MOTION TRACKING

Under the guidance of-
Prof. Swathi J N

Team members:
Aryan Patel(18BCE0897)
Yashasvi D Rattan(18BCE2152)
Kunwar Abhishek Singh(18BCE2176)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING(SCOPE)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

ABSTRACT: -

An adjustment in the estimation of speed or vector of an object or objects in the field of view is called motion. Identification of motion can be accomplished by electronic gadgets or mechanical gadgets that interact or measure the adjustments in the given condition. Four motion detectors are being compared to find the most suitable one to be used in an application.

INTRODUCTION: -

Object Tracking -

The objective of object tracking is to keep watch on something. Regularly based upon or in a collaboration with object detection and recognition, tracking algorithms are intended to find (and keep a relentless watch on) a moving object (or many moving items) after some time in a video stream.

There's a location history of the object (following dependably handles outlines in relationship to each other) which enables us to know how its position has changed after some time. Also, that implies we have a model of the object's movement. A Kalman channel, an arrangement of scientific conditions, can be utilized to decide the future area of a object. By utilizing a progression of estimations set aside a few minutes, this algorithm gives a way to evaluating past, present and future states.

Object tracking is the process of:

- Taking an initial set of object detections (such as an input set of bounding box coordinates)
- Creating a unique ID for each of the initial detections
- And then tracking each of the objects as they move around frames in a video, maintaining the

assignment of unique IDs

Furthermore, object tracking allows us to **apply a unique ID to each tracked object**, making it possible for us to count unique objects in a video.

An ideal object tracking algorithm will:

- Only require the object detection phase once (i.e., when the object is initially detected)
- Will be extremely fast — *much* faster than running the actual object detector itself
- Be able to handle when the tracked object "disappears" or moves outside the boundaries of the video frame
- Be robust to occlusion
- Be able to pick up objects it has "lost" in between frames We will implement **centroid tracking with OpenCV**, an easy to understand, yet highly effective tracking algorithm. Simple face detection and object tracking with OpenCV

LITERATURE SURVEYS: -

1. Attentiveness Measure in Classroom Environment using Face Detection

Source:
2021 6th International Conference on Inventive Computation Technologies (ICICT) Inventive Computation Technologies (ICICT), 2021 6th International Conference on. :1053-1058 Jan, 2021

Abstract:
The concentration of students in the classroom is very important for an effective learning process. If the students are getting deviated, it must be detected then the teacher should be able to take the necessary steps to avoid the situation. A Multitasking Deep Neuro-Fuzzy Model (MDNFM) model is proposed for the accurate prediction of the attentiveness of the students in the classroom. Initially, the images are acquired and transferred to the Capture, Transform and Flow (CTF) tool. Later, these images are preprocessed to make them suitable for face detection (FD) and activity monitoring (AM). This article mainly applies the color models for face detection and proposes a methodology to track the student's attention and produces the output. This system can provide information to the teacher as well as the student.

Authors:
Pandey, Rahul Kumar
Faridi, Ayaz Ahmed
Shrivastava, Gyanesh

2. Optimized Face Detection and Alignment for Low-Cost and Low-Power IoT Systems

Source:
2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS) Internet of Things and Intelligence System (IoTaIS), 2020 IEEE International Conference on. :129- 135 Jan, 2021

Abstract:
Face detection and alignment are challenging operations due to variations in image angles, background lighting conditions and intermediate blocking objects. Recent work has shown that these tasks can be improved through the use of a multi-task cascaded convolutional neural network (MTCNN) architecture. However, it is difficult to implement such an approach in a low-end edge AI system because of its high computational complexity. This paper presents the design of an MTCNN based on a low-cost and low-power processor/FPGA system that can be used in IoT applications. First, we analyze the computational requirements of the algorithm. Based on this analysis, we develop an optimized implementation to achieve real-time processing, taking advantage of the available hardware resources. In order to enhance the throughput and reduce the power consumption for AI edge devices, we store all intermediate results in on-chip block RAM. We achieve a frame rate of 15.2 frames per second, which meets the needs of security cameras that are widely used in IoT systems. Furthermore, our approach has a 2.67 times lower power consumption than for a previous MTCNN implementation.

Authors:
Choi, Kyubaik
Sobelman, Gerald E

3. Improvement of Face Detection Algorithm Based on Lightweight Convolutional Neural Network

Source:
2020 IEEE 6th International Conference on Computer and Communications (ICCC) Computer and Communications (ICCC), 2020 IEEE 6th International Conference on. :1191-1197 Dec, 2020

Abstract:
In recent years, the application of visual processing algorithm based on deep network on mobile terminal has become a popular research problem. In this paper, a new face loss function is designed to solve the problems of limited computing power and storage resource of mobile devices when face

detection is applied to mobile terminal. The face detection algorithm based on SSH (single stage headless face detector) is improved in two aspects: firstly, the feature of sample data is extracted by using the lightweight convolutional neural network based on MobileNet, which can effectively reduce the amount of parameters and calculation of the model; Secondly, the deformable convolution layer is introduced into the detection module of SSH network to improve the modelling ability of face deformation. At the same time, the method of knowledge distillation is introduced. The high-precision large network model is used as the teacher network, and the small network model of the improved SSH algorithm is used as the student network to further improve performance of the detection model.

Authors:
Lingling, Zhu
Fucai, Chen
Chao, Gao

4. Face Detection based on SSD and CamShift

Source:
2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC) Information Technology and Artificial Intelligence Conference (ITAIC), 2020 IEEE 9th Joint International. 9:2324-2328 Dec, 2020

Abstract:
A face detection method combining SSD target detection algorithm and CamShift tracking algorithm was designed for fatigue driving detection. ResNet50 was used to replace the feature extraction network of the original SSD target detection algorithm to improve the accuracy of face location. CamShift and Kalman filter algorithm were used to track the face area to improve the detection speed and reduce the burden of system operation. The real vehicle test shows that the strategy combining SSD network and improved CamShift algorithm significantly improves the detection efficiency, and has a strong robustness to the effects of light change, occlusion loss and skin-like interference.

Authors:
Hu, Xizhi
Huang, Bingyu

5. An approach towards development of automated attendance system using face detection and recognition

Source:
2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON) Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2020 11th IEEE Annual. :0333-0340 Nov, 2020

Abstract:
Conventionally, the process of taking attendance of students in a classroom is quite a laborious task, wherein either the teacher has to call out names of each individual student, or the student has to sign an attendance sheet. In recent times, due to the Covid-19 pandemic, special importance has been laid on facial recognition techniques, which are contact-free (unlike fingerprint scanners), and are in accordance with social distancing norms. In this paper, a software system automating the attendance-taking scheme is presented. This software integrates face detection, image processing and face recognition approaches to come up with a consolidated attendance system capable of overcoming the disadvantages of manual attendance. In the system, an end user has to first log in and subsequently, an IP camera (which is to be installed in the classroom) gets turned on, and the camera starts taking photographs of the classroom. The user can also manually upload images into the system, in case calculation of attendance is not immediately required.

Authors:
Seal, Sayan
Sen, Aishee
Mukerjee, Ritodeep
Das, Asit Kumar

6. **Shen et al. (2013)** proposed a novel hierarchical moving target detection method based on spatiotemporal saliency. Further, they achieved the refined detection results using temporal and spatial saliency information. The experimental results show that this approach identifies moving substances in the airborne video with high accuracy and efficiency. Additionally, this method does not have the effect of time delay when compared with HMI technique. However, this method evaluated object locations in all the video frames as self-sufficiently, false alarms which are inevitable.

7. Guo et al. (2012) suggested object detection approach for tracking the objects in video frames. The simulation result shows this technique was effective and accurateness, robust for generic object classes' detection with good performance. Further needs to focus towards increase classification accurateness in real-time object recognition.

8. **Ben Ayed et al. (2015)** proposed a method for detection of text data based on a texture in video frames by big data analytics. The video frames are decomposed into various fixed size blocks and these blocks are analyzed using haar wavelet transform technique. Further, they used a neural network to classify the text and non-text blocks. However, this study needs to concentrate on extracting the regions towards remove the noisy regions as well as exclude the text like sections.

9. Viswanath et al. (2015) suggested and modeled the approach using non-panoramic background modeling. By the use of this approach, they modeled the entire picture element with one Spatio-temporal Gaussian. The simulations result shows this method able to identify the moving substances with fewer false alarms. However, this method fails once the adequate features are not obtainable from the section.

10. Soundrapandiyan and Mouli (2015) suggested a novel and adaptive method for pedestrian detection. Further, they separated the foreground objects from the background by image pixel intensities. Subsequently, they used high boost filter for enhancing the foreground edges. The efficacy of the proposed method is evident from the subject evaluation results as well as objective evaluation with around 90% of pedestrian's detection rate compared to the other single image existing methods. In future, they planned to improve the performance of the method with higher detection rate and low false positives on par with sequence image methods.

11. Ramya and Rajeswari (2016) suggested a modified frame difference method which uses the correlation between blocks of current image and background image to categorize the pixels as foreground and background. The blocks in the current image which are highly correlated with the background image are considered as background. For the other block, the pixel-wise comparison is made to categorize it as foreground or background. The experiments conducted proved this approach improves the frame difference method particularly as finding accuracy with speed. However, this study needs to focus towards other information available in the blocks such as shape and edge can be used to improve the detection accuracy.

12. Risha and Kumar (2016) suggested an optic flow with the morphological operation for object detection in video. Further applied morphological operation towards obtaining clear moving target image. This study only concentrated on static camera. So need to focus on moving the camera as well as identify multiple objects in video frames.

13. Najva and Bijoy (2016) proposed a model for detection and classification of objects in videos by combining Tensor features with SIFT approach towards classifying the detected objects using Deep Neural Network (DNN). The DNN capable of handling large higher dimensional data with billions of parameters as like human brain. Simulation results obtained illustrate that the proposed classifier model produces more accurate results than the existing methods, which combines both SIFT and tensor features for feature extraction and DNN for classification.

14.Context-Aware Face Detection for Occluded Faces

Source:
2020 6th International Conference on Interactive Digital Media (ICIDM) Interactive Digital Media (ICIDM), 2020 6th International Conference on. :1-4 Dec, 2020

Abstract:
Detecting of occluded faces is a real challenge for face detection in the computer vision area. This challenge tends to be higher when the occlusion is more than half of the face. Regardless to the great success of recent face detection systems, face detection with occlusion is not yet saturated and requires more attention. We proposed in this work an occluded faces dataset in high degree of occlusion along with context-based labelling, which is a useful technique that let CNN to learn more features during training. The experimental benchmark result showed weak performance of current face detection models in our proposed dataset.

Authors:
Alashbi, Abdulaziz Ali Saleh
Sunar, Mohd Shahrizal
Alqahtani, Zieb

15.Comparative Evaluation of Face Detection Algorithms

Source:
2020 16th International Computer Engineering Conference (ICENCO) Computer Engineering Conference (ICENCO), 2020 16th International. :64-71 Dec, 2020

Abstract:
Computer vision is shaping a new era with its constant development of SOTA algorithms. One heavily contested sub-field of computer vision is face detection, due to its versatile usage in many fields such as security, medical diagnosis, entertainment, and military applications. As the technology develops, it aims to run faster and more accurately on mobile devices and remote computers. In this paper, we aim to compare a number

of the best face recognition algorithms and analyze the performance of each of them by deploying each algorithm on a Jetson Nano Developer Kit. Among the 6 algorithms discussed, Haar_FA1 showed the best performance in terms of precision. On the other hand, OpenCV_dnn had the best results in terms of recall. Lastly, in terms of execution time - as it is an important metric- face detectors that belong to the Haar classifier family dominated the comparison.

Authors:
Yamout, Ahmed
Abdelmawgood, Ahmed
Sadick, Ebraam
Naguib, Mohamed

16. Adaptive Nonlinear Tracking Approach for Motion Tracking Applications

Source:
2020 International Conference Nonlinearity, Information and Robotics (NIR) Nonlinearity, Information and Robotics (NIR), 2020 International Conference. :1-5 Dec, 2020

Abstract:
The problem of output regulation for systems affected by nonlinear reference signal, which is caused by exosystem with parametric uncertainties, is addressed. The control law for a nonlinear trajectory is proposed with the design of an adaptive internal model. The efficiency of the proposed algorithm was proved by experiments using the setup that includes an articulated robot. The experimental results are presented.

Authors:
Gromov, Vladislav S.
Meshkov, Aleksei V.
Pyrkin, Anton A.

HARDWARE: -

- Windows-10, i7 processor, 8gb ram
- Webcam/Camera

SOFTWARES USED: -

- PyCharm 2018.2.3
- Additional libraries:
  - OpenCV
  - NumPy
  - Argparse
  - Imutiles
  - time

METHODOLOGY USED: -

The object tracking algorithm used is called centroid tracking as it relies on the Euclidean distance between
(1) existing object centroids (i.e., objects the centroid tracker has already seen before) and
(2) new object centroids between subsequent frames in a video.

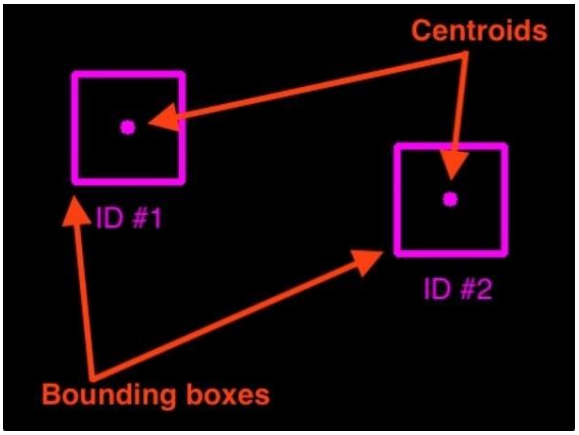
We implanted a Python class to contain this centroid tracking algorithm and then create a Python script to actually run the program and apply it to input videos.

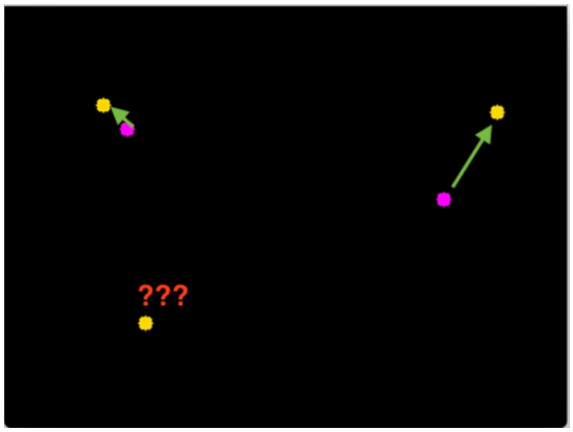
The centroid tracking algorithm accept that we are going in an arrangement of bouncing box (x, y)- coordinates for each distinguished object in each and every casing.

These jumping boxes can be created by a object locator you might want (color thresholding + contour extraction, Haar falls, HOG + Linear SVM, SSDs, Faster R-CNNs, and so forth.), gave that they are figured to each frame in the video. When we have the bouncing box facilitates we should figure the "centroid", or all the more just, the middle (x, y) of the bouncing box.

For each resulting casing in our video stream we apply the previous step of figuring object centroids; be that as it may, rather than relegating another one of a kind ID to each distinguished question (which would nullify the point of object tracking), we first need to decidewhether we can connect the new object centroids with the old object centroids. To achieve this procedure, we process the Euclidean separation between each combine of existing item centroids and input object centroids.

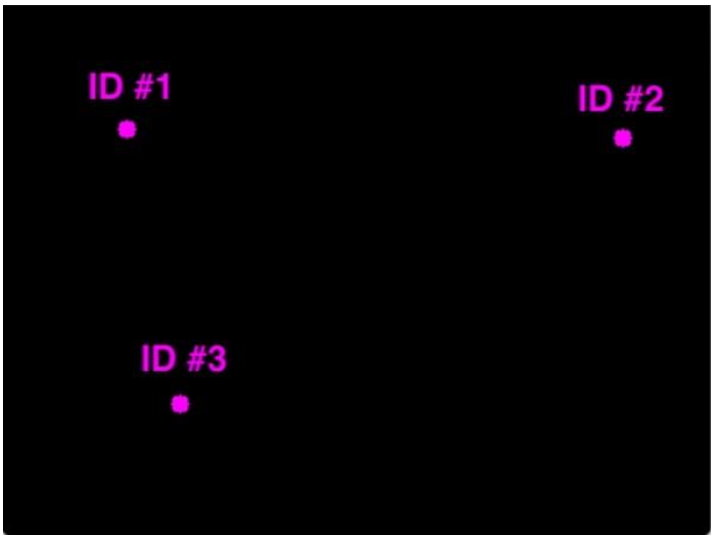
The primary assumption of the centroid tracking algorithm is that a given object will potentially move in between subsequent frames, but the distance between the centroids for frames and will be smaller than all other distances between objects. Hence, if we choose to associate centroids with min distances between the frames which then created our object tracker. F_t F_{t+1}





In the event that there are more input detections than existing objects being tracked, we have to register the new object. “Registering” simply means that we are adding the new object to our list of tracked objects by:

- Assigning it a new object ID
- Storing the centroid of the bounding box coordinates for that object



To use this object tracker specifically to detect the face we have used two files .prototxt and .caffemodel which are part of OpenCV deep learning face detector. However, we can also use another form of detection.

CODE: -

prj.py (initial code)-

```
import cv2
import numpy as np

cap = cv2.VideoCapture('abc.mp4')
frame_width = int( cap.get(cv2.CAP_PROP_FRAME_WIDTH))

frame_height =int( cap.get( cv2.CAP_PROP_FRAME_HEIGHT))

fourcc = cv2.VideoWriter_fourcc('X','V','I','D')

out = cv2.VideoWriter("output.avi", fourcc, 5.0, (1280,720))

ret, frame1 = cap.read()
ret, frame2 = cap.read()

while cap.isOpened():
    diff = cv2.absdiff(frame1, frame2)
    gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5,5), 0)
    _, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY)
    dilated = cv2.dilate(thresh, None, iterations=3)
    contours, _ = cv2.findContours(dilated, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:
        (x, y, w, h) = cv2.boundingRect(contour)

        if cv2.contourArea(contour) < 900:
            continue
        cv2.rectangle(frame1, (x, y), (x+w, y+h), (0, 255, 0), 2)
        cv2.putText(frame1, "Status: {}".format('Movement'), (10,
20), cv2.FONT_HERSHEY_SIMPLEX,
                    1, (0, 0, 255), 3)
    #cv2.drawContours(frame1, contours, -1, (0, 255, 0), 2)

    image = cv2.resize(frame1, (1280,720))
    out.write(image)
    cv2.imshow("feed", frame1)
    frame1 = frame2
    ret, frame2 = cap.read()

    if cv2.waitKey(40) == 27:
        break

cv2.destroyAllWindows()
cap.release()
out.release()
```

tracker.py(final code – part 1) –

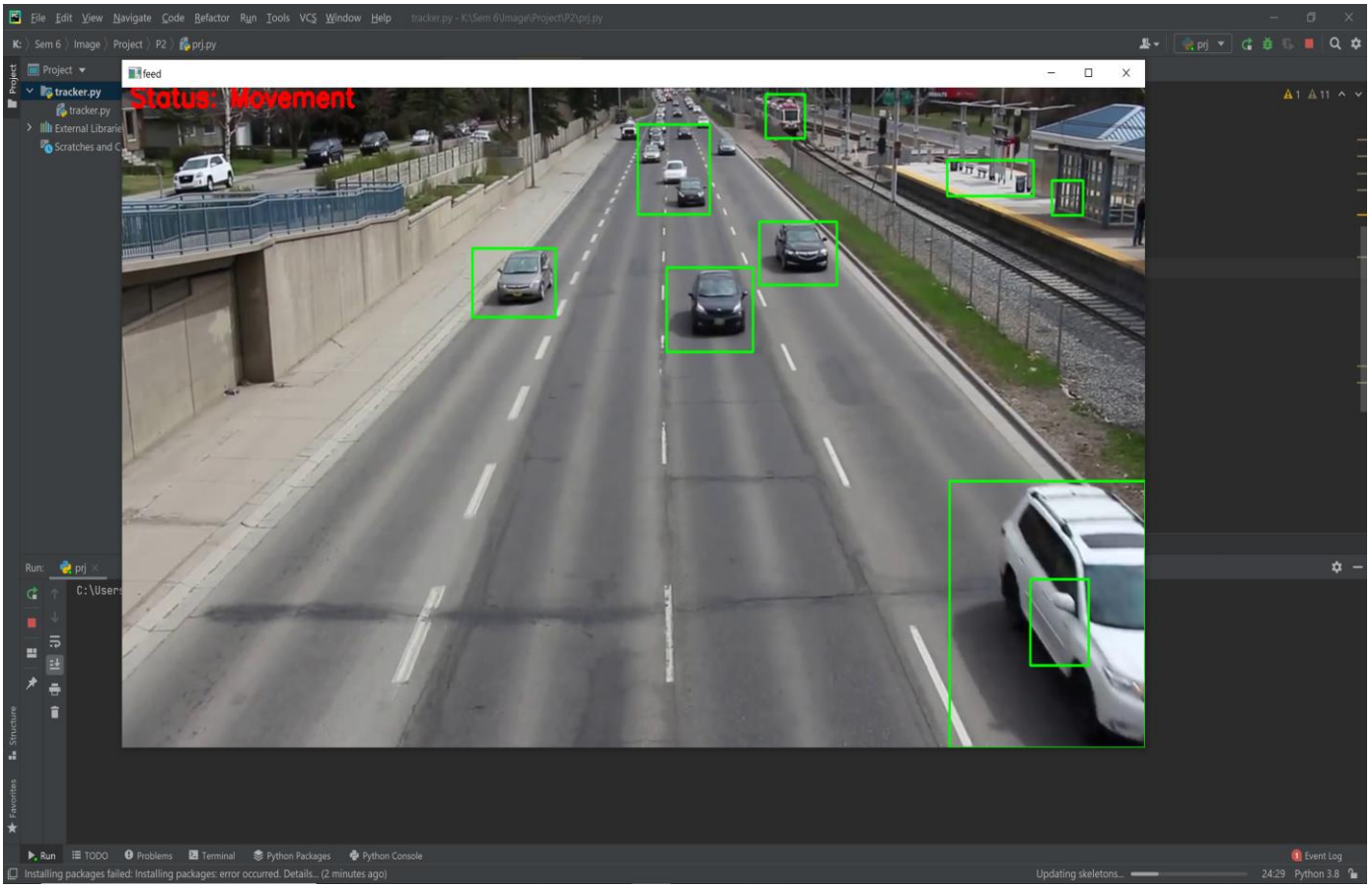
```python
import math


class EuclideanDistTracker:
    def __init__(self):
        # Store the center positions of the objects
        self.center_points = {}
        # Keep the count of the IDs
        # each time a new object id detected, the count will
increase by one
        self.id_count = 0


    def update(self, objects_rect):
        # Objects boxes and ids
        objects_bbs_ids = []

        # Get center point of new object
        for rect in objects_rect:
            x, y, w, h = rect
            cx = (x + x + w) // 2
            cy = (y + y + h) // 2

            # Find out if that object was detected already
            same_object_detected = False
            for id, pt in self.center_points.items():
                dist = math.hypot(cx - pt[0], cy - pt[1])

                if dist < 25:
                    self.center_points[id] = (cx, cy)
                    print(self.center_points)
                    objects_bbs_ids.append([x, y, w, h, id])
                    same_object_detected = True
                    break

            # New object is detected we assign the ID to that object
            if same_object_detected is False:
                self.center_points[self.id_count] = (cx, cy)
                objects_bbs_ids.append([x, y, w, h, self.id_count])
                self.id_count += 1

        # Clean the dictionary by center points to remove IDS not
used anymore
        new_center_points = {}
        for obj_bb_id in objects_bbs_ids:
            _, _, _, _, object_id = obj_bb_id
            center = self.center_points[object_id]
            new_center_points[object_id] = center

        # Update dictionary with IDs not used removed
        self.center_points = new_center_points.copy()
        return objects_bbs_ids
```

main.py (final code – part 2) -

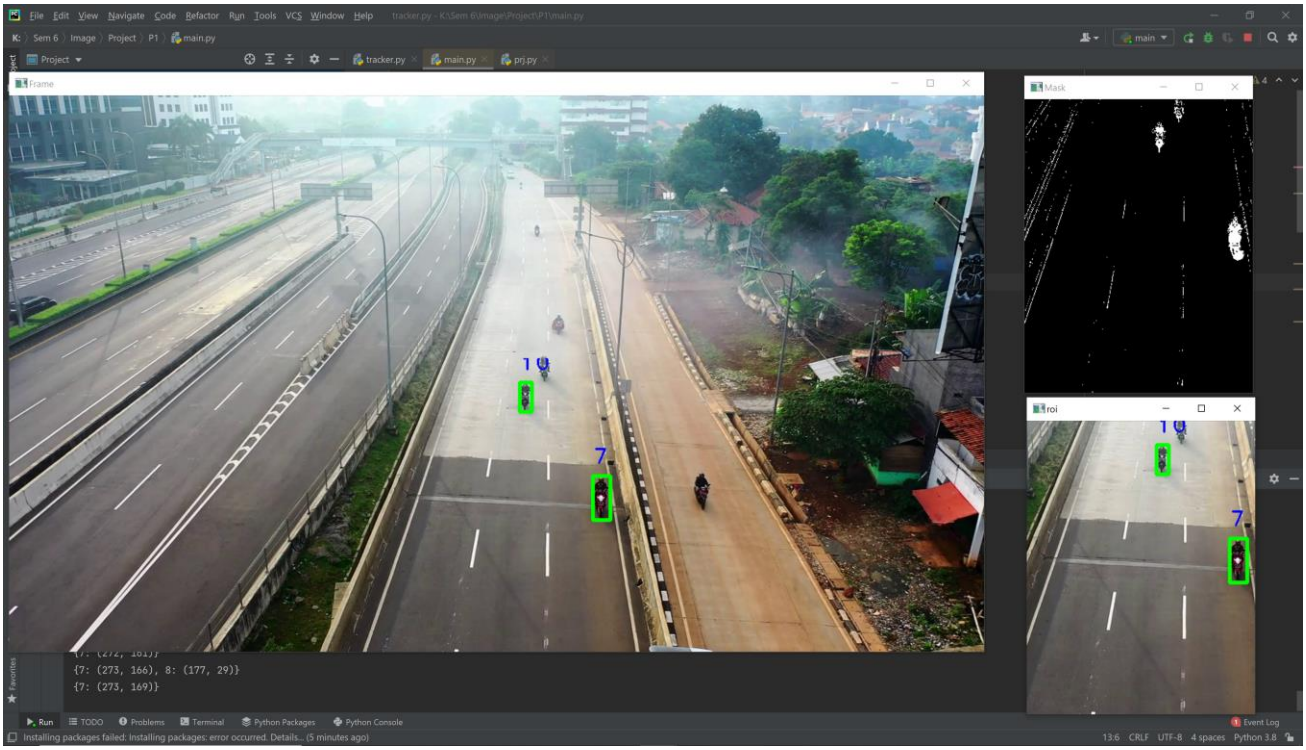
```
import cv2
from tracker import *

# Create tracker object
tracker = EuclideanDistTracker()

cap = cv2.VideoCapture("highway.mp4")

# Object detection from Stable camera
object_detector = cv2.createBackgroundSubtractorMOG2(history=100,
varThreshold=40)

while True:
    ret, frame = cap.read()
    height, width, _ = frame.shape

    # Extract Region of interest
    roi = frame[340: 720,500: 800]

    # 1. Object Detection
    mask = object_detector.apply(roi)
    _, mask = cv2.threshold(mask, 254, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    detections = []
    for cnt in contours:
        # Calculate area and remove small elements
        area = cv2.contourArea(cnt)
        if area > 100:
            #cv2.drawContours(roi, [cnt], -1, (0, 255, 0), 2)
            x, y, w, h = cv2.boundingRect(cnt)


            detections.append([x, y, w, h])

    # 2. Object Tracking
    boxes_ids = tracker.update(detections)
    for box_id in boxes_ids:
        x, y, w, h, id = box_id
        cv2.putText(roi, str(id), (x, y - 15),
cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 2)
        cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3)

    cv2.imshow("roi", roi)
    cv2.imshow("Frame", frame)
    cv2.imshow("Mask", mask)

    key = cv2.waitKey(60)
    if key == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

OUTPUT: -

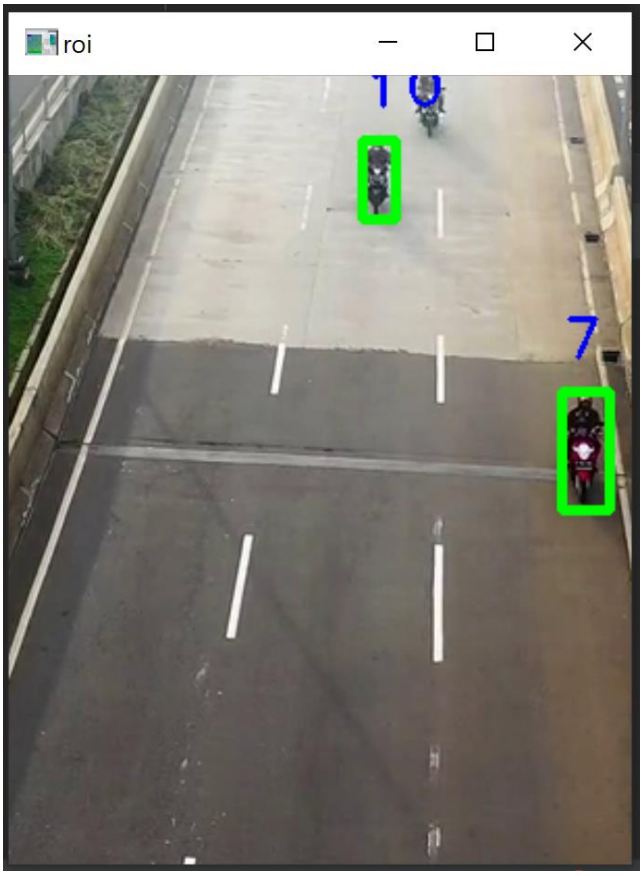
Initial Code:



Final Code:

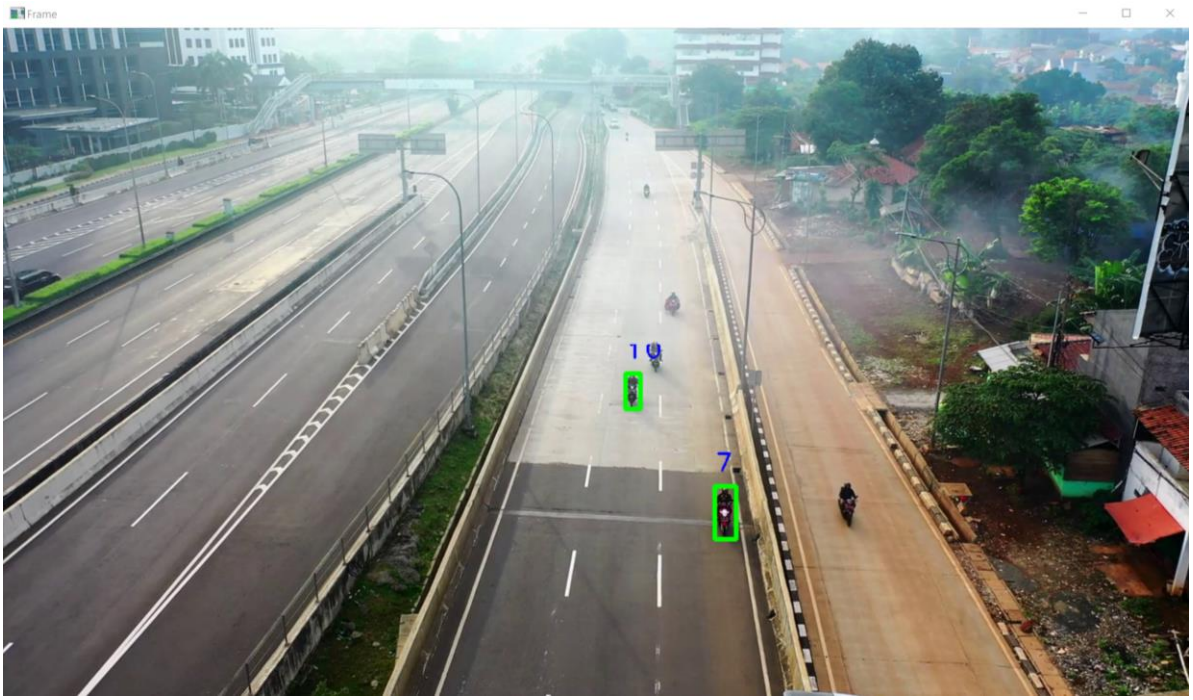


Roi-



Mask-

Frame with count-



The object tracker can be used to detect more than one face. However, if an object is removed from the frame the tracker will remain till the object has existed outside the field of view of tracker for more than 50 frames, then the object will be deregistered.

RESULT: -

While our centroid tracker worked great in this example, there are two primary drawbacks of this object tracking algorithm. The first is that it requires that object detection step to be run on every frame of the input video.

The second drawback is related to the underlying assumptions of the centroid tracking algorithm itself — *centroids must lie close together between subsequent frames.*

- This assumption typically holds, but keep in mind we are representing our 3D world with 2D frames — **what happens when an object overlaps with another one?** The answer is that **object ID switching could occur**.
- If two or more objects overlap each other to the point where their centroids intersect and instead have the minimum distance to the other respective object, the algorithm may (unknowingly) swap the object ID.
- However, the problem is more pronounced with centroid tracking as we relying strictly on the Euclidean distances between centroids and no additional metrics, heuristics, or learned patterns.

CONCLUSION: -

From this project we learned that how to perform object tracking using OpenCV library and Centroid tracking algorithm. The centroid tracking algorithm works by:

- Accepting bounding box coordinates for each object in every frame
- Computing the Euclidean distance between the centroids of the *input* bounding boxes and the centroids of *existing* objects that we already have examined.
- Updating the tracked object centroids to their new centroid locations based on the new centroid with the smallest Euclidean distance.
- And if necessary, marking objects as either "disappeared" or deregistering them completely.

The centroid tracking used has two primary cons:

- We have to run the tracker for every frame of the video.
- Overlapping of objects is not properly handled due to the Euclidean distance and the ids of the objects might end up being swapped.

Despite of its downsides, the tracker is still very efficient with some advantages of its own (1) since we can control the environment of where it is used, there is less worry of objects overlapping and (2) we can use it in real-tim

REFERENCES: -

[1] https://github.com/soeaver/caffe-model

[2] https://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and- tracking-with-python-and-opencv/

[3] Anderson, F. (1990, September). Real time, video image centroid tracker. In *Acquisition, Tracking, and Pointing IV* (Vol. 1304, p. 82). International Society for Optics and Photonics.

[4] Venkateswarlu, R., Sujata, K. V., & Rao, B. V. (1992, November). Centroid tracker and aimpoint selection. In *Acquisition, Tracking, and Pointing VI* (Vol. 1697, pp. 520-530). International Society for Optics and Photonics.

[5] https://www.pyimagesearch.com/2018/07/30/opencv-object-tracking/

[6] https://github.com/lazyoracle/motion-detection

[7] Yin, F., Makris, D., & Velastin, S. A. (2007, October). Performance evaluation of object tracking algorithms. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, Rio De Janeiro, Brazil* (p. 25).

[8] Johnsen, S., & Tews, A. (2009, May). Real-time object tracking and classification using a static camera. In *Proceedings of IEEE International Conference on Robotics and Automation, workshop on People Detection and Tracking*.