People Flow Analysis using Jetson Nano Developer Kit

A PROJECT REPORT

Submitted by

NISHOK KATHIRAVEN [RA1911003040012]

Under the guidance of

DEEPA R

(Assistant Professor, Department of Computing Technologies)

in partial fulfilment for the award of

the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

Department of Computer Science & Engineering
Vadapalani Campus,Chennai

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that 18CSP109L project report titled "People Flow Analysis using Jetson Nano Developer Kit" is the bonafide work of **NISHOK KATHIRAVEN [RA1911003040012] , UMEIR AHMED [RA1911003040136] and MIZBAH SYED [RA1911003040168]** , who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

GUIDE
Deepa R
Assistant Professor
Dept.of Computer Science & Engg
SRMIST,Vadapalani Campus

HEAD OF THE DEPARTMENT
Dr.S.Prasanna Devi, B.E.,M.E.,
Ph.D., PGDHRM.,PDF(IISc)
Professor
Dept.of Computer Science & Engg
SRMIST,Vadapalani Campus

INTERNAL EXAMINER

EXTERNAL EXAMINER

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Own Work Declaration Form

This sheet must be filled in and signed with date along with the student registration number, work will not be marked unless this is done.

To be completed by the student:

Degree/ Course :

Student Name :

Registration Number :

Title of Work :

I / We hereby certify that this work compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this project is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

Signature of the Student(s)

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr.C.Muthamizhchelvan**, Vice Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support. We extend our sincere thanks to **Our Dean Dr.C.V.Jayakumar**, SRM Institute of Science and Technology, Vadapalani campus for his invaluable support. We wish to thank **Dr.S.Prasanna Devi**, Professor & Head, Department of CSE, SRM Institute of Science and Technology, Vadapalani Campus for her valuable suggestions and encouragement throughout the period of the project work.

Our inexpressible respect and thanks to my guide Mrs. Deepa R, Associate Professor, Department of **CSE**, SRM Institute of Science and Technology, Vadapalani Campus for providing me an opportunity to pursue my project under his/her mentorship. He / She provided me the freedom and support to explore the research topics of my interest.

We sincerely thank our Management, staff, and students of the Department of **CSE**, SRM Institute of Science and Technology, Vadapalani Campus who have directly or indirectly helped our project. Finally, we would like to thank my parents, our family members, and our friends for their unconditional love, constant support, and encouragement.

Nishok Kathivaren

Umeir Ahmed

Mizbah Syed

ABSTRACT

People Flow Analysis is a novel technique that can be very helpful for enhancing safety and short-term planning in the stage of organising and operating big public spaces is online simulation of people movement in public buildings. Shopping malls, train stations, airports, and many more structures and amenities are expanding in size and drawing an increasing number of visitors. These crowds must be efficiently channelled through the structures. The development of design standards for pedestrian pathways, sporting arenas, and evacuation plans has historically been accomplished through pedestrian movement studies. Numerous studies have been done in these emphasis areas to examine the flow rates on walkways, down stairs, and through exit routes.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 JETSON NANO

NVIDIA Jetson Nano Developer Kit is a small, powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. All in an easy-to-use platform that runs in as little as 5 watts. Just insert a microSD card with the system image, boot the developer kit, and begin using the same NVIDIA Jetpack SDK used across the entire NVIDIA Jetson™ family of products. Jetpack is compatible with NVIDIA's world-leading AI platform for training and deploying AI software, reducing complexity and effort for developers. The project will make use of TensorFlow 2.01, Keras 2.1, and OpenCV 4.1. A pre-requisite is to install Cuda10.0 and Visual Studio Express 17.0 to leverage the GPU speed gains in case the laptop comes with an NVIDIA-enabled GPU.

1.1 PEOPLE FLOW ANALYSIS

People flow analysis in surveillance cameras is a key technology for understanding the flow population and generating heat maps. It is a device used to measure the number and direction of people traversing a certain passage or entrance per unit time. In recent years, people detection performance has been greatly improved with the development of object detection algorithms using deep learning. In this project, we focus our attention on one outstanding topic in computer vision: pedestrian analysis. We made this selection based on the diverse use cases that the analysis of people behaviour can provide to local authorities to tailor the needs of the local population where this technology might be positioned. For instance, we envision the following applications:

- Counting people leaving and entering into subway or bus stations to regulate pedestrian transit.
- Counting people looking at articles in the windows of stores to establish a number of potential customers.

CHAPTER 2
LITERATURE SURVEY

This section discusses the prior literature as well as several people flow analysis methods. More research is being done on video surveillance and systems that can identify crowds as a result of the industry-revolutionizing rise in video surveillance usage. Brostow and Cipolla develop a method for spotting particular individuals in crowds. The technology has a flaw, though, when they run into other objects like businesses and kiosks or loud noises [3]. For Pathan et al., they use a system that counts incorrect moves in public spaces. Due to the methodologies utilised, the subtraction process in detecting people, the system has had issues with the accuracy of the output [20]. Krausz and Bauckage, meanwhile, propose the concept of a system that will automatically recognise the crucial circumstance when there is congestion by using an alarm system. When the system is put into place, they run into a problem [12]. Furthermore, because every individual's appearance is unique, Zhao and Nevatia use articulated ellipsoids in their research to further simulate the human shape [33]. putting into practise the enhanced Gaussian distribution for the backdrop model. The MCMC technique is used in conjunction with the moving head pixel detection to increase the likelihood of detecting crowds in busy areas.

CHAPTER 3
PROPOSED FRAMEWORK & METHODOLOGY

The following are the major modules of the project that was implemented in the project to obtain a system for People Flow Analysis. The NVIDIA's Jetson Nano employs the PyCUDA python library to have access to CUDA's parallel computation API. At the same time, it creates a streaming attribute that fetches the snapped images in the Jetson Nano's memory to perform inference operations using deep learning trained models.

3.1 Design Considerations and Specifications
3.2 Camera Calibration for Detection
3.3 Object Detection and Tracking (Using Centroid Tracker)

3.1 Design Considerations and Specifications

When developing this programme, a variety of factors were taken into consideration.

- The use of snapshots rather than videos to enable quick analysis of a variety of objects and to evaluate the system's robustness.
- The software's capacity to use other, equally human-defining qualities outside facial detection.
- Modular, straightforward design and explicit comments on the programme to make it simple to add new features and make enhancements.
- To speed up processing, convert the photos to grayscale before detection.

This method performs multi-scale object detection on the input image. The detector can detect a variety of objects, including faces and a person's upper body. The default is configured to detect faces. in this work five (5) 'profilers' were used, they are:

1) Face
2) Upper body
3) Nose
4) Mouth
5) Profile Face

3.2 Camera Calibration for Detection

The inherent camera characteristics and the existence of distortion have an impact on the image the camera captures. The focal length and main point are the intrinsic camera parameters. It is typically essential to correct the distortion using the intrinsic camera characteristics in order to do picture processing. The frames are read from the video and a desired reference line is drawn on the input frame. The people are detected using the object detection model. The centroid is marked on the detected person and the movement of the marked centroid is tracked for further analysis such as counting, calculating direction of movement. Based on the counting, the number of people are incremented up or down on the counter.

3.3 Object Detection and Tracking (Using Centroid Tracker)

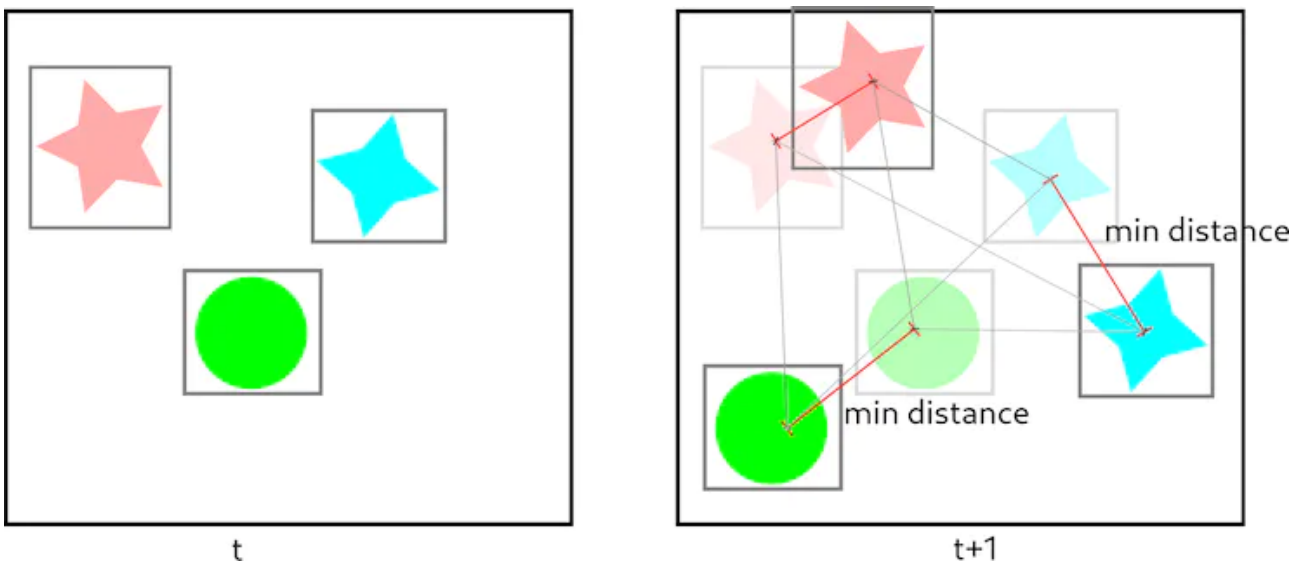
The object tracking technique used is Centroid Tracker. The counting operation and object IN-OUT operation can be applied to the detected item once we have a firm grasp on it throughout the frames.The centroid tracker has the following steps:

- Accepts the bounding box coordinates and computes the centroid.
- The algorithm accepts the bounding box coordinates that are xmin, ymin, xmax, and ymax and the gives (x_center, y_center) coordinates for each of the detected objects in each frame.
- The Centroid is calculated is given below:

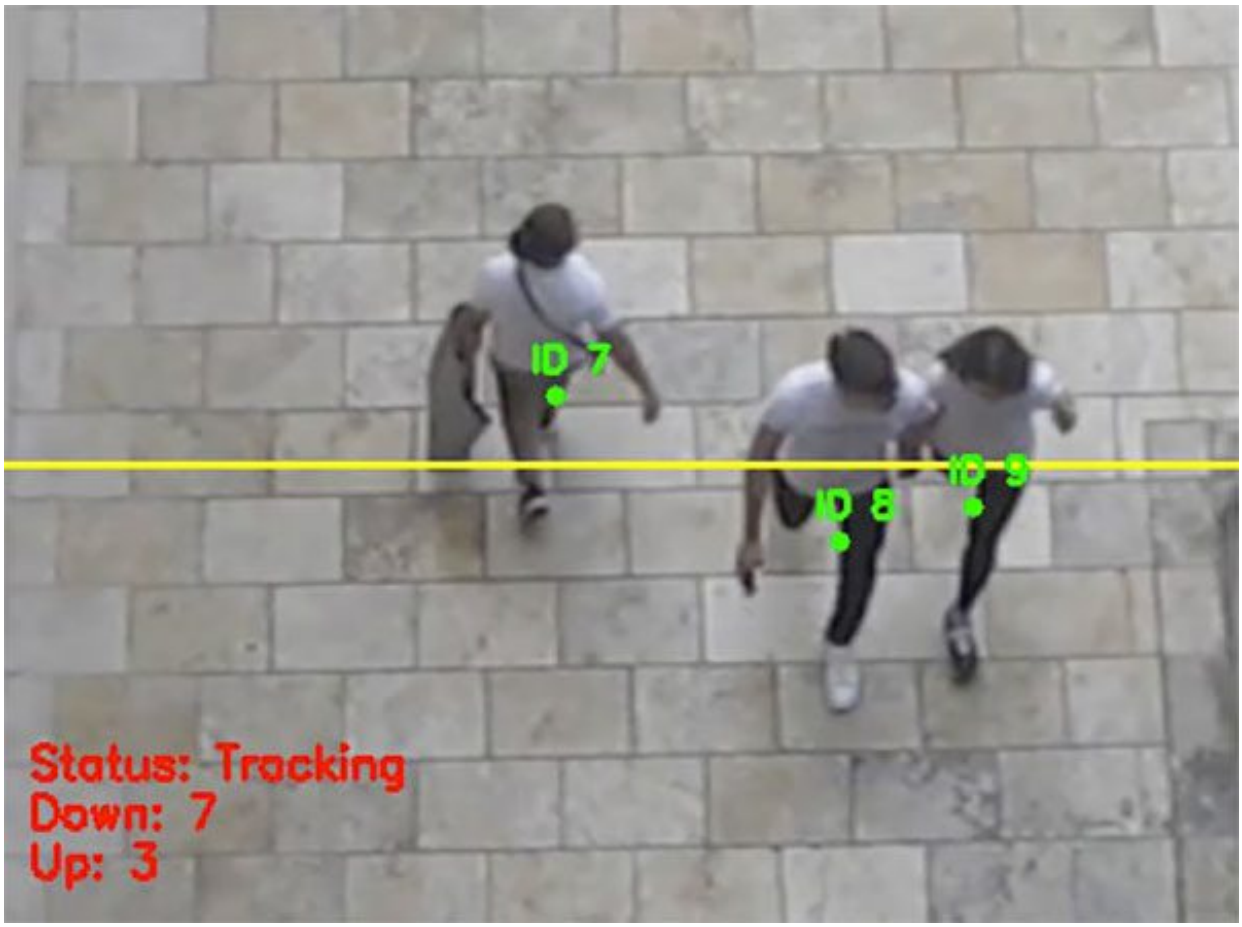
$$X\_cen = ((xmin + xmax) // 2$$
$$Y\_cen = ((ymin + ymax) // 2)$$

where xmin, ymin, xmax, and ymax are the bounding box coordinates for object detection model

- Then, it calculates the euclidian distance between the new detected bounding box and the existing object.
- Update the centroid for the existing object. After calculating the euclidian distance between the detected bounding box and the existing bounding box, it will update the position of the centroid in the frame, there by tracking the object.
- Registering new objects. When a new object enters or the same object is been detected, the centroid tracker will register the new object with a unique ID, so that it becomes helpful for different applications.
- De-registering the previous objects. Once the object is not in the frame, the algorithm will de-register the object ID, stating that the object is not available or left the frame.



Schema of the object tracking algorithm (Centroid Tracker)



Sample snippet of a video, tracking the entry and exit of people

CHAPTER 4
IMPLEMENTATION

NVIDIA'S Jetson Nano comes with a Software Development Kit, known as Jetpack 4.3.1, which is a variation of the GNU/Linux Ubuntu operating system, used for building artificial intelligence applications.

The application performs the following tasks:

- Capturing the frames from the camera source.
- Loading an object detection algorithm.
- Running the inference on each frame.
- Filtering images.
- Displays results.

CODE:

```
from mylib.centroidtracker import CentroidTracker
from mylib.trackableobject import TrackableObject
from imutils.video import VideoStream
from imutils.video import FPS
from mylib.mailer import Mailer
from mylib import config, thread
import time, schedule, csv
import numpy as np
import argparse, imutils
import time, dlib, cv2, datetime
from itertools import zip_longest

t0 = time.time()

def run():

    # construct the argument parse and parse the arguments
    ap = argparse.ArgumentParser()
    ap.add_argument("-p", "--prototxt", required=False,
        help="path to Caffe 'deploy' prototxt file")
```

```
ap.add_argument("-m", "--model", required=True,
    help="path to Caffe pre-trained model")
ap.add_argument("-i", "--input", type=str,
    help="path to optional input video file")
ap.add_argument("-o", "--output", type=str,
    help="path to optional output video file")
# confidence default 0.4
ap.add_argument("-c", "--confidence", type=float, default=0.4,
    help="minimum probability to filter weak detections")
ap.add_argument("-s", "--skip-frames", type=int, default=30,
    help="# of skip frames between detections")
args = vars(ap.parse_args())

# initialize the list of class labels MobileNet SSD was trained to
# detect
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
    "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
    "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
    "sofa", "train", "tvmonitor"]

# load our serialized model from disk
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

# if a video path was not supplied, grab a reference to the ip camera
if not args.get("input", False):
    print("[INFO] Starting the live stream..")
    vs = VideoStream(config.url).start()
    time.sleep(2.0)

# otherwise, grab a reference to the video file
else:
    print("[INFO] Starting the video..")
    vs = cv2.VideoCapture(args["input"])
```

```
# initialize the video writer (we'll instantiate later if need be)
writer = None

# initialize the frame dimensions (we'll set them as soon as we read
# the first frame from the video)
W = None
H = None

# instantiate our centroid tracker, then initialize a list to store
# each of our dlib correlation trackers, followed by a dictionary to
# map each unique object ID to a TrackableObject
ct = CentroidTracker(maxDisappeared=40, maxDistance=50)
trackers = []
trackableObjects = {}

# initialize the total number of frames processed thus far, along
# with the total number of objects that have moved either up or down
totalFrames = 0
totalDown = 0
totalUp = 0
x = []
empty=[]
empty1=[]

# start the frames per second throughput estimator
fps = FPS().start()

if config.Thread:
    vs = thread.ThreadingClass(config.url)

# loop over frames from the video stream
while True:
```

```
# grab the next frame and handle if we are reading from either
# VideoCapture or VideoStream
frame = vs.read()
frame = frame[1] if args.get("input", False) else frame

# if we are viewing a video and we did not grab a frame then we
# have reached the end of the video
if args["input"] is not None and frame is None:
    break

# resize the frame to have a maximum width of 500 pixels (the
# less data we have, the faster we can process it), then convert
# the frame from BGR to RGB for dlib
frame = imutils.resize(frame, width = 500)
rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

# if the frame dimensions are empty, set them
if W is None or H is None:
    (H, W) = frame.shape[:2]

# if we are supposed to be writing a video to disk, initialize
# the writer
if args["output"] is not None and writer is None:
    fourcc = cv2.VideoWriter_fourcc(*"mp4v")
    writer = cv2.VideoWriter(args["output"], fourcc, 30,
        (W, H), True)

# initialize the current status along with our list of bounding
# box rectangles returned by either (1) our object detector or
# (2) the correlation trackers
status = "Waiting"
rects = []
```

```
# check to see if we should run a more computationally expensive
# object detection method to aid our tracker
if totalFrames % args["skip_frames"] == 0:
    # set the status and initialize our new set of object trackers
    status = "Detecting"
    trackers = []

    # convert the frame to a blob and pass the blob through the
    # network and obtain the detections
    blob = cv2.dnn.blobFromImage(frame, 0.007843, (W, H), 127.5)
    net.setInput(blob)
    detections = net.forward()

    # loop over the detections
    for i in np.arange(0, detections.shape[2]):
        # extract the confidence (i.e., probability) associated
        # with the prediction
        confidence = detections[0, 0, i, 2]

        # filter out weak detections by requiring a minimum
        # confidence
        if confidence > args["confidence"]:
            # extract the index of the class label from the
            # detections list
            idx = int(detections[0, 0, i, 1])

            # if the class label is not a person, ignore it
            if CLASSES[idx] != "person":
                continue

            # compute the (x, y)-coordinates of the bounding box
            # for the object
            box = detections[0, 0, i, 3:7] * np.array([W, H, W, H])
```

```
            (startX, startY, endX, endY) = box.astype("int")



            # construct a dlib rectangle object from the bounding
            # box coordinates and then start the dlib correlation
            # tracker
            tracker = dlib.correlation_tracker()
            rect = dlib.rectangle(startX, startY, endX, endY)
            tracker.start_track(rgb, rect)

            # add the tracker to our list of trackers so we can
            # utilize it during skip frames
            trackers.append(tracker)

    # otherwise, we should utilize our object *trackers* rather than
    # object *detectors* to obtain a higher frame processing throughput
    else:
        # loop over the trackers
        for tracker in trackers:
            # set the status of our system to be 'tracking' rather
            # than 'waiting' or 'detecting'
            status = "Tracking"

            # update the tracker and grab the updated position
            tracker.update(rgb)
            pos = tracker.get_position()

            # unpack the position object
            startX = int(pos.left())
            startY = int(pos.top())
            endX = int(pos.right())
            endY = int(pos.bottom())
```

```
            # add the bounding box coordinates to the rectangles list
            rects.append((startX, startY, endX, endY))

    # draw a horizontal line in the center of the frame -- once an
    # object crosses this line we will determine whether they were
    # moving 'up' or 'down'
    cv2.line(frame, (0, H // 2), (W, H // 2), (0, 0, 0), 3)
    cv2.putText(frame, "-Prediction border - Entrance-", (10, H - ((i * 20) + 200)),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1)

    # use the centroid tracker to associate the (1) old object
    # centroids with (2) the newly computed object centroids
    objects = ct.update(rects)

    # loop over the tracked objects
    for (objectID, centroid) in objects.items():
        # check to see if a trackable object exists for the current
        # object ID
        to = trackableObjects.get(objectID, None)

        # if there is no existing trackable object, create one
        if to is None:
            to = TrackableObject(objectID, centroid)

        # otherwise, there is a trackable object so we can utilize it
        # to determine direction
        else:
            # the difference between the y-coordinate of the *current*
            # centroid and the mean of *previous* centroids will tell
            # us in which direction the object is moving (negative for
            # 'up' and positive for 'down')
            y = [c[1] for c in to.centroids]
            direction = centroid[1] - np.mean(y)
```

```
to.centroids.append(centroid)

# check to see if the object has been counted or not
if not to.counted:
    # if the direction is negative (indicating the object
    # is moving up) AND the centroid is above the center
    # line, count the object
    if direction < 0 and centroid[1] < H // 2:
        totalUp += 1
        empty.append(totalUp)
        to.counted = True

    # if the direction is positive (indicating the object
    # is moving down) AND the centroid is below the
    # center line, count the object
    elif direction > 0 and centroid[1] > H // 2:
        totalDown += 1
        empty1.append(totalDown)
        #print(empty1[-1])
        # if the people limit exceeds over threshold, send an email alert
        if sum(x) >= config.Threshold:
            cv2.putText(frame, "-ALERT: People limit exceeded-", (10, frame.shape[0] - 80),
                cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 255), 2)
            if config.ALERT:
                print("[INFO] Sending email alert..")
                Mailer().send(config.MAIL)
                print("[INFO] Alert sent")

        to.counted = True

    x = []
    # compute the sum of total people inside
    x.append(len(empty1)-len(empty))
```

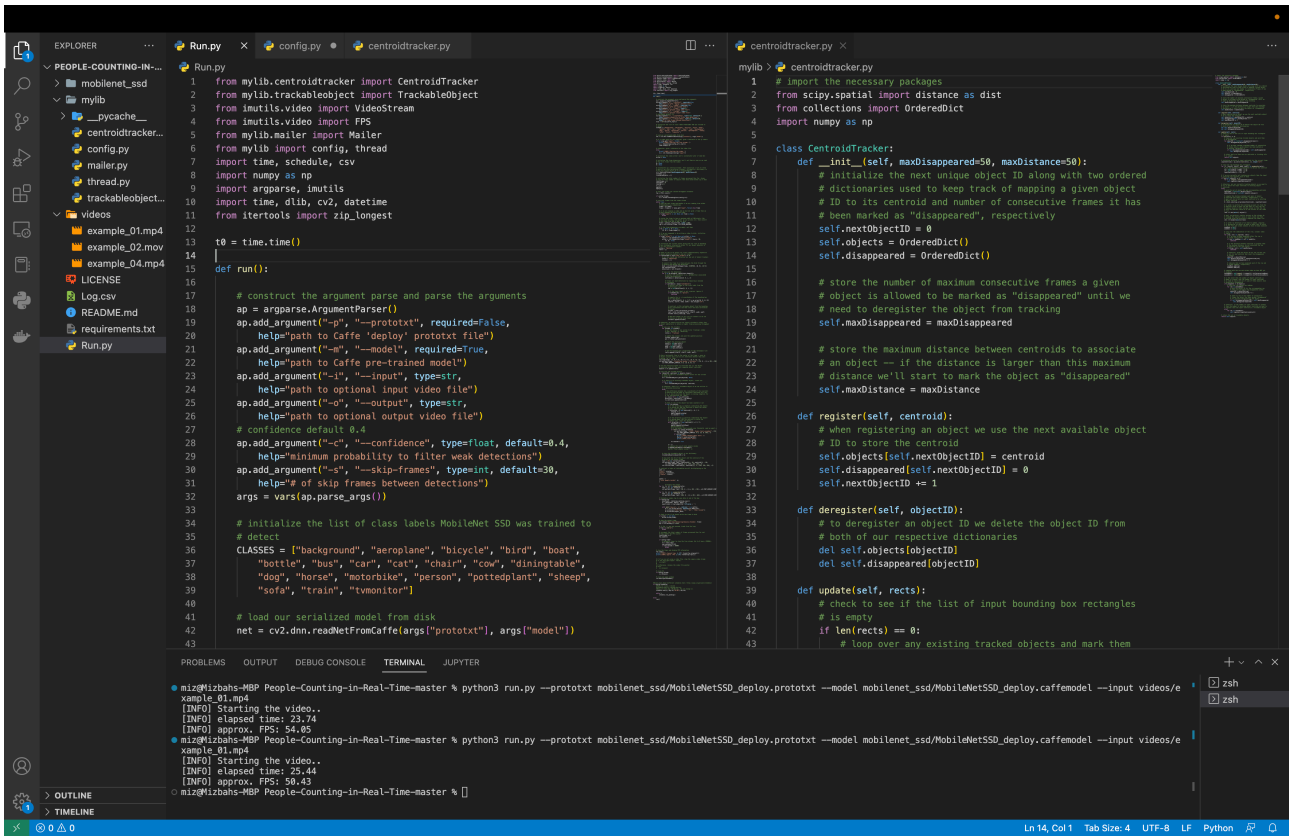
```
            #print("Total people inside:", x)


        # store the trackable object in our dictionary
        trackableObjects[objectID] = to

        # draw both the ID of the object and the centroid of the
        # object on the output frame
        text = "ID {}".format(objectID)
        cv2.putText(frame, text, (centroid[0] - 10, centroid[1] - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)
        cv2.circle(frame, (centroid[0], centroid[1]), 4, (255, 255, 255), -1)

    # construct a tuple of information we will be displaying on the
    info = [
    ("Exit", totalUp),
    ("Enter", totalDown),
    ("Status", status),
    ]

    info2 = [
    ("Total people inside", x),
    ]


        # Display the output
    for (i, (k, v)) in enumerate(info):
        text = "{}: {}".format(k, v)
        cv2.putText(frame, text, (10, H - ((i * 20) + 20)), cv2.FONT_HERSHEY_SIMPLEX, 0.6,
(0, 0, 0), 2)

    for (i, (k, v)) in enumerate(info2):
        text = "{}: {}".format(k, v)
```

```
        cv2.putText(frame, text, (265, H - ((i * 20) + 60)), cv2.FONT_HERSHEY_SIMPLEX, 0.6,
(255, 255, 255), 2)
    # Initiate a simple log to save data at end of the day
    if config.Log:
        datetimee = [datetime.datetime.now()]
        d = [datetimee, empty1, empty, x]
        export_data = zip_longest(*d, fillvalue = '')

        with open('Log.csv', 'w', newline='') as myfile:
            wr = csv.writer(myfile, quoting=csv.QUOTE_ALL)
            wr.writerow(("End Time", "In", "Out", "Total Inside"))
            wr.writerows(export_data)
    # check to see if we should write the frame to disk
    if writer is not None:
        writer.write(frame)
    # show the output frame
    cv2.imshow("Real-Time Monitoring/Analysis Window", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

    # increment the total number of frames processed thus far and
    # then update the FPS counter
    totalFrames += 1
    fps.update()

    if config.Timer:
        # Automatic timer to stop the live stream. Set to 8 hours (28800s).
        t1 = time.time()
        num_seconds=(t1-t0)
        if num_seconds > 28800:
```

```
            break

    # stop the timer and display FPS information
    fps.stop()
    print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
    print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

    # # if we are not using a video file, stop the camera video stream
    # if not args.get("input", False):
    #   vs.stop()
    #
    # # otherwise, release the video file pointer
    # else:
    #   vs.release()

    # issue 15
    if config.Thread:
        vs.release()

    # close any open windows
    cv2.destroyAllWindows()
##learn more about different schedules here: https://pypi.org/project/schedule/
if config.Scheduler:
    ##Runs for every 1 second
    #schedule.every(1).seconds.do(run)
    ##Runs at every day (09:00 am). You can change it.
    schedule.every().day.at("09:00").do(run)

    while 1:
        schedule.run_pending()

else:
    run()
```

CHAPTER 5
RESULT AND DISCUSSION

The object detection algorithm built on top of the Tensor RT framework can be executed smoothly for extended periods of time thanks to the Maxwell GPU integrated on the NVIDIA's Jetson Nano because it offers a good trade-off between performance and power consumption. This is because it has an attached external fan and passive cooling. Unexpectedly, we improved speed by using the SSD Mobilenet model's inference phase. Object tracking is executed by the camera to detect and count people for entry or exit and can also be done by using a sample video as input to detect and analyse the number of people entering, exiting and also the number of people that are staying inside.

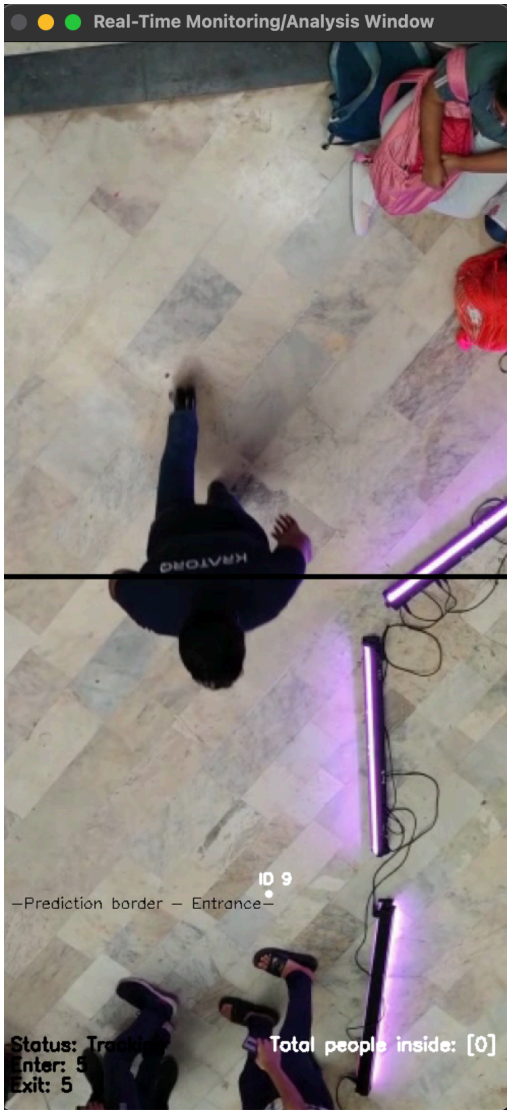


SAMPLE VIDEO

TESTING

TEST 1: Entry 1, Exit 1

TEST 2: Entry 5, Exit 5



CHAPTER 6
CONCLUSION AND FUTURE WORK

In conclusion, by our implementation of the detection and tracking deep learning algorithm a people flow analysis system is successfully created and the required results were obtained. The project is tested and it runs without being overwhelmed by the large number of objects, in this case people present.

Finally, we would like to emphasise that this is just one example of the various applications that deep learning algorithms, when used on a reliable platform like NVIDIA's Jetson Nano, may offer to address common issues in society. By introducing new detection techniques, upgrading the capturing tools used in this work, or simply creating new vulnerability conditions during the transfer of data to a centralised place in the cloud, several expansions to this work can be further used.

CHAPTER 7
REFERENCES

[1] Andy S. 2018. Algorithm Spotlight: Crowd Counter https://algorithmia.com/blog/algorithm-spotlight-crowdcounter .

[2] Arteta C, Lempitsky V, Noble J A and Zisserman A 2014 Interactive object counting Springer Cham pp 504-518.

[3] Brostow G and Cipolla R. 2006. Unsupervised Bayesian Detection of Independent Motion in Crowds University of Cambridge Vol 1.

[4] Chan A B, Liang Z S J and Vasconcelos N 2008 Privacy preserving crowd monitoring: Counting people without people models or tracking San Diego pp. 1-7.

[5] Change L C, Gong S, and Xiang T 2013. From semi-supervised to transfer counting of crowds. London, United Kingdom pp 2256-226.

[6] Dailey K 2013. The rise of CCTV surveillance in the US https://www.bbc.com/news/magazine-22274770.

[7] Faisal E, Sleit A, and Alsayyed R.2018. Crowd Counting Mapping to Make a Decision University of Jordan Amman, Jordan Vol 9.

[8] Gao G, Gao J and Liu Q 2020. CNN-based Density Estimation and Crowd Counting: A Survey.

[9] Goodfellow I, Bengio, Y and Courville A .2016. Deep learning MIT Press.

[10] Idrees H, Saleemi I, Seibert C and Shah M. 2013. Multi-source multi-scale counting in extremely dense crowd images University of Central Florida pp 2547-2554.

[11] Kong D, Gray D and Tao H. 2006. A viewpoint invariant approach for crowd counting Santa Cruz, California vol 3 pp 1187-1190.

[12] Krausz B and Bauckhage C. 2011. Automatic detection of dangerous motion behavior in human crowds, in Advanced Video and Signal-Based Surveillance (AVSS) pp 224-229.

[13] Kurama V .2019. Dense and Sparse Crowd Counting Methods and Techniques: A Review https://nanonets.com/blog/crowd-counting-review/#what-is-crowd-counting.

[14] Lee W Y, Ko K E, Geem Z W and Sim K B 2017 Method that determining the Hyperparameter of CNN using HS algorithm vol 27 pp 22-28.

[15] Lempitsky V and Zisserman A 2010. Learning to count objects in images. In Advances in neural information processing systems pp 1324-1332.

[16] Lin S F and Lin C D 2006. Estimation of the pedestrians on a crosswalk. Savitribai Phule Pune University Pune, India pp 4931–4936.

[17] Loy C, Chen K, Gong S and Xiang T. 2013. Crowd Counting and Profiling: Methodology and Evaluation London, United Kingdom pp 347-382.

[18] Luo H, Sang J, Wu W, Xiang H, Xiang Z, Zhang Q, and Wu Z. 2018. Applied Chongqing, Chine Vol 8 pp 1–12.
[19] Nakatsuka M, Iwatani H, and Katto J. 2008. International Conference on Mobile Computing and Ubiquitous Networking Tokyo, Japan pp 1–6.
[20] Pathan S, Al-Hamadi A and Michaelis, B. 2010. Crowd Behavior Detection by Statistical Modeling of Motion Patterns in Soft Computing and Pattern pp 8186.
[21] Rabaud, V., & Belongie, S. 2006. Counting crowded moving objects vol 1 pp 705-711.
[22] Regazzoni C S and Tesei A 1995 Signal Processing vol 53 pp 47–63.
[23] Roqueiro D and Petrushin V A. 2007. Counting people using video cameras vol 22 pp 193–209.
[24] Rukmanda T, Sugeng K and Murfi H .2018. Modification of architecture learning convolutional neural network for graph.
[25] Saha S 2018. A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way3bd2b1164a53.
[26] Topkaya I, Erdogan H and Porikli F 2014. Counting people by clustering person detector outputs pp 313- 318.
[27] Wang M and Wang X 2011. Automatic adaptation of a generic pedestrian detector to a specific traffic scene. pp 3401-3408.
[28] Wu J, Li Z, Qu W and Zhou Y. 2019. One Shot Crowd Counting with Deep Scale Adaptive Neural Network Vol 8 p 701.
[29] Xu M, Papageorgiou D, Abidi S, Dao M, Zhao H and Karniadakis G. 2017. A deep convolutional neural network for classification of red blood cells in sickle cell anemia Qing Nie California, United States Vol 13.
[30] Zhang C, Li H, Wang X and Yang X 2015. Cross-scene crowd counting via deep convolutional neural networks Shanghai China pp 833–841.
[31] Zhang D, Peng H, Haibin Y, and Lu Y 2013. Crowd Abnormal Behavior Detection Based on Machine Learning vol 12 pp 1199–1205.
[32] Zhao T and Nevatia R 2004. Tracking Multiple Humans in Complex Situations Vol 26 pp 1208–1221.
[33] Zou Z, Cheng Y, Qu X, Ji S, Guo X and Zhuo P .2019. Attend To Count: Crowd Counting with Adaptive Capacity Multi-Scale CNNs pp 1–10.