a multitude of tasks such as Image & Video recognition, Image Analysis & Classification, Media Recreation, Recommendation Systems, Natural Language Processing, etc. The advancements in Computer Vision with Deep Learning has been constructed and perfected with time, primarily over one particular algorithm — a Convolutional Neural Network**.** CNNs are regularized versions of multilayer perceptron. Multilayer perceptron usually means fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

3.3.5 The Histogram of Oriented Gradients (HOG)

The Histogram of Oriented Gradients (HOG) is a feature descriptor used in computer vision and image processing applications for the purpose of the object detection. It is a technique that counts events of gradient orientation in a specific portion of an image or region of interest. HOG focuses on the structure of the object. It extracts the information of the edges magnitude as well as the orientation of the edges.

It uses a detection window of 64x128 pixels, so the image is first converted into (64, 128) shape.

The image is then further divided into small parts, and then the gradient and orientation of each part is calculated. It is divided into 8x16 cells into blocks with 50% overlap, so there are going to be 7x15 = 105 blocks in total, and each block consists of 2x2 cells with 8x8 pixels.

We take the 64 gradient vectors of each block (8x8 pixel cell) and put them into a 9-bin histogram.

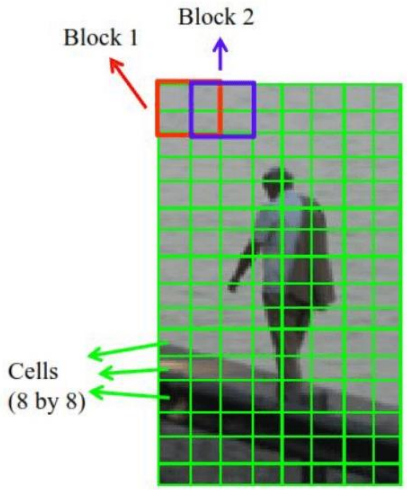


Fig 3.3.2: Blocks made in a frame

Fig 3.3.3: HOG version of image

Encode a picture using the HOG algorithm to create a simplified version of the image. Using this simplified image, find the part of the image that most looks like a generic HOG encoding of a face.
Figure out the pose of the face by finding the main landmarks in the face. Once we find those landmarks, use them to warp the image so that the eyes and mouth are centered.
Pass the centered face image through a neural network that knows how to measure features of the face. Save those 128 measurements.
Looking at all the faces we've measured in the past, see which person has the closest measurements to our face's measurements. That's our match!

The hog() function takes 6 parameters as input:

- image: The target image you want to apply HOG feature extraction.
- orientations: Number of bins in the histogram we want to create, the original research paper used 9 bins so we will pass 9 as orientations.
- pixels_per_cell: Determines the size of the cell, as we mentioned earlier, it is 8x8.
- cells_per_block: Number of cells per block, will be 2x2 as mentioned previously.
- visualize: A boolean whether to return the image of the HOG, we set it to True so we can show the image.
- multichannel: We set it to True to tell the function that the last dimension is considered as a color channel, instead of spatial.

3.3.6 People Counting

Required libraries:

OpenCV library in python is a computer vision library, mostly used for image processing, video processing, and analysis, facial recognition and detection, etc.
Dlib library in python contains the pre-trained facial landmark detector, that is used to detect the (x, y) coordinates that map to facial structures on the face.

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object and tools for working with these arrays.

- Step 1: Import required libraries
- Step 2: Open the default camera to capture faces and use the dlib library to get coordinates.
- Step 3: Count the number of faces.
    - Capture the frames continuously.
    - Convert the frames to grayscale
    - Take an iterator *i* and initialize it to zero.
    - Each time you get the coordinates to the face structure in the frame, increment the iterator by 1
    - Plot the box around each detected face along with its face count

- Step 4: Compare the unique features of that face to all the people you already know to determine the person's name.
- Step 5: Output displayed.

3.4 MODULES DESCRIPTION

1. FACE RECOGNITION: -

- Detect face from real time data by video surveillance system.
- Extract features using Haar Feature Selection.
- Apply Ada Boost algorithm to pick best features to train classifier.
- Gabor filter is applied for image extraction process.
- Face recognition by comparison process with the databases images.

Input: *CCTV Footage*

Output: *Object name with a bounding box*

2. CROWD DENSITY MONITORING: -

- Capture real time data by video surveillance system.
- Process them by image processing system.
- Count the identified objects and pedestrians moving from one region to another.

Input: *CCTV Footage*

Output: *No of persons obtained after object identification*

CHAPTER 4

IMPLEMENTATION DETAILS

4.1 PLATFORM

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

4.2 DATASET

- Surveillance Video
- Images of people to be recognized

4.3 REQUIREMENTS

4.3.1 certifi==2020.6.20

4.3.2 chardet==3.0.4

4.3.3 click==7.1.2

4.3.4 cmake==3.18.2.post1

4.3.5 decorator==4.4.2

4.3.6 dlib==19.18.0

4.3.7 face-recognition==1.3.0

4.3.8 face-recognition-models==0.3.0

4.3.9 idna==2.10

4.3.10 imageio==2.9.0

4.3.11 imageio-ffmpeg==0.4.2

4.3.12 moviepy==1.0.3

4.3.13 numpy==1.18.4

4.3.14 opencv-python==4.4.0.46

4.3.15 Pillow==8.0.1

4.3.16 proglog==0.1.9

4.3.17 requests==2.24.0

4.3.18 tqdm==4.51.0

4.3.19 urllib3==1.25.11

4.3.20 wincertstore==0.2

4.3.1 certifi==2020.6.20

Certifi provides Mozilla's carefully curated collection of Root Certificates for validating the trustworthiness of SSL certificates while verifying the identity of TLS hosts. It has been extracted from the Requests project.

4.3.2 chardet==3.0.4

Chardet: The Universal Character Encoding Detector

Detects

- ASCII, UTF-8, UTF-16 (2 variants), UTF-32 (4 variants)
- Big5, GB2312, EUC-TW, HZ-GB-2312, ISO-2022-CN (Traditional and Simplified Chinese)
- EUC-JP, SHIFT_JIS, CP932, ISO-2022-JP (Japanese)
- EUC-KR, ISO-2022-KR (Korean)
- KOI8-R, MacCyrillic, IBM855, IBM866, ISO-8859-5, windows-1251 (Cyrillic)
- ISO-8859-5, windows-1251 (Bulgarian)
- ISO-8859-1, windows-1252 (Western European languages)
- ISO-8859-7, windows-1253 (Greek)
- ISO-8859-8, windows-1255 (Visual and Logical Hebrew)
- TIS-620 (Thai)

4.3.3 click==7.1.2

Click is a Python package for creating beautiful command line interfaces in a composable way with as little code as necessary. It's the "Command Line Interface Creation Kit". It's highly configurable but comes with sensible defaults out of the box.

It aims to make the process of writing command line tools quick and fun while also preventing any frustration caused by the inability to implement an intended CLI API.

Click in three points:

- Arbitrary nesting of commands
- Automatic help page generation
- Supports lazy loading of subcommands at runtime

4.3.4 cmake==3.18.2.post1

CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment of your choice.

The suite of CMake tools were created by Kitware in response to the need for a powerful, cross-platform build environment for open-source projects such as ITK and VTK.

4.3.5 decorator==4.4.2

The goal of the decorator module is to make it easy to define signature-preserving function decorators and decorator factories. It also includes an implementation of multiple dispatch and other niceties (please check the docs). It is released under a two-clauses BSD license, i.e. basically you can do whatever you want with it but I am not responsible.

4.3.6 dlib==19.18.0

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high performance computing environments. Dlib's open source licensing allows you to use it in any application, free of charge.

- Routines for reading and writing common image formats.
- Automatic color space conversion between various pixel types
- Common image operations such as edge finding and morphological operations
- Implementations of the SURF, HOG, and FHOG feature extraction algorithms.
- Tools for detecting objects in images including frontal face detection and object pose estimation.
- High quality face recognition

4.3.7 face-recognition==1.3.0

Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library. Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark. This also provides a simple face_recognition command line tool that lets you do face recognition on a folder of images from the command line!

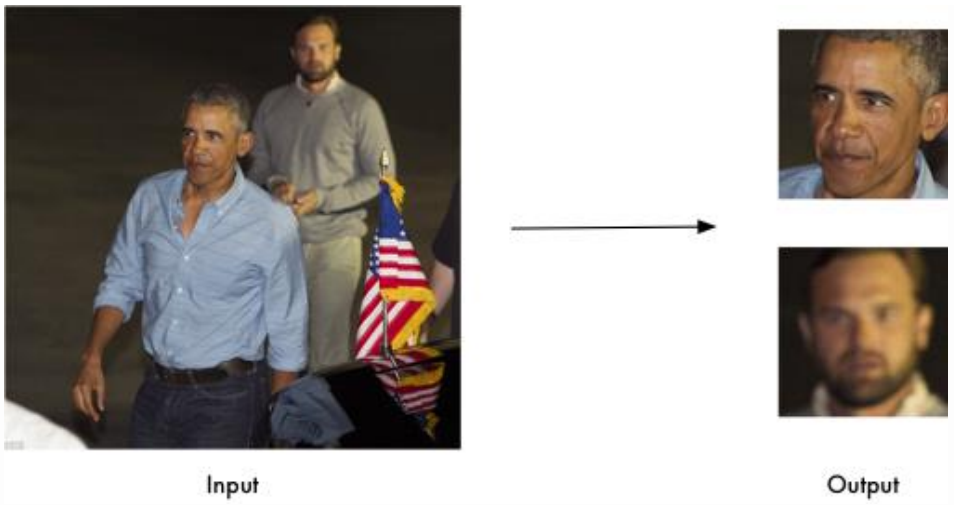


Fig 4.3.1: Face Recognition

4.3.8 face-recognition-models==0.3.0

This package contains only the models used by face_recognition.

4.3.9 idna==2.10

Support for the Internationalised Domain Names in Applications (IDNA) protocol as specified in RFC 5891. This is the latest version of the protocol and is sometimes referred to as “IDNA 2008”.

This library also provides support for Unicode Technical Standard 46, Unicode IDNA Compatibility Processing.

This acts as a suitable replacement for the “encodings.idna” module that comes with the Python standard library, but which only supports the old, deprecated IDNA specification (RFC 3490).

4.3.10 imageio==2.9.0

Imageio is a Python library that provides an easy interface to read and write a wide range of image data, including animated images, volumetric data, and scientific formats. It is cross-platform, runs on Python 3.5+, and is easy to install.

4.3.11 imageio-ffmpeg==0.4.2

FFMPEG wrapper for Python. ffmpy is a simplistic FFmpeg command line wrapper. It implements a Pythonic interface for FFmpeg command line compilation and uses Python subprocess module to execute compiled command line.

4.3.12 moviepy==1.0.3

MoviePy (full documentation) is a Python library for video editing: cutting, concatenations, title insertions, video compositing (a.k.a. non-linear editing), video processing, and creation of custom effects. See the gallery for some examples of use.

MoviePy can read and write all the most common audio and video formats, including GIF, and runs on Windows/Mac/Linux, with Python 2.7+ and 3 (or only Python 3.4+ from v.1.0).

4.3.13 numpy==1.18.4

It provides:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities
- and much more

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

All NumPy wheels distributed on PyPI are BSD licensed.

4.3.14 opencv-python==4.4.0.46

Unofficial pre-built CPU-only OpenCV packages for Python.

Check the manual build section if you wish to compile the bindings from source to enable additional modules such as CUDA.

4.3.15 Pillow==8.0.1

The Python Imaging Library adds image processing capabilities to your Python interpreter.

This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.

The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

Image Processing

The library contains basic image processing functionality, including point operations, filtering with a set of built-in convolution kernels, and colour space conversions.

The library also supports image resizing, rotation and arbitrary affine transforms.

There's a histogram method allowing you to pull some statistics out of an image. This can be used for automatic contrast enhancement, and for global statistical analysis.

4.3.16 proglog==0.1.9

Proglog is a progress logging system for Python. It allows to build complex libraries while giving the user control on the management of logs, callbacks and progress bars.

What problems does it solve ?

Libraries like tqdm or progress are great for quickly adding progress bars to your scripts, but become difficult to manage when building larger projects.

For instance, you will need to write different code depending on whether you are displaying the progress in a console, a Jupyter notebook, or a webpage.

Sometimes a single program may have to handle many logs and progress bars coming from different subprograms and libraries, at which case you may want to let the final user decide which progress bars they want to display or to mute, even when these progress bars are handled deep down in your program.

For instance if your program 1 calls a program 2 and program 3 (possibly from other libraries), you may want to be able to silence the progress bars of routine 2, or to only show the progress bars of routine 1. As long as all routines use Proglog, this will be easy to do.

Fig 4.3.2: proglog

4.3.17 requests==2.24.0

Requests is a simple, yet elegant HTTP library.

Requests allows you to send HTTP/1.1 requests extremely easily. There's no need to manually add query strings to your URLs, or to form-encode your POST data. Keep-alive and HTTP connection pooling are 100% automatic, thanks to urllib3.

4.3.18 tqdm==4.51.0

tqdm is a Python library that allows you to output a smart progress bar by wrapping around any iterable. A tqdm progress bar not only shows you how much time has elapsed, but also shows the estimated time remaining for the iterable.

4.3.19 urllib3==1.25.11

urllib3 is a powerful, user-friendly HTTP client for Python. Much of the Python ecosystem already uses urllib3 and you should too. urllib3 brings many critical features that are missing from the Python standard libraries:

Thread safety.

- Connection pooling.
- Client-side SSL/TLS verification.
- File uploads with multipart encoding.
- Helpers for retrying requests and dealing with HTTP redirects.
- Support for gzip, deflate, and brotli encoding.

- Proxy support for HTTP and SOCKS.
- 100% test coverage.

4.3.20 wincertstore==0.2

wincertstore provides an interface to access Windows' CA and CRL certificates. It uses ctypes and Windows's sytem cert store API through crypt32.dll. wincertstore 0.1 used to return all certificates although some are not suitable to verify TLS/SSL server certificates. wincertstore 0.2 only returns certificates for SERVER_AUTH enhanced key usage by default.

CHAPTER 5

WORK PLAN

Table 1: WorkPlan

To Do	*Done By*	*Rough Estimate of Finishing*
Dataset Collection	Nirmal Das	October(Completed)
Object Identification(Image Frame)	Krupa Bose	October(Completed)
Object Identification(Video Frame)	Steffi Johnson	December(Completed)
Object Counting	Lijo Joy	December(Completed)
Feature Extraction	Nirmal,Lijo	January(Completed)
Face Recognition	Lijo,Nirmal,Krupa,Steffi	February(Completed)
Final Result	Lijo,Nirmal,Krupa,Steffi	March(Completed)

CHAPTER 6

SAMPLE SCREENSHOTS

6.1 DATASET

Fig 6.1.1 Dataset

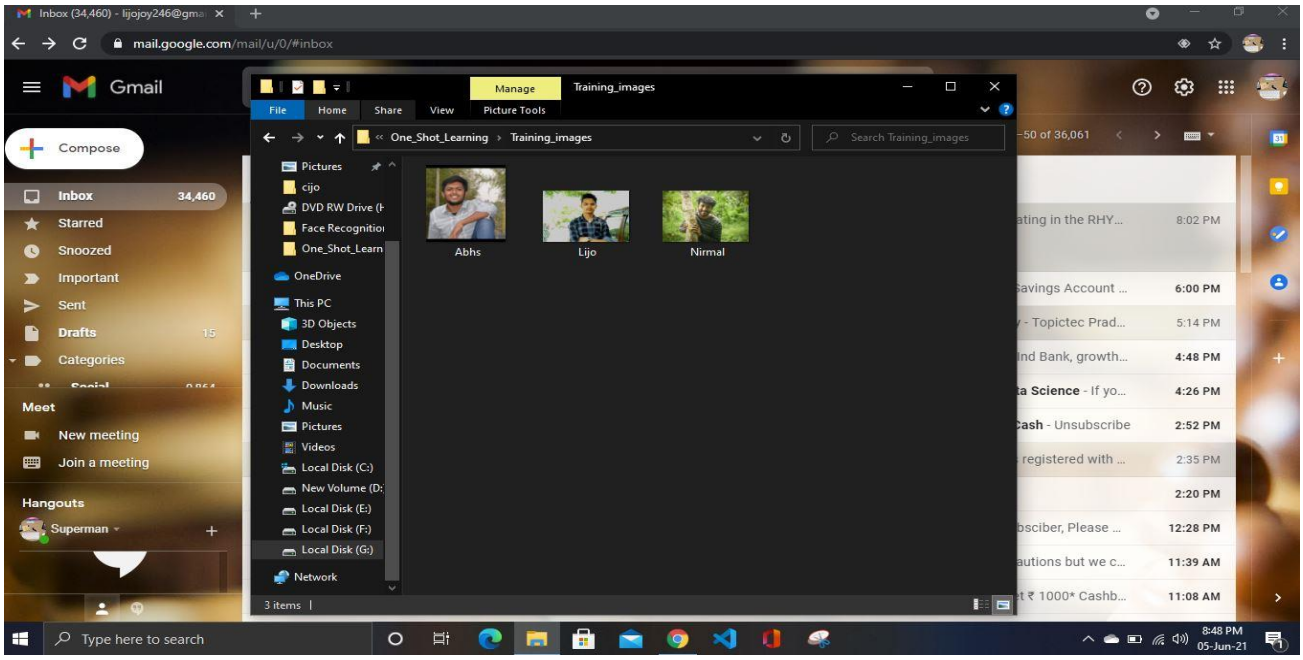


Fig : 6.1.2 Training Dataset

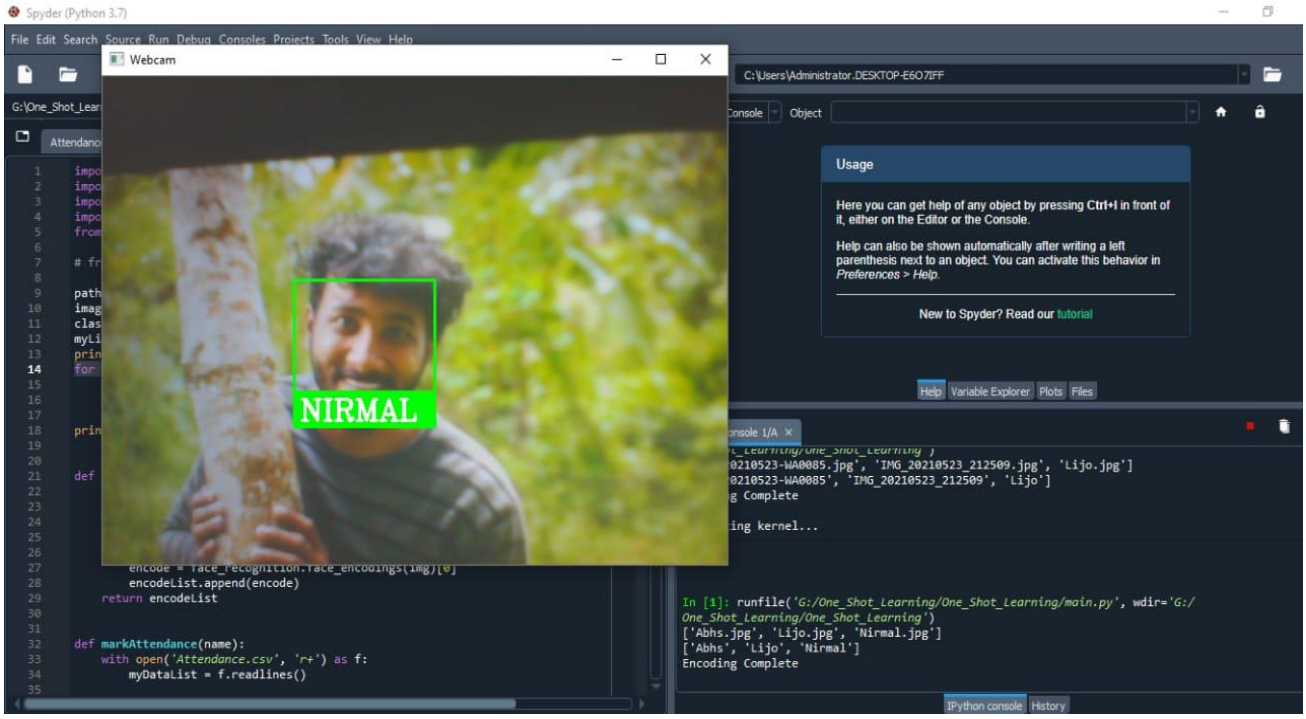


Fig : 6.1.3 Face Recognition

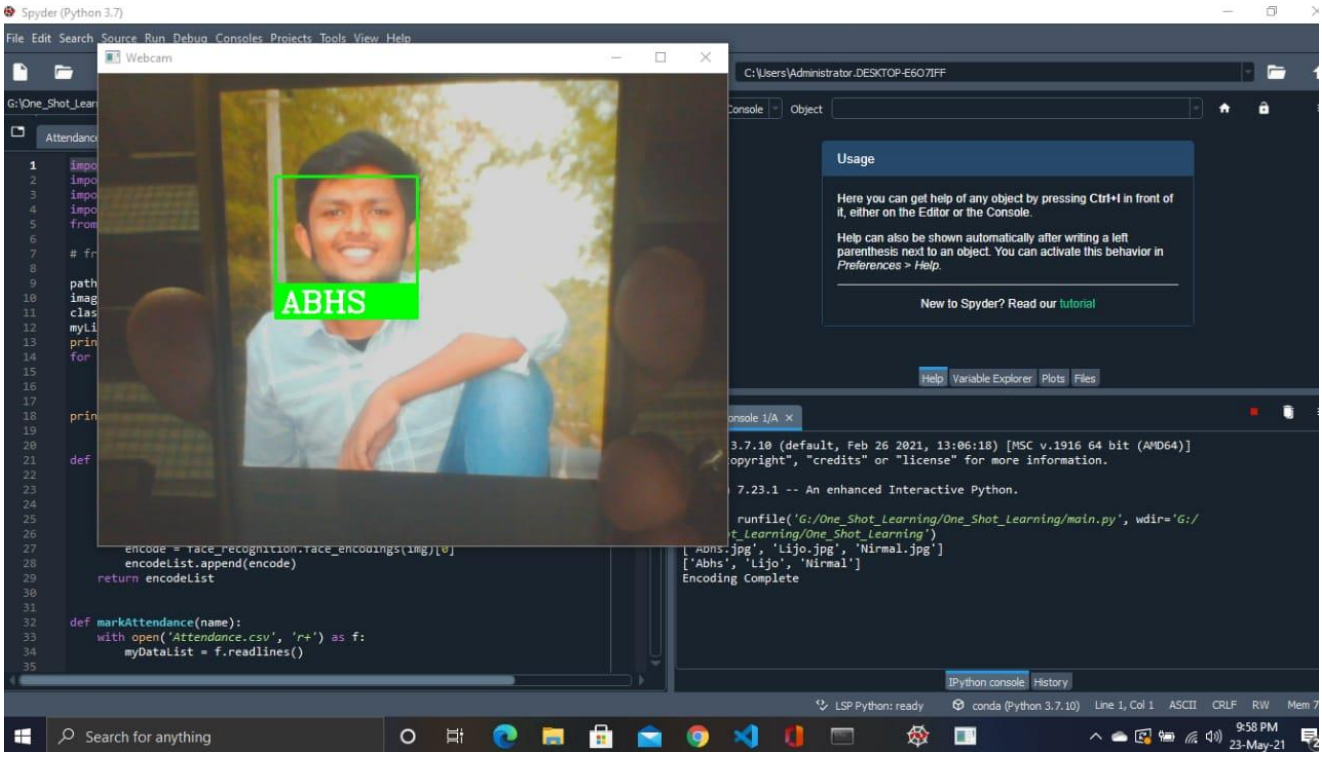


Fig : 6.1.4 Face Recognition

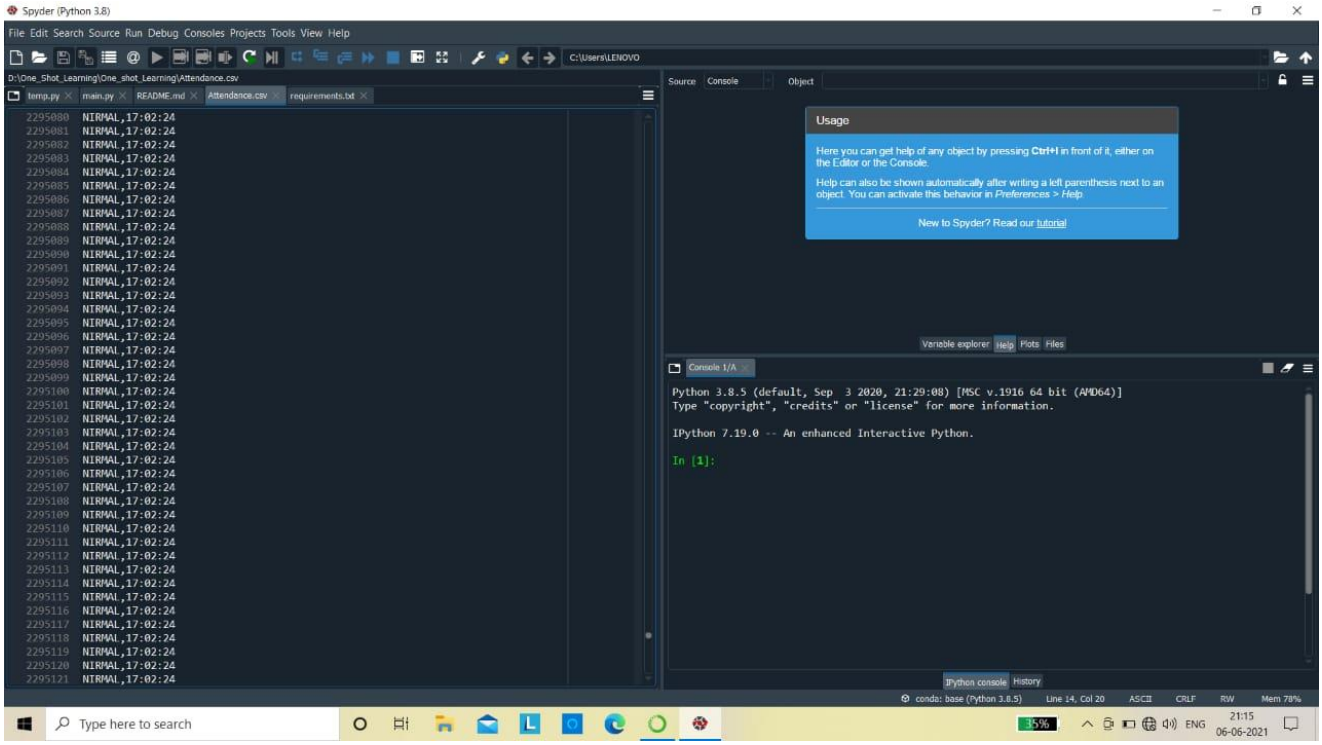


Fig : 6.1.5 People Counting

6.2 SOURCE CODE

```
import cv2

import numpy as np

import face_recognition

import os

from datetime import datetime



# from PIL import ImageGrab



path = 'Training_images'

images = []

classNames = []

myList = os.listdir(path)

print(myList)

for cl in myList:

    curImg = cv2.imread(f'{path}/{cl}')

    images.append(curImg)

classNames.append(os.path.splitext(cl)[0])
```

```
print(classNames)


def findEncodings(images):
    encodeList = []



    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)
    return encodeList



def markA(name):
    with open('A.csv', 'r+') as f:
        myDataList = f.readlines()
```

```
    nameList = []
        for line in myDataList:
            entry = line.split(',')
            nameList.append(entry[0])
            if name not in nameList:
                now = datetime.now()
                dtString = now.strftime('%H:%M:%S')
                f.writelines(f'\n{name},{dtString}')
    #### FOR CAPTURING SCREEN RATHER THAN WEBCAM
    # def captureScreen(bbox=(300,300,690+300,530+300)):
    #     capScr = np.array(ImageGrab.grab(bbox))
    #     capScr = cv2.cvtColor(capScr, cv2.COLOR_RGB2BGR)
    #     return capScr

    encodeListKnown = findEncodings(images)
    print('Encoding Complete')
cap = cv2.VideoCapture(0)
```

```
  while True:
    success, img = cap.read()

# img = captureScreen()
    imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

    facesCurFrame = face_recognition.face_locations(imgS)
    encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)

    for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
        matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
        faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
# print(faceDis)
        matchIndex = np.argmin(faceDis)

        if matches[matchIndex]:
          name = classNames[matchIndex].upper()
```

```
# print(name)
            y1, x2, y2, x1 = faceLoc
            y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
            cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)


 cv2.putText(img, name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
            markA(name)


   cv2.imshow('Webcam', img)
   cv2.waitKey(1)
```

CHAPTER 7

CONCLUSION

In this study, we have considered a number of classical and CNN-based object recognition techniques for real-time video analytics, and have developed a software platform implementing methods suitable for deployment for crowd monitoring in public spaces. Real-world validation of our solution has been carried out with both controlled and non-controlled tests, and the results strongly indicate good accuracy, even in outdoor conditions. This yields great confidence in expanding the deployment of the developed system into other venues. Our proposed automated video analytics for crowd monitoring and tracking will enable significant manpower savings, especially in key security-sensitive installations such as public transport facilities and protected areas, where CCTV monitoring is oftentimes performed by human operators. Data collection of crowd density and movement can be performed more consistently and with better accuracy than otherwise achievable with manual monitoring. It is noted that the software solution developed here can accept multiple video streams from a centralized storage location, suitable for operation in facilities management or public spaces with multiple installed security cameras. Other useful applications of the current framework may include utilization in factories to detect personnel in restricted places or unsafe proximity to equipment. Further extensions of the current framework may include layering facial identification on top of the current object recognition and tracking for enhanced surveillance capabilities, or to configure recognition for potentially dangerous items such as knives or firearms in the fight against terrorism, to enhance commuter safety and security.

REFERENCES

[1] Kang Hao Cheong, Sandra Poeschmann, Joel Weijia Lai, U.Rajendra Acharya, Simon Ching Man Yu and Kenneth Jian Wei Tang ‘‘Practical Automated Video Analytics for Crowd Monitoring and Counting’’ IEEE Access. vol. 7, Dec. 2019.

[2] T. Ko, ‘‘Proposed System for Criminal Dectection and Recognition on CCTV data using Machine Learning,’’ in Proc. 37th IEEE Appl. Imag. Pattern Recognit. Workshop, Oct. 2019, pp. 1–8.

[3] A. C. Caputo, Digital Video Surveillance and Security. Oxford, U.K.: Butterworth-Heinemann, 2014.

[4] P. Remagnino, G. A. Jones, N. Paragios, and C. S. Regazzoni, VideoBased Surveillance Systems: Computer Vision and Distributed Processing. Boston, MA, USA: Springer, 2002.

[5] T. P. Chen, H. Haussecker, A. Bovyrin, R. Belenov, K. Rodyushkin, A. Kuranoc, and V. Eruhimov, ‘‘Computer vision workload analysis: Case study of video surveillance systems,’’ Int. Technol. J., vol. 9, no. 2, pp. 109–118, 2005. 183258 VOLUME 7, 2019 K. H. Cheong et al.: Practical Automated Video Analytics for Crowd Monitoring and Counting

[6] S. A. M. Saleh, S. A. Suandi, and H. Ibrahim, ‘‘Recent survey on crowd density estimation and counting for visual surveillance,’’ Eng. Appl. Artif. Intell., vol. 41, pp. 103–114, May 2015.

[7] L. Zhang, M. Shi, and Q. Chen, ‘‘Crowd counting via scale-adaptive convolutional neural network,’’ in Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), Mar. 2018, pp. 1113–1121.

[8] H. Liu, S. Chen, and N. Kubota, ‘‘Intelligent video systems and analytics: A survey,’’ IEEE Trans. Ind. Informat., vol. 9, no. 3, pp. 1222–1233, Aug. 2013.

[9] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, ‘‘Real-time video analytics: The killer app for edge computing,’’ Computer, vol. 50, no. 10, pp. 58–67, 2017.

[10] A. A. Adams and J. M. Ferryman, ‘‘The future of video analytics for surveillance and its ethical implications,’’ Secur. J., vol. 28, no. 3, pp. 272–289, 2015.

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 6, 2019, accepted December 2, 2019, date of publication December 6, 2019,
date of current version December 27, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2958255

Practical Automated Video Analytics for Crowd Monitoring and Counting

KANG HAO CHEONG[1,2], (Member, IEEE), SANDRA POESCHMANN[3], JOEL WEIJIA LAI[1,2], JIN MING KOH[1], U. RAJENDRA ACHARYA[4], SIMON CHING MAN YU[5], AND KENNETH JIAN WEI TANG[1]

[1]Science and Math Cluster, Singapore University of Technology and Design (SUTD), Singapore 487372
[2]SUTD-MIT International Design Centre, Singapore 487372
[3]Engineering Cluster, Singapore Institute of Technology, Singapore 138683
[4]Department of Electronics and Computer Engineering, Ngee Ann Polytechnic, Singapore 599489
[5]Interdisciplinary Division of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong

Corresponding author: Kang Hao Cheong (kanghao_cheong@sutd.edu.sg)

This work was supported by the SUTD-MIT International Design Centre (IDC) under Grant IDG21900101 and Grant IDIN19001.

ABSTRACT Video surveillance is gaining popularity in numerous applications, including facility management, traffic monitoring, crowd analysis, and urban security. Despite the increasing demand for closed-circuit television (CCTV) and related infrastructure in public spaces, there remains a notable lack of readily-deployable automated surveillance systems. In this study, we present a low-cost and efficient approach that integrates the use of computational object recognition to perform fully-automated identification, tracking, and counting of human traffic on camera video streams. Two software implementations are explored and the performance of these schemes is compared. Validation against controlled and non-controlled real-world environments is also demonstrated. The implementation provides automated video analytics for medium crowd density monitoring and tracking, eliminating labor-intensive tasks traditionally requiring human operation, with results indicating great reliability in real-life scenarios.

INDEX TERMS Crowd monitoring, counting, traffic monitoring, data analytics, background subtraction, security.

I. INTRODUCTION

Video surveillance is an integral component of modern urban security, and when coupled with computational analytics, can have greatly expanded functionality including facial recognition, motion detection, traffic and crowd monitoring, and automated hazard alarms [1]–[7]. The continued advancement in computational tools and machine learning has in principle enabled automation of a wide variety of practical analyses on image and video inputs [8]–[14]; more advanced machine intelligence systems are also increasingly capable of fulfilling traditionally human-controlled tasks that require real-time complex decisions, for instance initiating mitigation measures for severe traffic congestion or the dispatching of emergency services [15]–[18]. In general, automated surveillance eliminates the need for round-the-clock manual monitoring, thereby reducing manpower requirements [19]. This stands to yield operational cost reductions and productivity improvements.

Nonetheless, amidst the progressing state-of-the-art, integration of automated analytics in commercial video surveillance for crowd monitoring and counting is an area that can be further explored [19]; and there is at present limited literature on demonstrated effective low-cost systems for deployment. In security and management sectors, there remains a great reliance on traditional manual monitoring of CCTV footage [20], [21], and human patrols to conduct crowd monitoring and tracking. Utilizing computer vision and real-time automated analytics in replacement of manual labour not only reduces operational costs but also eliminates human errors and lapses [22]–[24] —we seek to develop a viable deployment-ready implementation in this study.

In this paper, we examine several viable approaches to automated crowd monitoring and tracking in indoor and outdoor scenarios, ultimately selecting a statistical background subtraction (BGS) scheme and a convolutional neural

The associate editor coordinating the review of this manuscript and approving it for publication was Dongxiao Yu.

network-based single shot detector (SSD). These methods are easily deployable in the real world. A software solution is developed for use in general public spaces, and we validate the performance of the platform through indoor controlled tests in a shopping mall, and outdoor non-controlled tests in a public transport hub with considerable human traffic. It is noted that the implementation of this video analytics system can also be applied to a broader range of scenarios—for instance, in factories to detect personnel in restricted places or in dangerous proximity to machinery, or in high-rise buildings to detect crowd densities that exceed safe thresholds for timely evacuation in case of emergencies. Time-oriented data collected from such deployments can also be logged and transmitted to a dashboard for data-informed planning and predictive analytics [25].

The structure of the paper is as follows—a technical review is first provided (Section II), followed by a discussion on the methodology employed and the development of the software solution (Section III), and finally validation test results (Section IV) and concluding remarks (Section V).

II. TECHNICAL REVIEW

We first provide an overview of object recognition frameworks, the challenges associated with achieving satisfactory performance, and the application of these frameworks in automated video analytics. A fundamental operational requirement of automated video surveillance analytics is the ability to identify and track different objects within the recorded footage, hence the need for object recognition; in the current context of crowd analysis, recognition of human subjects is critically relevant.

While visual recognition and classification of objects is intuitive to human perception, robust computational implementation is challenging [26]. Varying exposure to outdoor conditions, changing illumination levels and direction, intermittent and sustained visual obstruction, and unpredictable movement of tracked subjects must all be overcome for reliable operation of recognition systems, oftentimes with limitations on available computational power [27]–[30]. The resolution and clarity of available video footage is also typically non-ideal, limiting the effectiveness of pre-processing techniques aimed at compensating for variance in image conditions. In our context of recognition and tracking of human subjects, additional complexities arise from the wide range of possible dynamical behaviour—for instance, two persons in physical contact may be detected as a single entity, and the shape profile of a person may change drastically because of carried items or differing attire.

Advanced machine vision systems are already being developed for security- and safety-critical applications, such as driverless vehicles and autonomous drones [31]–[35]. These systems typically employ convolutional neural network (CNN)-based solutions that are trained on massive datasets of numerous modalities, including infrared and visible video input, lidar data, sound pick-ups from microphones, and navigational data from GPS or inertial guidance. Existing studies have shown excellent performance in the identification of key markers, such as lane boundaries, traffic signs, and pedestrians on systems intended for driverless vehicles [36]–[39] under a wide range of lighting and driving conditions. CNN-based image recognition has also been applied very successfully to facial identification tasks [40]–[44], achieving large reductions in error rate when compared to non-CNN methods. These types of CNN-based methods are presently employed for automated user identification and tagging systems in prominent social media platforms [45].

In general, CNN-based systems are hugely robust to changing background conditions and object appearances, but are typically computational expensive to train and run. In comparison to explicit rule-based or statistical methods, the employed CNN architectures are also more akin to black boxes and offer limited tractability—troubleshooting and tuning the systems for specific environments can therefore be challenging. There is also recent evidence attributing the efficacy of neural network deep-learning solutions to fine-tuning rather than a fundamental architectural advantage, suggesting that a properly tuned classical method may be able to achieve similar performance in certain scenarios [46]–[50].

Indeed, non-CNN methods have been deployed to perform similar tasks. A real-time system for pedestrian tracking using gray-scale images from stationary cameras has been demonstrated [51], with satisfactory robustness to visual occlusions and ambiguities in perceived subject shape profiles. The implementation relied on Gaussian-mixture foreground masking followed by contour detection through a principle component analysis (PCA) model. Numerous studies on pedestrian and traffic tracking, motion detection and analysis, and object classification using non-CNN methods have also been presented to date [30], [52]–[55], suggesting good viability in these approaches. Non-CNN methods may hence be preferred in some scenarios.

III. METHODOLOGY

The software package developed in this study comprises a video processing back-end encompassing human subject recognition and tracking, and a front-end graphical interface for operators. The software implementation is broadly discussed in Section III-A, with object recognition methods in Sections III-B1–III-B2, and lastly tracking and counting techniques in Sections III-C–III-D. A block diagram summarizing the video tracking and counting process is given in Figure 1.

A. SOFTWARE IMPLEMENTATION

Our software package is implemented on *Python* with the Open Source Computer Vision (*OpenCV*) library. *OpenCV* supports machine deep-learning frameworks, and provides image manipulation, object identification, and motion tracking tools that are greatly relevant for the development of software in our context [56], [57]. Our specific implementation assumes a pre-existing video surveillance system that writes to a centralized storage pool, from which footage may be

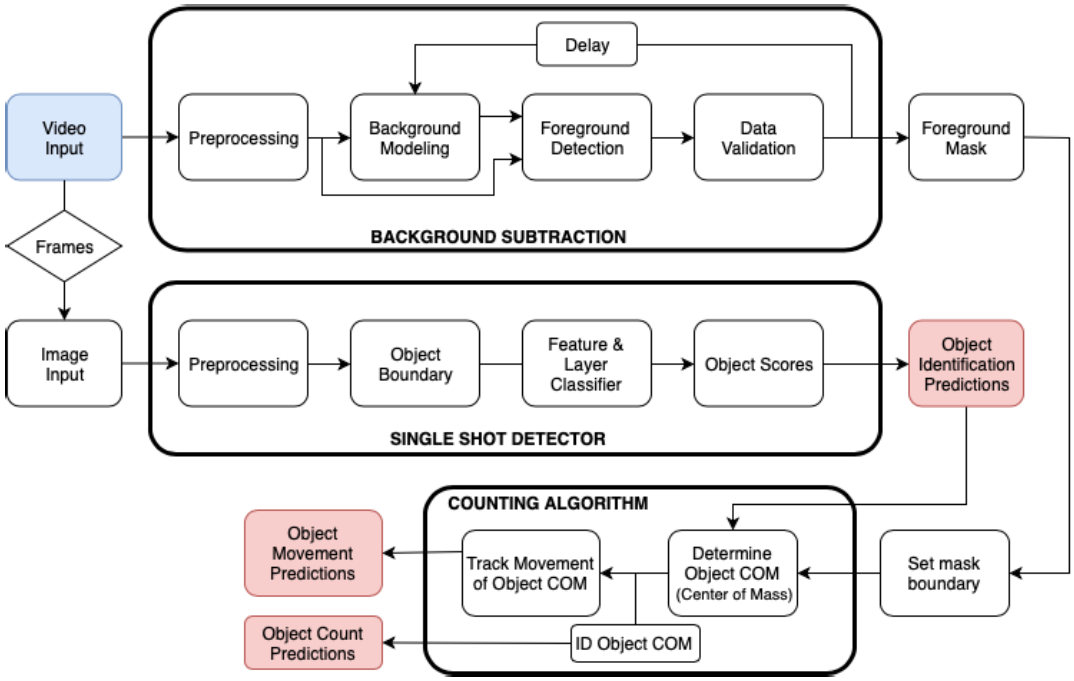


FIGURE 1. Block diagram depicting data flow in the adopted video analytics pipeline. The input undergoes either BGS or SSD, yielding human personnel identification and count.

pulled in real-time for analysis; as such all functionalities are developed and tested in a stream-based format. The software implementation accommodates video streams of general frame size and rate, but we use footage of 720p at 30 fps for illustrative purposes in this paper, unless otherwise stated.

B. OBJECT RECOGNITION

We examine two categories of object recognition methods in detail—background subtraction and CNN-based image classifiers. A comparison of the performance between chosen variants of these two methods in controlled and non-controlled test environments is later presented in Section IV.

1) BACKGROUND SUBTRACTION

A widely used method for detecting moving objects from a stationary camera placement is background subtraction (BGS) [58]. In general, the operation of such a method relies on a known background frame with no present objects. This background reference is then subtracted from each frame of the video footage, or subset of frames to reduce computational cost, therefore yielding frames containing only foreground objects. Appropriate contour detection or region segmentation models can then be applied to isolate distinct objects in these frames. An example illustrating the operation of BGS is presented in Figure 2. BGS-based methods are presently applied in commercial video surveillance systems for malls and public spaces.

Simplistic implementations of BGS typically suffer from limited reliability, due mostly to changing background conditions. In an outdoor environment, volatile weather, illumination changes, and reflections from surfaces on moving objects can all diminish the ability of the reference frame subtraction to separate background and foreground elements. A number of methods to overcome these problems have been utilized to date. Pre-processing of footage frames to remove glare and illumination changes can be utilized; major changes in background and ambient conditions can be detected through regression against a history of frames, and the background reference either adjusted or recaptured at opportune times; a comprehensive set of background references can also be captured *a priori* against possible ambient conditions, selections of which are subtracted from footage frames on a trial basis until a sufficiently clean output is produced. Movement patterns of detected objects across numerous frames can also be used as an additional filter against false positives—for instance, human subjects must realistically be in contact, or otherwise close proximity, with the ground at all times.

An early variant of a BGS-based object recognition framework is the Mixture of Gaussians (MOG) method introduced in 2001, utilizing a Gaussian mixture background/foreground segmentation algorithm [59], [60]. An improved version, named MOG2, was later presented, with a significant improvement being an automatic selection scheme for the number of Gaussian kernels used for each pixel, in place of the constant number of distribution kernels in the original MOG [61], [62]. As a result, MOG2 provides better adaptability to changing illumination conditions in scenes. A more recent algorithm is the GMG [63], named after its founders, which combines statistical background image estimation and per-pixel Bayesian segmentation. GMG uses the first few hundred frames of the input footage to construct

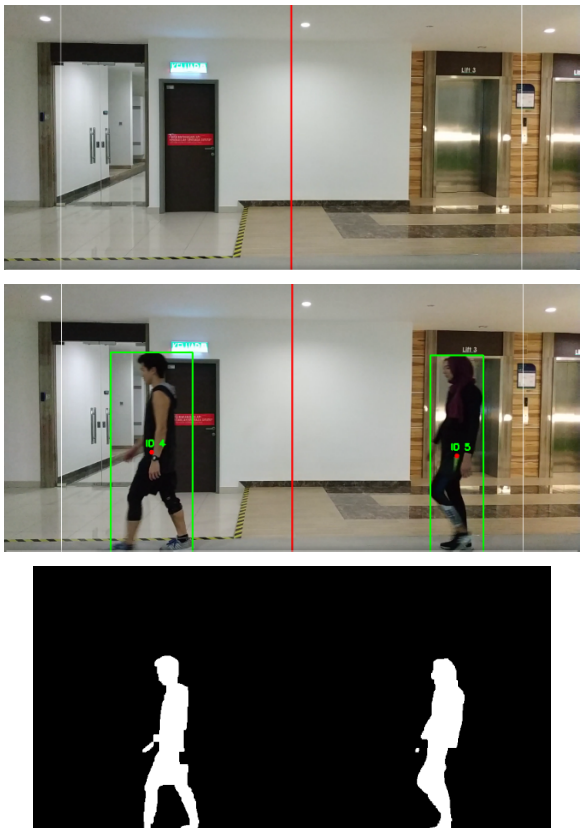


FIGURE 2. Illustration of the functioning of BGS. (a) An empty scene as the background mask; (b) a scene with human subjects in the foreground; and (c) the scene after background subtraction, separating the human subjects.

FIGURE 3. Frame comparison of the MOG, MOG2 and GMG background subtraction schemes, showing a cleaner result from MOG2.

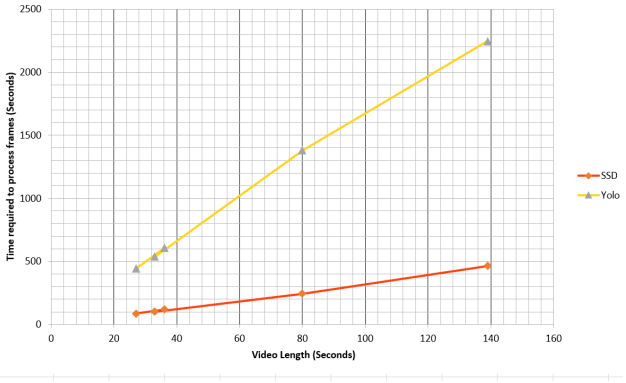


FIGURE 4. Pilot study results comparing the performance of YOLO and SSD object recognition schemes, suggesting vastly cheaper computation using SSD.

a background model, which is then used for backgound subtraction.

We note that GMG is limited in suitability for outdoor scenes with constant motion of objects, as there is often no dedicated time periods available for background capture. Deploying GMG in the intended use case of automated crowd analysis in busy public spaces is therefore not viable. On the other hand, there is evidence in existing literature that MOG2 provides good results in practice [64], [65], and is also sufficiently computationally cheap to deploy in large-scale video analysis applications. Our own pilot programme also suggests that MOG2 consistently produces better human subject identification and segmentation results than MOG and GMG in indoor use cases (example in Figure 3). We therefore employ MOG2 in our BGS-based implementation.

A three-step processing pipeline is utilized for incoming video frames. First, a brightness and colour-correction filter is applied to adjust for under- or over-exposure, or changing daylight conditions. Noise reduction is also used to improve image quality. Next, a background subtraction mask is computed and applied through MOG2. Lastly, contours on the masked image are identified through contrast segmentation, to extract the bounding boxes and positions of the human subjects. This process is computationally cheap, but face potential limitations—detection accuracy is compromised if BGS is not completely successful due to background fluctuations, and differentiation between human and non-human subjects may not be ideal.

2) CONVOLUTIONAL NEURAL NETWORKS

We consider CNN-based recognition frameworks You-Only-Look-Once (YOLO) [66], a state of the art real-time object detection system shown to be capable of identifying and classifying objects effectively, and the Single Shot Detector (SSD) [67], also a highly-established method. Both of these frameworks run the full incoming frames through a CNN in a region-wise manner to yield bounding boxes and class probabilities on identified objects. These CNNs are pre-trained on large imagery datasets. A pilot programme had been carried out to compare the computational running cost of YOLO and SSD on preliminary hardware (Figure 4), revealing that SSD is considerably faster in processing incoming frames, and is therefore more viable in achieving real-time stream-based automated video analysis with. We choose SSD as the preferred CNN-based solution. We utilize *MobileNet* supported on the deep neural network (DNN) module of *OpenCV* for SSD implementation. *MobileNet* provides pre-trained CNNs for image classification, and can robustly handle frames of different aspect ratios and sizes.

FIGURE 5. Recognition of a moving human subject through SSD.

Specifically, the model utilized in this paper is an implementation of Google's *MobileNet SSD* [68], which was initially trained on the Common Objects in Context (COCO) dataset [69]. The model was further refined on PASCAL VOC0712 [68]. A sample snapshot of the execution of this implementation on real-world footage is presented in Figure 5, on which a moving human subject with headwear is identified with $> 99\%$ confidence.

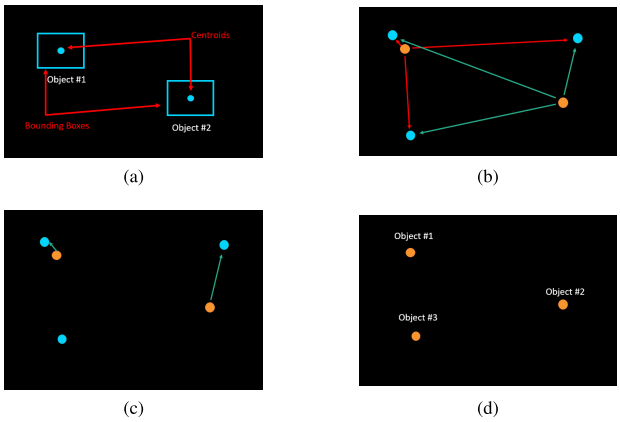


FIGURE 6. Illustration of the subject tracking mechanism. (a) Two subjects are recognized in the scene and labelled with distinct identification numbers, and the centroids of their bounding boxes are computed; (b) the subjects have moved in the next frame, and the new positions are compared against the previous; (c) identifications are made on a nearest-distance basis; (d) identification numbers are consistently applied, with a new subject deduced to have entered the scene. Images adapted from [70].

C. TEMPORAL TRACKING

Analyzing incoming video streams frame-wise does not guarantee temporal continuity in the identified subjects, and therefore cannot immediately support counting functions. In order to support reliable counting of moving subjects, a temporally-consistent labelling of subjects must be achieved between frames, such that distinct objects are not misidentified as being identical (leading to under-counting), and identical objects are not misidentified as being distinct (leading to over-counting). In essence, objects undergoing movement has to be continuously tracked across all frames in which they appear.

We implement this by comparing the centroids of identified bounding boxes for each frame against those of the previous, and labelling pairs as identical on a nearest-distance basis. Centroids in the previous frame that have no matching counterpart in the current are deemed to have left the scene, and centroids in the current frame that have no match in the previous are deemed as new subjects that have entered. This is illustrated in Figure 6. To cope with subject occlusions and intermittent image quality issues, a loss-of-visibility threshold of $n_{lv} = 18$ frames is set, such that if a subject disappears from view within the scene and reappears within n_{lv} frames in a location deemed to be matching by the nearest-distance scheme, a new subject identification will not be assigned and the reappeared subject will be considered identical to the previous. A movement rate threshold can also be set (say, to the typical human running speed), such that subjects that move faster than expected between frames are identified as distinct.

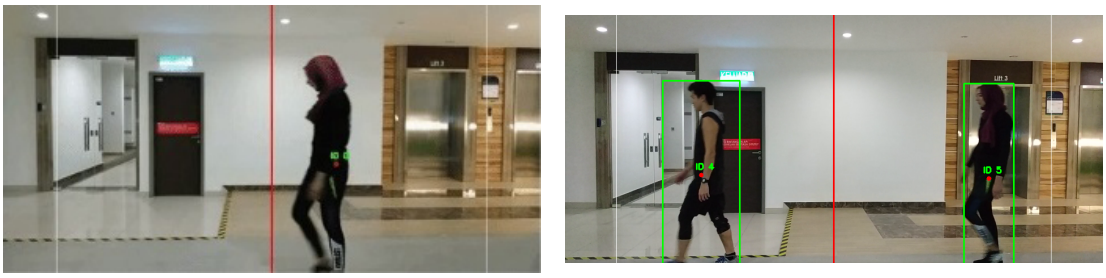
D. SUBJECT COUNTING

The counting of identified subjects can be set to be scene-wise—subjects are counted the moment they enter the scene captured by the video camera, and real-time on-scene subject counts are recorded together with cumulative counts (starting from a specified time, say, the start of each day). This mode is useful if on-scene subject counts are of interest, for instance, to monitor the number of people in an enclosed space, or congestion conditions along corridors and passageways. Alternatively, counting can be set to be portal-wise, that is, subjects are added to a cumulative count only when they cross a specified boundary in the captured scene. This mode is useful to monitor crowd influx or outflux through key doorways or area perimeters, for instance, in tracking the boarding of public buses or commuter movement through security checkpoints.

In the scene-wise mode, extremal boundaries can be set by the user on the video scene, only within which counting is active. Subject tracking is maintained throughout the entire scene regardless of boundary settings, however, so that identical subjects repeatedly crossing the set boundaries will not be misidentified as distinct instances, so long as they remain within the scene throughout. In the portal-wise mode, the user may select from a default preset of 10 regularly spaced boundary lines, or freely draw a desired boundary. Movement direction filtering can also be set, such that only subjects moving to the left or right are counted.

E. DATA PROCESSING

It is important to keep the analysis low-cost, as practical deployment hardware may be limited in speed, especially when there are multiple video streams sharing compute time. To reduce computational load, the program can be configured to perform BGS or SSD object identification only every N_0 frames; in our implementation, subject tracking and counting is set to occur every $N_0 = 6$ frames, corresponding to a $\sim$ 200 ms refresh rate. These limits were found to be sufficient in providing good tracking results for typical pedestrian traffic encountered in our test environments (Section IV), and can



(a) (b)

FIGURE 7. Video snapshots of the controlled environment tests, for (a) a constrained case of a single subject in-scene at any point in time, and (b) multiple subjects in-scene simultaneously.

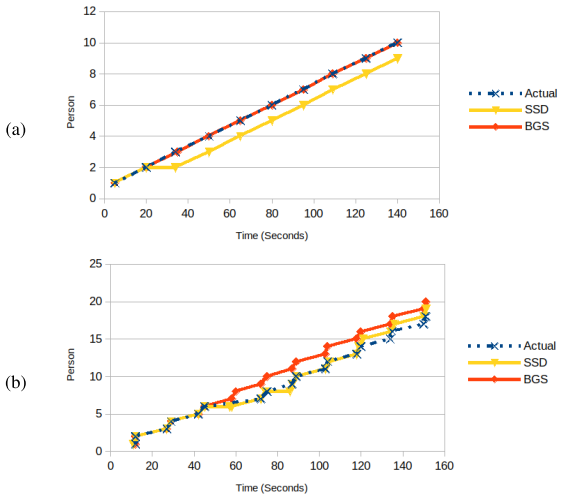


FIGURE 8. Subject counting results in the controlled environment comparing BGS and SSD methods, in (a) the constrained case of a single subject in-scene at any point in time, and (b) with multiple subjects in-scene simultaneously. The actual counts were obtained through manual counting by watching the same video footage, matched against an on-site surveyor for consistency.

be adjusted for different hardware capabilities. While such an approach effectively reduces the imposed computational load per video stream, the trade-off between practicality and accuracy stands to be further characterized; an important line of development for future work is also in studying alternative approaches that does not impact analysis frame rate.

F. COMPUTATIONAL RESOURCE

Reasonable computational cost is a requisite for viable deployability in the real-world—a considerably modest workstation was hence used for testing purposes. The workstation was a laptop equipped with an Intel I5 8250U quad-core processor at 1.6 GHz (base), 8 GB of RAM, and an Intel UHD Graphics 620 graphics processor, running the Ubuntu (Linux) operating system. On this platform, the SSD method runs in real-time on the CPU.

IV. RESULTS & DISCUSSION

The developed video analysis software was tested in two types of environments—first a controlled environment (Section IV-A), and then a non-controlled environment (Section IV-B) for validation. In these validation tests, portal-wise subject counting mode was utilized.

A. CONTROLLED ENVIRONMENT

An indoor area in a shopping mall, in proximity to a lift lobby, was used as an indoor controlled environment. A number of subjects, dressed in varying attire, was sent to walk across the captured space at varying speeds, ranging from a slow walk typical of the elderly to fast jogs. These subjects are of mixed genders and of varied heights. The controlled studies were conducted with steady artificial lighting and primarily constant environmental parameters; each test lasted a duration of 150 seconds with both the BGS and SSD methods, and a manual on-site count was performed simultaneously to match the results against. Snapshots of the controlled validation tests are shown in Figure 7, and subject counting results are presented in Figure 8. These results indicate satisfactory counting accuracy for both BGS and SSD methods, with BGS notably achieving perfect accuracy in the idealized single-subject scenario, but is ultimately outperformed by SSD in more realistic multiple-subject scenarios. The SSD method yielded a maximum of a single miscount in these controlled tests, suggesting good deployment viability in the significantly more demanding non-controlled outdoor environments.

B. NON-CONTROLLED ENVIRONMENT

Non-controlled validation tests were performed at a public transport hub with considerable human traffic. The camera placements were chosen for the purpose of tracking commuter volume boarding public buses at various terminals, suitable for assessing the ability of the system to cope with massive crowd surges. The tests were conducted over typical bus-boarding durations of approximately 20 seconds, with both BGS and SSD used, and a simultaneous manual count to match results against. Sample snapshots showing the recorded environments are presented in Figure 9, and a comparison of subject counting results is given in Figure 10.

It was observed that SSD yields $> 92\%$ accuracy (mean square error), illustrating the robustness of SSD in handling large crowd densities and volatile outdoor illumination conditions. The obvious failure of BGS in comparison to the controlled tests can be attributed to its inability to handle rapidly

(a) (b) (c)

FIGURE 9. Video snapshots of the non-controlled environment tests. (a) Full scene captured by the camera; (b) zoomed snapshot of boarding queue onto a public bus; and (c) zoomed snapshot of the queue at a later point in time.

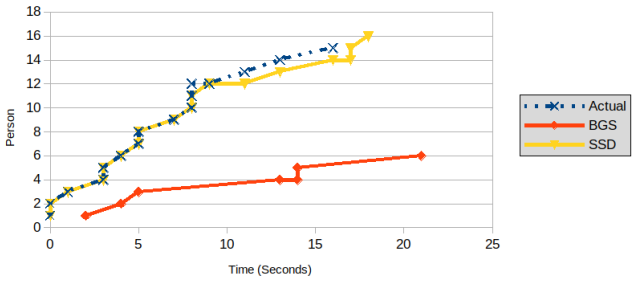


FIGURE 10. Subject counting results in the non-controlled environment, comparing BGS and SSD methods. The actual counts were obtained through manual counting by watching the same video footage, matched against an on-site surveyor for consistency.

FIGURE 11. Separated foreground elements by BGS, corresponding to the input frame shown in Figure 9(c). The fragmented masking of human subjects is clearly observed.

varying backgrounds. We illustrate this in Figure 11, in which the imprecise and fragmented masking of overlapping subjects by BGS can be seen. The MOG2 implementation of BGS statistically constructs a background mask from a subsample of video frames, and is thus theoretically able to readjust for changing background conditions; but in this real-world deployment in a transport hub, there is insufficient time for such a mechanism to work as intended. With background masks of inadequate quality, multiple subjects in close proximity are frequently misidentified as a single subject, hence resulting in the severe under-counting. It is thus obvious that the SSD implementation is greatly more suitable for use, especially in places of significant human traffic.

V. CONCLUSION

In this study, we have considered a number of classical and CNN-based object recognition techniques for real-time video analytics, and have developed a software platform implementing BGS and SSD methods suitable for deployment for crowd monitoring in public spaces. Real-world validation of our solution has been carried out with both controlled and non-controlled tests, and the results strongly indicate good accuracy of the SSD system, even in outdoor conditions. This yields great confidence in expanding the deployment of the developed system into other venues.

Our proposed automated video analytics for crowd monitoring and tracking will enable significant manpower savings, especially in key security-sensitive installations such as public transport facilities and protected areas, where CCTV monitoring is oftentimes performed by human operators. Data collection of crowd density and movement can be performed more consistently and with better accuracy than otherwise achievable with manual monitoring. It is noted that the software solution developed here can accept multiple video streams from a centralized storage location, suitable for operation in facilities management or public spaces with multiple installed security cameras. Other useful applications of the current framework may include utilization in factories to detect personnel in restricted places or unsafe proximity to equipment.

Further extensions of the current framework may include layering facial identification on top of the current object recognition and tracking for enhanced surveillance capabilities, or to configure recognition for potentially dangerous items such as knives or firearms in the fight against terrorism, to enhance commuter safety and security.

COMPETING INTERESTS

The authors declare that they have no competing interests.

REFERENCES

[1] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl, and S. Pankanti, "Smart video surveillance: Exploring the concept of multiscale spatiotemporal tracking," *IEEE Signal Process. Mag.*, vol. 22, no. 2, pp. 38–51, Mar. 2005.

[2] T. Ko, "A survey on behavior analysis in video surveillance for homeland security applications," in *Proc. 37th IEEE Appl. Imag. Pattern Recognit. Workshop*, Oct. 2008, pp. 1–8.

[3] A. C. Caputo, *Digital Video Surveillance and Security*. Oxford, U.K.: Butterworth-Heinemann, 2014.

[4] P. Remagnino, G. A. Jones, N. Paragios, and C. S. Regazzoni, *Video-Based Surveillance Systems: Computer Vision and Distributed Processing*. Boston, MA, USA: Springer, 2002.

[5] T. P. Chen, H. Haussecker, A. Bovyrin, R. Belenov, K. Rodyushkin, A. Kuranoc, and V. Eruhimov, "Computer vision workload analysis: Case study of video surveillance systems," *Int. Technol. J.*, vol. 9, no. 2, pp. 109–118, 2005.

[6] S. A. M. Saleh, S. A. Suandi, and H. Ibrahim, "Recent survey on crowd density estimation and counting for visual surveillance," *Eng. Appl. Artif. Intell.*, vol. 41, pp. 103–114, May 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197615000081

[7] L. Zhang, M. Shi, and Q. Chen, "Crowd counting via scale-adaptive convolutional neural network," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 1113–1121.

[8] H. Liu, S. Chen, and N. Kubota, "Intelligent video systems and analytics: A survey," *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp. 1222–1233, Aug. 2013.

[9] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[10] A. A. Adams and J. M. Ferryman, "The future of video analytics for surveillance and its ethical implications," *Secur. J.*, vol. 28, no. 3, pp. 272–289, 2015.

[11] O. Javed and M. Shah, "Tracking and object classification for automated surveillance," in *Computer Vision*, A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, Eds. Berlin, Germany: Springe, 2002, pp. 343–357.

[12] J. Barthélemy, N. Verstaevel, H. Forehead, and P. Perez, "Edge-computing video analytics for real-time traffic monitoring in a smart city," *Sensors*, vol. 19, no. 9, p. 2048, 2019.

[13] M. A. Uddin, A. Alam, N. A. Tu, M. S. Islam, and Y.-K. Lee, "SIAT: A distributed video analytics framework for intelligent video surveillance," *Symmetry*, vol. 11, no. 7, p. 911, 2019.

[14] X. Kang, B. Song, and F. Sun, "A deep similarity metric method based on incomplete data for traffic anomaly detection in IoT," *Appl. Sci.*, vol. 9, no. 1, p. 135, 2019.

[15] M. Naphade, G. Banavar, C. Harrison, J. Paraszczak, and R. Morris, "Smarter cities and their innovation challenges," *Computer*, vol. 44, no. 6, pp. 32–39, Jun. 2011.

[16] R. Sundar, S. Hebbar, and V. Golla, "Implementing intelligent traffic control system for congestion control, ambulance clearance, and stolen vehicle detection," *IEEE Sensors J.*, vol. 15, no. 2, pp. 1109–1113, Feb. 2015.

[17] S. Y. Kim, Y. Jang, A. Mellema, D. S. Ebert, and T. Collinss, "Visual analytics on mobile devices for emergency response," in *Proc. IEEE Symp. Vis. Anal. Sci. Technol.*, Oct./Nov. 2007, pp. 35–42.

[18] S. Kim, R. Maciejewski, K. Ostmo, E. J. Delp, T. F. Collins, and D. S. Ebert, "Mobile analytics for emergency response and training," *Inf. Vis.*, vol. 7, no. 1, pp. 77–88, 2008.

[19] M. Zabocki, K. Gościewska, D. Frejlichowski, and R. Hofman, "Intelligent video surveillance systems for public spaces—A survey," *J. Theor. Appl. Comput. Sci.*, vol. 8, no. 4, pp. 13–27, 2014.

[20] H. Kruegle, *CCTV Surveillance: Video Practices and Technology*. Amsterdam, The Netherlands: Elsevier, 2011.

[21] H. Keval and M. Sasse, "Man or gorilla? Performance issues with CCTV technology in security control rooms," in *Proc. 16th World Congr. Ergonom. Conf., Int. Ergonom. Assoc.*, 2006, pp. 10–14.

[22] A. Şentaş, I. Tashiev, F. Küçükayvaz, S. Kul, S. Eken, A. Sayar, and Y. Becerikli, "Performance evaluation of support vector machine and convolutional neural network algorithms in real-time vehicle type classification," in *Proc. Int. Conf. Emerg. Internetworking, Data Web Technol.* Cham, Switzerland: Springer, 2018, pp. 934–943.

[23] S. Kul, S. Eken, and A. Sayar, "Evaluation of real-time performance for BGSLibrary algorithms: A case study on traffic surveillance video," in *Proc. 6th Int. Conf. IT Converg. Secur. (ICITCS)*, 2016, pp. 1–4.

[24] S. Kul, S. Eken, and A. Sayar, "Distributed and collaborative real-time vehicle detection and classification over the video streams," *Int. J. Adv. Robot. Syst.*, vol. 14, no. 4, pp. 1–12, 2017.

[25] M. Ulvi, S. Eken, and A. Sayar, "Service oriented visual interpretation tool for time series data," *Anadolu Univ. J. Sci. Technol.-Appl. Sci. Eng.*, vol. 14, pp. 191–198, Jan. 2013.

[26] E. R. Davies, *Machine Vision: Theory, Algorithms, Practicalities*. Amsterdam, The Netherlands: Elsevier, 2004.

[27] L. Tyapi and K. Sowmya, "Real time human detection from video surveillance," *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 3, no. 5, pp. 4413–4417, 2015.

[28] J. C. S. J. Junior, S. R. Musse, and C. R. Jung, "Crowd analysis using computer vision techniques," *IEEE Signal Process. Mag.*, vol. 27, no. 5, pp. 66–77, Sep. 2010.

[29] C. Regazzoni, A. Cavallaro, Y. Wu, J. Konrad, and A. Hampapur, "Video analytics for surveillance: Theory and practice," *IEEE Signal Process. Mag.*, vol. 27, no. 5, pp. 16–17, Sep. 2010.

[30] S. A. Velastin, "CCTV video analytics: Recent advances and limitations," in *Proc. Int. Vis. Informat. Conf.* Berlin, Germany: Springer, 2009, pp. 22–34.

[31] E. D. Dickmanns, "The development of machine vision for road vehicles in the last decade," in *Proc. Intell. Vehicle Symp.*, vol. 1, Jun. 2002, pp. 268–281.

[32] B. Ranft and C. Stiller, "The role of machine vision for intelligent vehicles," *IEEE Trans. Intell. Vehicles*, vol. 1, no. 1, pp. 8–19, Mar. 2016.

[33] M. Slade and F. Marmoiton, "Toward smart autonomous cars," in *Intelligent Transportation Systems*. Boca Raton, FL, USA: CRC Press, 2016, pp. 94–120.

[34] V. Moskalenko, A. Moskalenko, A. Korobov, O. Boiko, S. Martynenko, and O. Borovenskyi, "Model and training methods of autonomous navigation system for compact drones," in *Proc. IEEE 2nd Int. Conf. Data Stream Mining Process. (DSMP)*, Aug. 2018, pp. 503–508.

[35] A. A. Zhilenkov and I. R. Epifantsev, "The use of convolution artificial neural networks for drones autonomous trajectory planning," in *Proc. IEEE Conf. Russian Young Res. Elect. Electron. Eng. (EIConRus)*, Jan./Feb. 2018, pp. 1044–1047.

[36] Y. Satılmiş, F. Tufan, M. Şara, M. Karslı, S. Eken, and A. Sayar, "CNN based traffic sign recognition for mini autonomous vehicles," in *Proc. Int. Conf. Inf. Syst. Archit. Technol.*, J. Świątek, L. Borzemski, and Z. Wilimowska, Eds. Cham, Switzerland: Springer, 2019, pp. 85–94.

[37] M. Hirz and B. Walzel, "Sensor and object recognition technologies for self-driving cars," *Comput.-Aided Des. Appl.*, vol. 15, no. 4, pp. 501–508, 2018.

[38] A. Shustanov and P. Yakimov, "CNN design for real-time traffic sign recognition," *Procedia Eng.*, vol. 201, pp. 718–725, Dec. 2017.

[39] H. Wang, B. Wang, B. Liu, X. Meng, and G. Yang, "Pedestrian recognition and tracking using 3D LiDAR for autonomous vehicle," *Robot. Auton. Syst.*, vol. 88, pp. 71–78, Feb. 2017.

[40] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 98–113, Jan. 1997.

[41] H. Jiang and E. Learned-Miller, "Face detection with the faster R-CNN," in *Proc. 12th IEEE Int. Conf. Autom. Face Gesture Recognit. (FG)*, May/Jun. 2017, pp. 650–657.

[42] S. Guo, S. Chen, and Y. Li, "Face recognition based on convolutional neural network and support vector machine," in *Proc. IEEE Int. Conf. Inf. Automat. (ICIA)*, Aug. 2016, pp. 1787–1792.

[43] X. Sun, P. Wu, and S. C. Hoi, "Face detection using deep learning: An improved faster RCNN approach," *Neurocomputing*, vol. 299, pp. 42–50, Mar. 2018, doi: 10.1016/j.neucom.2018.03.030.

[44] S. Balaban, "Deep learning and face recognition: The state of the art," *Proc. SPIE*, vol. 9457, May 2015, Art. no. 94570B, doi: 10.1117/12.2181526.

[45] K. Hazelwood, S. Bird, D. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, M. Fawzy, B. Jia, Y. Jia, A. Kalro, J. Law, K. Lee, J. Lu, P. Noordhuis, M. Smelyanskiy, L. Xiong, and X. Wang, "Applied machine learning at facebook: A datacenter infrastructure perspective," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2018, pp. 620–629.

[46] W. Brendel and M. Bethge, "Approximating CNNs with bag-of-local-features models works surprisingly well on imagenet," in *Proc. Int. Conf. Learn. Represent.*, 2019. [Online]. Available: https://openreview.net/forum?id=SkfMWhAqYQ

[47] F. Hu, G.-S. Xia, J. Hu, and L. Zhang, "Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery," *Remote Sens.*, vol. 7, no. 11, pp. 14680–14707, 2015.

[48] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[49] R. A. Minhas, A. Javed, A. Irtaza, M. T. Mahmood, and Y. B. Joo, "Shot classification of field sports videos using alexnet convolutional neural network," *Appl. Sci.*, vol. 9, no. 3, p. 483, 2019.

[50] M. He, H. Luo, B. Hui, and Z. Chang, "Pedestrian flow tracking and statistics of monocular camera based on convolutional neural network and Kalman filter," *Appl. Sci.*, vol. 9, no. 8, p. 1624, 2019.

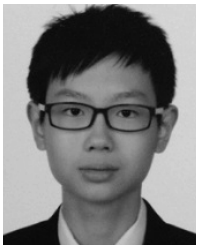
[51] V. Abrishami, A. Rezaee, H. Baherzadeh, and H. Abrishami, "Real-time pedestrian detecting and tracking in crowded and complicated scenario," in *Proc. 3rd Int. Conf. Imag. Crime Detection Prevention (ICDP)*, Dec. 2009, pp. 1–6.

[52] A. Dore, M. Soto, and C. S. Regazzoni, "Bayesian tracking for video analytics," *IEEE Signal Process. Mag.*, vol. 27, no. 5, pp. 46–55, Sep. 2010.
[53] M. Bilal, A. Khan, M. U. K. Khan, and C.-M. Kyung, "A low-complexity pedestrian detection framework for smart video surveillance systems," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 10, pp. 2260–2273, Oct. 2017.
[54] R. Lucas, A. Rowlands, A. Brown, S. Keyworth, and P. Bunting, "Rule-based classification of multi-temporal satellite imagery for habitat and agricultural land cover mapping," *ISPRS J. Photogramm. Remote Sens.*, vol. 62, no. 3, pp. 165–185, Aug. 2007.
[55] Z. Miao, S. Zou, Y. Li, X. Zhang, J. Wang, and M. He, "Intelligent video surveillance system based on moving object detection and tracking," in *Proc. Int. Conf. Inf. Eng. Commun. Technol.*, 2016, pp. 1–4.
[56] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, "Real-time computer vision with OpenCV," *Commun. ACM*, vol. 55, no. 6, pp. 61–69, Jun. 2012.
[57] I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifrek, "A brief introduction to OpenCV," in *Proc. 35th Int. Conv. MIPRO*, 2012, pp. 1725–1730.
[58] S. H. Shaikh, K. Saeed, and N. Chaki, *Moving Object Detection Using Background Subtraction*. Cham, Switzerland: Springer, 2014, pp. 15–23.
[59] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Video-Based Surveillance Systems*. Boston, MA, USA: Springer, 2002, pp. 135–144.
[60] E. Grevelink, "A closer look at object detection, recognition and tracking," Intel, Santa Clara, CA, USA, Tech. Rep., 2017. [Online]. Available: https://software.intel.com/en-us/articles/a-closer-look-at-object-detection-recognition-and-tracking
[61] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proc. ICPR*, 2004, pp. 28–31.
[62] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognit. Lett.*, vol. 27, no. 7, pp. 773–780, 2006.
[63] A. B. Godbehere, A. Matsukawa, and K. Goldberg, "Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation," in *Proc. Amer. Control Conf. (ACC)*, 2012, pp. 4305–4312.
[64] T. Trnovszký, P. Sýkora, and R. Hudec, "Comparison of background subtraction methods on near infra-red spectrum video sequences," *Procedia Eng.*, vol. 192, pp. 887–892, Jun. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877705817327005, doi: 10.1016/j.proeng.2017.06.153.
[65] I. Benraya and N. Benblidia, "Comparison of background subtraction methods," in *Proc. Int. Conf. Appl. Smart Syst. (ICASS)*, 2018, pp. 1–5.
[66] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 779–788.
[67] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 21–37.
[68] *Chuanqi305 Mobilenet-SSD*. Accessed: Oct. 29, 2019. [Online]. Available: https://github.com/chuanqi305/MobileNet-SSD
[69] *N/A, Common Objects in Context*. Accessed: Oct. 29, 2019. [Online]. Available: http://cocodataset.org/#home
[70] A. Rosebrock. (2018). *Simple Object Tracking With OpenCV*. Accessed: Aug. 8, 2019. [Online]. Available: https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/

KANG HAO CHEONG (M'18) received the B.Sc. degree (Hons.) from the Department of Mathematics and University Scholars Programme, National University of Singapore (NUS), in 2007, the Postgraduate diploma in education from the National Institute of Education, Singapore, and the Ph.D. degree from the Department of Electrical and Computer Engineering, NUS, in 2015. He was an Assistant Professor with the Engineering Cluster, Singapore Institute of Technology, from 2016 to 2018. He is currently an Assistant Professor with the Science and Math Cluster, Singapore University of Technology and Design (SUTD). He is also affiliated with the SUTD-MIT International Design Centre.

SANDRA POESCHMANN received the B.Eng. degree (Hons.) in sustainable infrastructure engineering (building services) from the Singapore Institute of Technology, in 2018. She is currently an Associate Geospatial Specialist with the Government Technology Agency (GovTech), Singapore.

JOEL WEIJIA LAI received the B.Sc. degree (Hons.) in physics with a second major in mathematical sciences from Nanyang Technological University (NTU), Singapore, in 2017. He is currently pursuing the Ph.D. degree with the Singapore University of Technology and Design (SUTD), Singapore. He stayed on NTU to develop technology enhanced learning tools for the Division of Physics and Applied Physics. He went to become a Research Assistant with SUTD.



JIN MING KOH received the NUS High School Diploma (High Distinction), in 2016. Since 2017, he has been undertaking research projects offered by K. H. Cheong. He is currently with the California Institute of Technology.

U. RAJENDRA ACHARYA received the Ph.D. degree from the National Institute of Technology Karnataka, Surathkal, India, and the D.Eng. degree from Chiba University, Japan. He is currently a Senior Faculty Member with Ngee Ann Polytechnic, Singapore. He is also an Adjunct Professor with Taylor's University, Malaysia, an Adjunct Faculty with the Singapore Institute of Technology–University of Glasgow, Singapore, and an Associate Faculty with the Singapore University of Social Sciences, Singapore. He has published more than 400 articles in refereed international SCI-IF journals (345), international conference proceedings (42), books (17) with more than 20 000 citations in Google Scholar (with H-index of 75), and Research Gate (RG) score of 47.1. He holds three patents. His major academic interests include biomedical signal processing, biomedical imaging, data mining, visualization, and biophysics for better healthcare design, delivery, and therapy. He is ranked in the top 1% of the Highly Cited Researchers for the last four consecutive years (2016–2019) in computer science according to the Essential Science Indicators of Thomson.

SIMON CHING MAN YU received the B.Eng. degree (Hons.) in ACGI and the Ph.D. degree in DIC from the Department of Mechanical Engineering, Imperial College, London, in 1987 and 1991, respectively. He is currently a Professor with the Interdisciplinary Division of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University. He joined Nanyang Technological University, Singapore, immediately after his graduation, as a Lecturer. He became a Senior Lecturer, in 1996, and an Associate Professor, in 2000. He has been involved in the University Administration, since 2000: a Principal Staff Officer (President's Office), from 2000 to 2003, a University Council Member, from 2001 to 2003, the Vice Dean, Admission Office, from 2003 to 2006, and the Head of the Division of Aerospace Engineering, School of Mechanical and Aerospace Engineering, from 2008 to 2013. He moved over to the Singapore Institute of Technology as a Professor and a Programme Director, in 2013, to establish one of the first engineering degree programmes offered solely by the University. He has published more than 200 research articles in archive journals and conferences. He managed to secure more than SGD 50 million external grant during his tenure in the Nanyang Technological University, especially during the period as the Head of the Division of Aerospace Engineering.

KENNETH JIAN WEI TANG received the B.Eng. degree (Hons.) in sustainable infrastructure engineering (building services) and the M.Eng.Tech. degree from the Singapore Institute of Technology (SIT), in 2018 and 2019, respectively. He is currently pursuing the Ph.D. degree with the Science and Math Cluster, Singapore University of Technology and Design.