Speed Detection using IOT

Rajalakshmi
Assistant Professor
Department of Computer Science and Engineering
Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya
Enathur, Kanchipuram 631502

G. Aravindh
Student,IV Year B.E.
Department of Information Technology
Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya
Enathur, Kanchipuram 631502

A. Kowshik
Student,IV Year B.E.
Department of Information Technology
Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya
Enathur, Kanchipuram 631502

ABSTRACT
Road accidents have been very common in the present world with the prime cause being the careless driving.The necessity to check this has been very `essential and different methods have been used for. However with the advancement in the technology, different governing bodies are demanding some sort of computerized technology to control this problem of over speed driving. At this scenario, we are proposing a system to detect the vehicle which are being driven above the given maximum speed limit that the respective roads or highway lights.

General Terms
VASCAR, Python, Speed Detection.

Keywords
Object tracking, Centroid Tracking, MobileNet SSD.

1. INTRODUCTION
Over speeding vehicles are major issues foe road safety and needs proper addressing to minimize the accidents. Excessive speed is factor in one third of all fatal crashes. Keeping in mind all these issues that are faced, this paper gives a clear view of how the process will flow through the software and hardware. A speed detection means detecting the speed of the vehicles at some certain range using the internet of things. In this, the monitoring of all the vehicles in the road and detecting the speed of the vehicle as well its number plate. If any over speed occurs then it can be detected by the system and information about the driver and will be uploaded to traffic police department so that automatic case filing can be done without any complaint. From this method people will automatically become aware of the speed in which they are riding and thus decreasing the speed leading to a safe environment.

Vehicle Speed Detection is done by using the Object Tracking and the Centroid Tracking where the object tracking is used to detect the objects on the road and the centroid tracking is used to detect the speed of the vehicle. The Algorithm used here is MobileNet SSD algorithm.

2. SYSTEM DESCRIPTION
The system aims to develop a hardware and software based speed detector where the accidents can be reduced and maintain safe environment. Automatic case filing can be done without any complaint. The architecture diagram describes the entire working process of the application. As shown in the architecture,the whole system is automated.

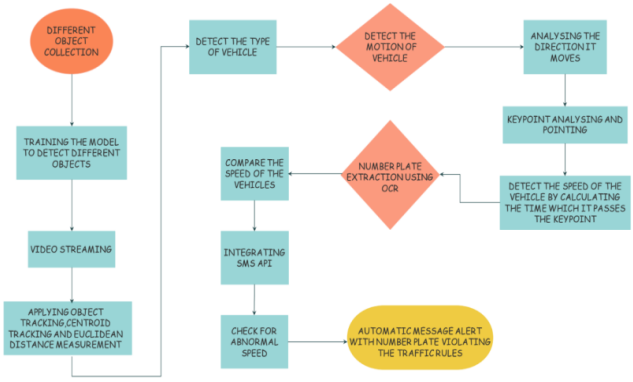

Fig.1 Descriptive System Architecture

The major elements of the system are installed that has a major sub elements exteriorly described in the system which are common.

Hardware Components:

- Raspberry pi 4 with 4GB RAM
- Raspberry pi 3b+
- Two cameras
- Power supply

Software Components:

- OS:linux
- Pycharm(code editor)
- Python

The speed formula is speed=distance/time. It has a known distance measured by a tape at the roadside. The cameras will face at the road perpendicular to the distance measurement unobstructed by obstacles. The meters per pixel are calculated by dividing the distance constant by the frame width in pixels. Distance in pixels is calculated as the difference between the centroids as they pass by the columns for the zone Distance in meters is then calculated for the particular zone. Four timestamps (t) will be collected as the car moves through the FOV past four waypoint columns of the video frame. Three pairs of the four timestamps will be used to determine three delta t values. It will calculate three speed values for each of the pairs of timestamps and estimated distances.The three speed estimates will be averaged for an overall speed

The speed is converted and made available in the Trackable Object class as speed MPH or speed KMPH.

3. IMPLEMENTATION

The objective of the project is to track the speed of the vehicle by four point tracking estimation and recognizes the number plate of the vehicle. The Arduino kit is placed with two cameras. When a vehicle crosses the first camera it detects the speed of the vehicle using speed detecting camera and as it detects over speed it captures the vehicle and stores the data. The image is then processed using the optical character recognition (OCR) to detect the registration number from the number plate.

A simple object tracking algorithm relies on keeping track of the centroids of objects.

Typically an object tracker works hand-in-hand with a less-efficient object detector. The object detector is responsible for localizing an object. The *object* tracker is responsible for keeping track of which object is which by assigning and maintaining identification numbers (IDs).

This object tracking algorithm being implemented 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. The centroid tracking algorithm is a multi-step process. The five steps include:

1. Accept bounding box coordinates and compute centroids
2. Compute Euclidean distance between new bounding boxes and existing objects
3. Update *(x, y)*-coordinates of existing objects
4. Register new objects
5. Deregister old objects

In order to track and calculate the speed of objects in a video stream,it need an easy way to store information regarding the object itself, including:

- Its object ID.
- Its previous centroids (so it can easily compute the direction the object is moving).
- A dictionary of timestamps corresponding to each of the four columns in the frame.
- A dictionary of *x*-coordinate positions of the object. These positions reflect the actual position in which the timestamp was recorded so speed can accurately be calculated.
- The last point boolean serves as a flag to indicate that the object has passed the last waypoint (i.e. column) in the frame.
- The calculated speed in MPH and KMPH. It calculate both and the user can choose to use by a small modification to the driver script.
- A Boolean to indicate if the speed has been estimated (i.e. calculated) yet.
- A Boolean indicating if the speed has been logged in the .csv log file.
- The direction through the FOV the object is traveling (left-to-right or right-to-left).

The Trackable Object constructor accepts an object id and centroid. The centroids list will contain an object's centroid location history. It will have multiple trackable objects — one for each car that is being tracked in the frame.

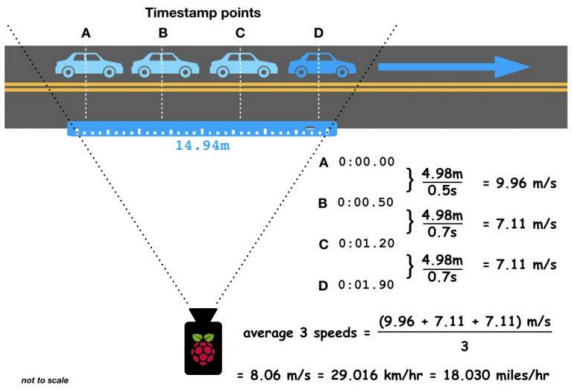

Fig.1 This OpenCV vehicle speed estimation project assumes the camera is aimed perpendicular to the road. Timestamps of a vehicle are collected at waypoints ABCD or DCBA. From there, our speed = distance / time equation is put to use to calculate 3 speeds among the 4 waypoints. Speeds are averaged together and converted to km/hr and miles/hr.

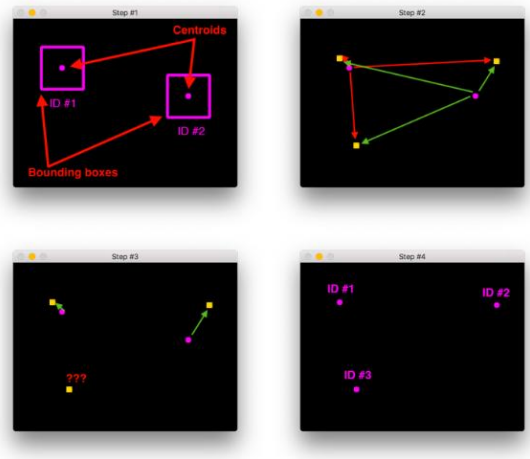

Fig.2 *Top-left*: To build a simple object tracking algorithm using centroid tracking, the first step is to accept bounding box coordinates from an object detector and use them to compute centroids. *Top-right*: In the next input frame, three objects are now present. It need to compute the Euclidean distances between each pair of original centroids (*circle*) and new centroids (*square*). *Bottom-left*: The simple centroid object tracking method has associated objects with minimized object distances. *Bottom-right*: It has a new object that wasn't matched with an existing object, so it is registered as object ID #3.

Once the speed is detected it automatically verifies it with the speed limit. As the vehicle crosses the speed limit it send the image to the higher officials. The image processing is done by OCR(Optical Character Recognition) technique to ensure the vehicle register number.

OpenCV Vehicle Detection, Tracking, and Speed Estimation

```
    # if the *display* flag is set, then display the current frame
    # to the screen and record if a user presses a key
    if conf["display"]:
        cv2.imshow("frame", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key is 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()

# 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()))

# check if the log file object exists, if it does, then close it
if logFile is not None:
    logFile.close()

# close any open windows
cv2.destroyAllWindows()

# clean up
print("[INFO] cleaning up...")
vs.stop()
```

Fig.3 Sample screenshot of source code

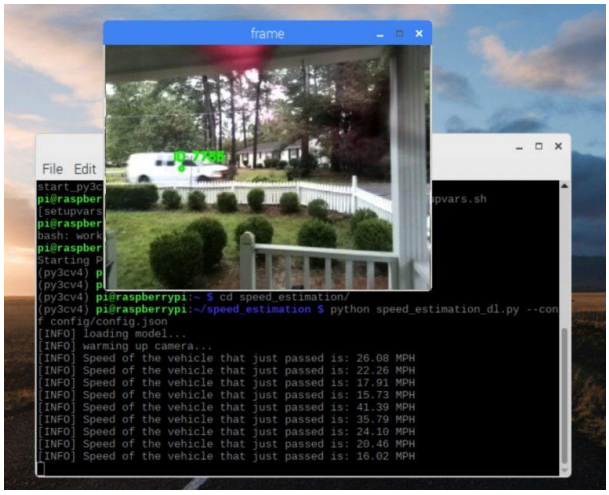

Fig.4 Sample screenshot of the implementation.

3.1 Hardware and Software Implementation

Intel core i5 processor is used with a hard disk comprising of 100GB space to ensure that the smooth functioning of the software adding up with a raspberry pi and an aurdino kit is connected to the system.

The raspberry pi comes with an operating system Linux which is used in overall of this system. The pycharm code editor is used to program the system with the python programming language.

The two cameras are connected to the raspberry kit to capture the image of the vehicles.

4. CONCLUSION

The accidents can be reduced and a safe environment is implemented. This device is suited for all roads which saves the lives of many people and is also used to find the stolen vehicles by using the registered vehicle number.The goal of the project is to save many lives and reduce accidents in a better way. Earlydays, the paper represents the aerial tracking in the hyperspectral domain i.e., the deep tracker performs exceptionally well with deep features setup in a synthetic hyperspectral video generated by the digital imaging and remote sensing image generation(DIRSIG) software where only the type of the type of the vehicle is determined. Also the present VASCAR technique is inefficient to track the speed of the vehicles.

5. FURTHER SCOPE

Every development has some drawback or lack of necessary feature that emerges with the usage and need. The project can be developed even more by adding vehicle detection wireless sensing networks and implementing a wireless sensor network will be another interesting which will open up much more application areas. Advanced image processing algorithms and libraries could be used so that the system can be used efficiently even during unfavourable lighting conditions and during the night time as well.

6. REFERENCES

[1] R. Pelapur, S. Candemir, F. Bunyak, M. Poostchi, G. Seetharaman, and K. Palaniappan, “Persistent target tracking using likelihood fusion in wide-area and full motion video sequences,” in Proc. 15th Int. Conf. Inf. Fusion (FUSION), Jul. 2012, pp. 2420–2427.

[2] J. Portmann, S. Lynen, M. Chli, and R. Siegwart, “People detection and tracking from aerial thermal views,” in Proc. IEEE Int. Conf. Robot. Automat. (ICRA), May/Jun. 2014, pp. 1794–1800.

[3] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. (2016). “ECO:Efficient convolution operators for tracking.” [Online]. Available: https:// arxiv.org/abs/1611.09224

[4] B. Uzkent, M. J. Hoffman, and A. Vodacek, “Efficient integration of spectral features for vehicle tracking utilizing an adaptive sensor,” Proc. SPIE, vol. 9407, p. 940707, Mar. 2015.

[5] B. Uzkent, Real-Time Aerial Vehicle Detection and Tracking Using a Multi-Modal Optical Sensor. Rochester, NY, USA: Rochester Institute Technology, 2016.

[6] B. Uzkent, A. Rangnekar, and M. J. Hoffman, “Aerial vehicle tracking by adaptive fusion of hyperspectral likelihood maps,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jul. 2017, pp. 233–242.

[7] AFRL. (2009). Wright-Patterson Air Force Basevvi (WPAFB) Dataset. [Online]. Available: https://www.sdms.afrl.af.mil/index.php?collection= wpafb2009

[8] AFRL. (2007). WAMI Columbus Large Image Format (CLIF) Dataset.

[9] [Online].Available:https://www.sdms.afrl.af.mil/index.php?collection= clif2007

[10] O. Russakovsky et al., “ImageNet large scale visual recognition challenge,” Int. J. Comput. Vis., vol. 115, no. 3, pp. 211–252, Dec. 2015.

[11] B. Uzkent, M. J. Hoffman, A. Vodacek, J. P. Kerekes, and B. Chen, “Feature matching and adaptive prediction models in an object tracking DDDAS,” Procedia Comput. Sci., vol. 18, pp. 1939–1948, Jan. 2013.

[12] B. Uzkent, M. J. Hoffman, A. Vodacek, and B. Chen, “Feature matching with an adaptive optical sensor in a

ground target tracking system," IEEE Sensors J., vol. 15, no. 1, pp. 510–519, Jan. 2015.

[13] R. D. Meyer, K. J. Kearney, Z. Ninkov, C. T. Cotton, P. Hammond, and B. D. Statt, "RITMOS: A micromirror-based multi-object spectrometer," Proc. SPIE, vol. 5492, pp. 200–219, Sep. 2004.

[14] B. Uzkent, M. J. Hoffman, and A. Vodacek, "Integrating hyperspectral likelihoods in a multidimensional assignment algorithm for aerial vehicle tracking," IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens., vol. 9, no. 9, pp. 4325–4333, Sep. 2016.

[15] S. Han et al., "Efficient generation of image chips for training deep learning algorithms," Proc. SPIE, vol. 10202, p. 1020203, May 2017.

[16] S. Han and J. P. Kerekes, "Overview of passive optical multispectral and hyperspectral image simulation techniques," IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens., vol. 10, no. 11, pp. 4794–4804, Nov. 2017.

[17] B. Uzkent, M. J. Hoffman, and A. Vodacek, "Spectral validation of measurements in a vehicle tracking DDDAS," Procedia Comput. Sci., vol. 51, pp. 2493–2502, Jan. 2015.

[18] B. Uzkent, Real-Time Aerial Vehicle Detection and Tracking Using a Multi-Modal Optical Sensor. Rochester, NY, USA: Rochester Institute Technology, 2016.

[19] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 3, pp. 583–596, Mar. 2015.

[20] S. Hare et al., "Struck: Structured output tracking with kernels," IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 10, pp. 2096–2109, Oct. 2016.