NUMBER PLATE DETECTION AND SPEED ESTIMATION FROM LIVE VIDEO

MACHINE LEARNING MODEL

PROJECT REPORT (Submitted by)

Ayan Panda (21BCY10173)

Dhruv Gupta (21BCY10178)

Rishav Kumar (21BCY10174)

Jayant Barange (21BCY10006)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

PROGRAMME OF STUDY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

VELLORE INSTITUTE OF TECHNOLOGY, BHOPAL UNIVERSITY

KOTHRI KALAN , SEHORE , MADHYA PRADESH - 466114

IN

JANUARY 2023

ACKNOWLEDGEMENT

First and foremost, We would like to thank the Lord Almighty for His presence and immense blessings throughout the project work. We wish to express my heartfelt gratitude to **Dr. Shishir Kumar Shandilya** , Head of the Department, School of Computer Science for much of his valuable support and encouragement in carrying out this. We would like to thank my internal guide **Dr. Adarsh Patel**, Assistant professor Grade 1, School of Computer Science, for continually guiding and actively participating in our project, giving valuable suggestions to complete the project work. We would like to thank all the technical and teaching staff of the School of Computer Science, who extended directly or indirectly all support. Last, but not least, We are deeply indebted to my parents who have been the greatest support while we worked day and night for the project to make it a success.

ABSTRACT

All forms of roadways, including off-roads, motorways, etc., need a well-organized traffic control system. Numerous laws have been passed, and different locations have adopted different speed-control tactics. Also Various roads may have different speed limits. In order to track objects, a number of methods utilizing computer vision and machine learning algorithms have been developed. In this case, videos captured with a security camera show and identify automobiles. The goal is to identify and track automobiles using the Haar Classifier, then calculate the vehicle's speed, and finally detect its license plate. Using machine learning to identify license plates and vehicle speed is a challenging but worthwhile undertaking.Convolution Neural Networks (CNN) have been widely employed in computer vision for the past several years to detect and identify vehicles. Dlibs are used to simultaneously track numerous items.

In general, video surveillance systems are utilized for monitoring and security purposes. However, one difficult aspect of video surveillance is the detection of moving objects. A video surveillance system is utilized for banking, military, and home security. Security at ATMs, traffic vigilance, etc. Human activity identification and monitoring has grown more feasible in recent years thanks to falling prices for high-quality video surveillance systems. As a result, automated systems have been created to perform a variety of detection duties, but the work of identifying unlawfully parked vehicles has generally been delegated to surveillance system operators. The most fascinating and difficult research topic over the past few years has been the identification of Indian automobiles by their license plates.

TABLE OF CONTENT

CHAPTER No.	TITLE	PAGE No.
	ACKNOWLEDGEMENT	**1**
	ABSTRACT	**2**
1.	**CHAPTER - 1 :** **PROJECT DESCRIPTION AND OUTLINE** **1.1 Introduction** **1.2 Motivation for the work** **1.3 Problem Statement**	**5**
2.	**CHAPTER - 2 :** **RELATED WORK AND INVESTIGATION**	**6**
3.	**CHAPTER - 3 :** **METHODOLOGY** **3.1 Proposed System** **3.2 Implementation of the System**	**7**
4.	**CHAPTER - 4 :** **FEASIBILITY OF PROJECT** **4.1.Key Considerations.** **4.2. Feasibility Study of This Project**	**22**
5.	**CHAPTER - 5 :** **EXPERIMENTAL ANALYSIS AND RESULTS** **5.1 System Configurations** **5.2 Source Code** **5.3 Pipeline** **5.4 Database**	**24**
6.	**CHAPTER - 6 :** **CONCLUSION AND FUTURE SCOPE** **6.1 Conclusion** **6.2 Future scope**	**53**
7.	**REFERENCES**	**54**

1. PROJECT DESCRIPTION AND OUTLINE

1.1 Introduction

One of the most crucial ideas in traffic control systems is the calculation of vehicle speed and the identification of license plates. Since speed limit violations must be controlled, drivers should not drive too fast. Additionally, we may work to reduce the number of traffic accidents by enforcing strict traffic laws. The primary goal is to identify overspeeding vehicles, which is followed by automatic license plate recognition and fines for the drivers. Currently employing Radar ideas that require manual input to identify overspeeding automobiles. However, we can recognise overspeeding automobiles without physical intervention thanks to the most recent advances in technology.

Autonomous license plate detection of speeding vehicles is made possible with the aid of computer vision and machine learning technology, and violators may face fines. There are still certain limits in the many approaches and strategies that have been presented to estimate the speed of moving cars, therefore research on license plate recognition and speed estimates is currently ongoing. It is simpler and more affordable to take the video from a security camera. Video cameras are therefore an effective tool for detecting overspeeding vehicles and automatically identifying the vehicle's license plate, especially when taking maintenance and cost into account.

This technique has advantages over the traditional artificial observation alarming method. Since post-accident treatment and responsibility identification are important and need to be addressed in order to use rapidly developing technology to make roads safer, there is a high demand for research in this area. Advantages of detection include reduced manual intervention, quick reaction, high detection rates, and treatment. In this study, automobiles are recognised utilizing machine learning algorithms and computer vision techniques for automatic license plate recognition and vehicle speed determination.

1.2 Motivation for the Work

The traffic monitoring cameras are stationary in urban areas. They will be elevated so that they can see clearly any moving cars on the road. Comparatively speaking, the detection of a vehicle from a stationary camera will be more simple than from a camera that records the dynamic movement of the vehicle and its surroundings. In this study, we analyze films taken by stationary cameras, and we employ an optical-flow based technique to identify several cars. By examining the velocity field, the optical-flow based technique indirectly detects barriers using recognition of numerous images at various periods. The classification of object detection algorithms depends on background subtraction and frame differencing.

The contributions of the proposed work are as follows:

- Using machine learning to identify speeding vehicle and their number plates ; and
- Provide with a proper traffic management system

In order to determine the speeding vehicle and its number plate we have divided our program into 4 steps as to simply the solution.

1.3 Problem Statement

Multiple image processing and algorithmic techniques must be applied inside a single application in order to automatically recognise license plates. In order to locate the license plate number in a given image or video frame, text localization, extraction, and enhancement, character segmentation, and recognition processes are used. The entire process of a typical LPR system, from picture acquisition through verification, was only partially covered by the prior studies. In this study, a full license plate identification system that functions in real time and is based on restrictions has been created. The phase of a typical system that takes the longest is license plate localization and extraction. For LPR systems to be able to locate license plates in real time, assumptions and optimizations are needed.

However, concurrent to this increase in computational needs. Constraints and prior information are used to reduce this adverse effect. The region is subsequently processed for character segmentation and recognition after the license plate area has been extracted.

2. RELATED WORK AND INVESTIGATION

Numerous studies have addressed the same issue of determining the speed of a vehicle using image processing and automatic license plate recognition utilizing machine learning techniques. The first survey is titled "Vehicle Speed Estimation Algorithm Based on DTW Approach" and was published in the IEEE Sensors Journal on April 15, 2017 [4]. According to experimental findings, it has a 96% accuracy rate. The most important area to study time series is DTW. But in DTW, finding the best temporal alignment path is important due to the large computational analytical cost. Additionally, DTW has quadratic complexity, meaning that performance is directly correlated with the input data set's square size. Survey 2: A Creative Motion Plane Based Approach to Estimating Vehicle Speed [5].Here, the center of the license plate is taken into account for estimating speed, and displacement is calculated for each frame to estimate the vehicle's speed.

The license plate's 3D location is taken into account while estimating speed. Problem is that it takes a lot of time. Template IJTSRD33395 International Journal of Trend in Scientific Research and Development (IJTSRD) | Volume 4 | Issue 6 | September-October 2020 | eISSN: 2456-6470 @ IJSRD The Page 488 matching procedures applicable to big image patterns is constrained and laborious. Survey 3: 2018 IEEE International Conference on Applied System Invention (ICASI) [6]: An effective license plate recognition system employing convolution neural networks. Benefits of using a conventional neural network to identify and detect license plates even in blurry photos. Drawbacks include the fact that this method is not used for commercial purposes on highways for moving vehicles.

3. METHODOLOGY

3.1 Proposed System

The presented approach is intended to identify license plates from moving objects. A license plate image captured by a camera serves as the system's input, while its output—the identification of characters on the license plate—occurs in a separate notepad window.

Estimation of vehicle speed, Image capture, License Plate Extraction, License Plate Segmentation, and License Plate Recognition make up the system's typical six key modules for an LPR system. Getting the image is the first task. The area containing the license plate is extracted in the second task.

In a formal system, a variety of sensors, including video sensors, acoustic sensors, loop detectors, ultrasonic sensors, and others, can be used to estimate the vehicle's speed. In this study, footage captured by a security camera is used to determine the vehicle's speed and license plate of the related car.

BLOCK DIAGRAM :

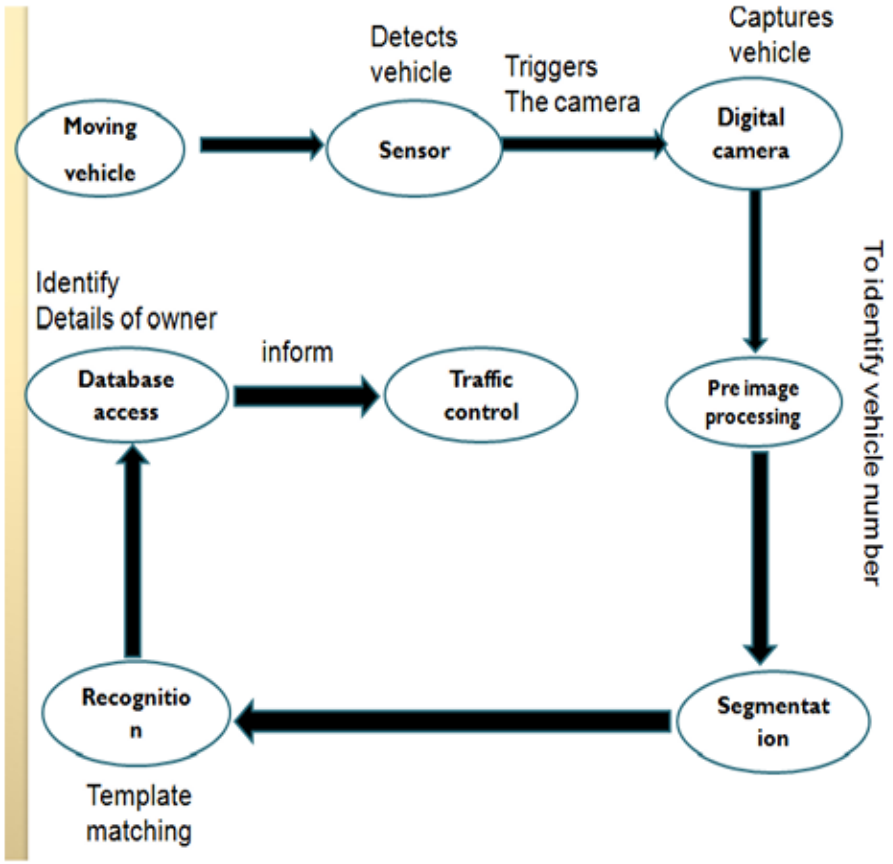


Figure: 1.4 Licence plate recognition

3.1.1 Background Subtraction Method

A practical and reliable technique for spotting moving objects in video footage is background removal. However, this method's usefulness is restricted because it presumes that image fluctuations are exclusively brought on by moving objects (i.e., the backdrop scene is supposed to be stable). The idea behind the method is to identify moving objects by comparing the current frame to a reference frame, often known as a "background image" or "background model." If the image in question is a part of a video feed, background removal is typically performed. The creation of a successful background removal algorithm is fraught with difficulties. It must be resistant to variations in illumination first.

Second, it should avoid picking up moving objects' shadows as well as non

stationary backdrop items like rain, snow, and moving leaves. Finally, the internal background model should respond fast to background changes like vehicle starts and stops.

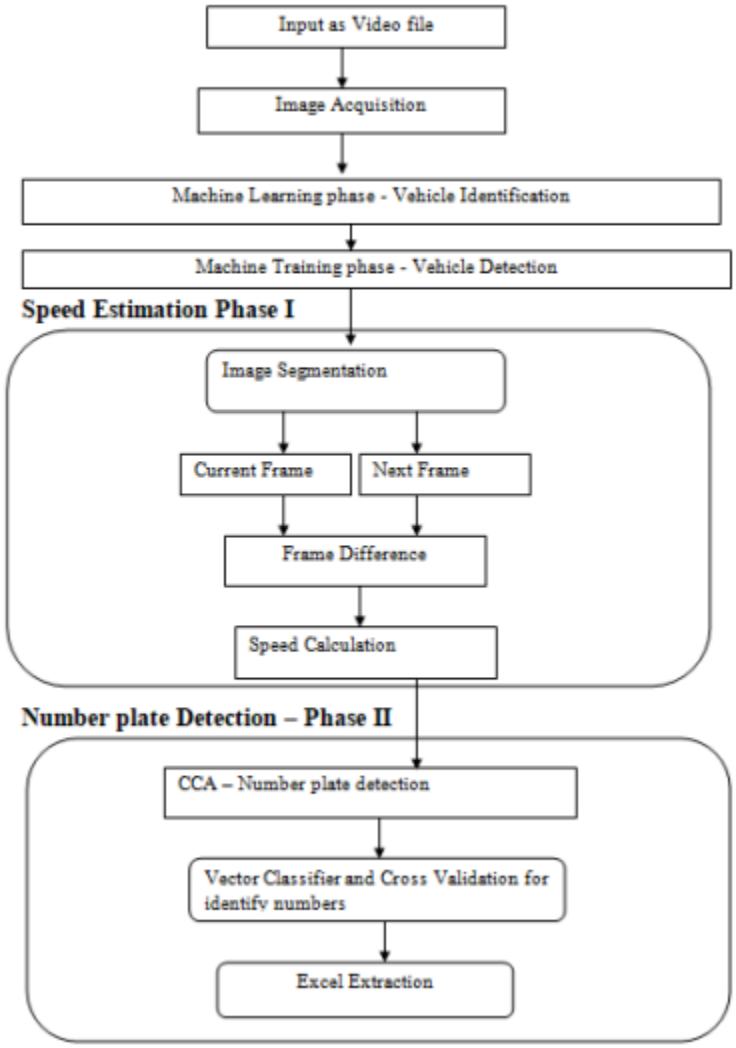
3.1.2 Optical Flow Method:

The pattern of apparent mobility of objects, surfaces, and edges in a visual scene that is brought on by the movement of the observer (an eye or a camera) in relation to the scene is known as optical flow. Optical flow is used to control motion, perceive shape, distance, and movement of things in the world, as well as to perceive the observer's movement in it. The estimation of motion as either instantaneous picture velocities or discrete image displacements, i.e., it corresponds to the movement of pixels in an image, is possible with sequences of ordered images. With optical flow approaches, the motion between two picture frames that are captured at times t and t+at for each vowel location is attempted to be calculated.. However, optical flow techniques take a very long time and have a lot of flaws and unsharp borders.

3.1.3 Adaptive Contrast Change Detection:

By integrating the temporal difference approach and the background removal method into a single algorithm and using the benefits of one algorithm to make up for the drawbacks of the other, adaptive change detection enhances change detection. The above method is useful to some extent because the backdrop detection method is ineffective because of changing backgrounds and shifting lighting conditions, and the temporal difference method only detects constantly moving objects and fails when an object stops. However, only static background situations can be employed with this technique.

3.2 Implementation of the System



3.2.1 DataSet Details

- **Image Acquisition:**

In an LPR system, this is the initial stage. Obtaining an image via an acquisition method is the focus of this phase. In our proposed system, we acquired the input image using a digital camera.

- **License Plate Extraction:**

An important phase in an LPR system is license plate extraction, which has a big impact on the system's accuracy. In this step, the collected image is used to extract the area of

interest, in this case the license plate. The suggested method uses histogram-based analysis to find the license plate's placement in the obtained image.

- **License Plate Segmentation:**

License plate segmentation, also known as character isolation, takes the area of interest and makes an effort to separate it into distinct characters.

- **License Plate Recognition:**

The character in each image can be recognised by alternately segmenting the retrieved license plate into individual character images. There are numerous techniques for identifying isolated characters. This system that we employ Two sets of templates were used to achieve template matching. Perfect characters made up one set. The split license plate characters are part of another set. The character segment and the templates in the database were compared using a correlation function. As the recognised character, the character with the best match was output.

3.2.2 Input Video and Image Acquisition

The majority of major roads are equipped with surveillance cameras that can be set to a specific distance and catch moving vehicles; these videos can be used as input. Today, the majority of homes have it installed because it is relatively simple. Image acquisition is the first stage of video processing in computer vision. After the video has been processed, it creates videos with a certain amount of frames. Additionally, there are numerous ways to analyze a picture in order to do the various visual tasks that are necessary.

However, if the necessary image cannot be acquired properly, the deliberate tasks must be redone. Moving objects need to be tracked in numerous frames in order to determine their speed, which is how speed is measured. Therefore, footage is divided into frames in order to estimate the speed of the vehicle. The vehicle is then traced as it moves from one frame to the next. The estimated speed of the vehicle is based on the distance covered and the time needed to get from one pixel to another. Having very few copies and losses of data is a major advantage of digital picture processing.

Extracting sequence of images from a video, then analysis them using cv2 (Open Source Computer Vision) library

1. Image Enhancement

By adjusting a picture's colors or intensities, image enhancement techniques in Visual Processing Toolbox make it possible to improve the signal-to-noise ratio and highlight image features. A generalized multidimensional filtering function that can handle integer image types, provide different boundary padding options, conduct convolution, and correlate are included in the toolbox along with specific filtering techniques.

Using predefined filters and functions there is a possibility to:

- Filter with morphological operators
- De-blur and sharpen
- Remove noise with linear, median, or adaptive filtering
- Perform histogram equalization
- Remap the dynamic range
- Adjust the gamma value
- Adjust contrast

There are also multi-dimensional filtering methods and specialised filtering procedures. Both predefined filters and tools for creating and using custom filters are available.

2. Image Deblurring:

Along with blind, Lucy-Richardson, Wiener, and regularized filter deconvolution, other image deblurring techniques also convert between point spread and optical transfer functions. These features aid in removing blur brought on by out-of-focus lenses, camera or subject movement during image capture, climatic circumstances, brief exposure times, and other causes. Multidimensional photos can be used with all deblurring techniques.

3. Image Analysis:

Extracting relevant information from photos, such as discovering forms, counting items, recognising colors, or measuring object attributes, is the process of image analysis. For image

analysis tasks like statistical analysis, feature extraction, and property measurement, Image Processing Toolbox offers a complete array of reference-standard methods and visualization features.

3.2.3 Machine Learning – Haar Classifier: Vehicle Identification

If there are several images in each frame, vehicle classification or identification refers to the computer or software system's ability to track down and identify each vehicle in the scene or image. The identification of the objects in a given image is greatly aided by object detection, which is used in applications like face recognition, pedestrian counting, vehicle detection, security systems, web images, and driverless cars.

Numerous applications with rapidly developing technology have exploited these object detecting techniques. It has many uses and incredible creative applications, just like other technologies. Here, a machine learning system known as the Haar cascade classifier is used to detect such cars as objects.

Steps for identifying the objects are:

- Collecting data – Positive and Negative images.
- Training system
- Validating Result

Collecting Data

The gathering of data is the first step in the Haar classifier process, where we may need to train the system using several positive and negative images.Here Positive images such as vehicles and Negative Images are other than Vehicles.

- Vehicle
- Non Vehicle Images

It is required to assemble thousands of images i.e. Positive Images – True images of vehicles which need to be recognized by the system. Negative Images – False Images, which system wants to omit. Applying the positive and Negative Images to the system makes it identify which is vehicle and which is non – vehicle.

Vehicle detection in Haar cascade classifier has 4 stages:

- Selection of objects to be identified
- Modeling Integral Images
- Adaptive boosting Training
- Cascading Classifiers for multiple detect : Cascade Classifier and detect Multi Scale

The system gradually learns the true and false items using the training set. The system's ability to distinguish between genuine and fake images may be tested using the validation set.The system is able to distinguish between vehicles and non-vehicles when negative images are fed into it.

Fig 1-Positive Images

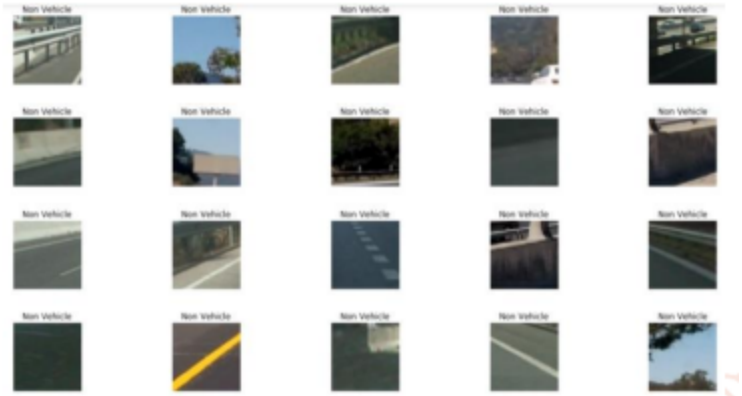


Fig 2- Negative Images

A crucial and difficult duty is always capturing all of the vehicles in the frame. Therefore, we must train the system using a large number of photos from both the training set and the validation set. For vehicle detection, it employs Haar characteristics, which is akin to facial recognition. One of the efficient object detection techniques put out by Viola and Jones uses Haar cascade classifiers.

Always getting all the vehicles in the frame is an essential and challenging task. In order to train the system, we must use a large number of images from both the training set and the validation set. It uses Haar features, which are related to facial recognition, for vehicle detection. Haar cascade classifiers are one of the effective object detection methods proposed by Viola and

Jones.

3.2.4 Tracking of Vehicle

Here, the term "tracking" refers to the process by which each and every vehicle is recognised and monitored every frame in order to calculate its speed. It can be done by utilizing the correlation tracker function of the dlibs function in Python. By giving each vehicle a tracking ID, it is possible to track many vehicles at once and keep track of their movement from one frame to the next in each frame.

By figuring out how far an object has travelled from one frame to the next, speed can be determined. In each frame, an object moved from one pixel to the next. The speed of the vehicle is determined iteratively by taking the average speed throughout a range of speeds and reporting that speed. Since the speed is collected and saved inside every two frames, we can determine the average speed and return it to the user within ROI. Similar to how the correlation ID, tracking process, and tracking ID are withdrawn when the object leaves the region of the index.

Multiple tracking of the same vehicle can therefore be prevented. Thus, the primary goals of this phase are to identify the object, track it in each frame, calculate the distance travelled in each frame with ROI, and remove the tracking ID after the object leaves ROI. We might be able to estimate the vehicle's average speed at the conclusion. One Python library function called Dlibs makes it possible to track many items simultaneously.

3.2.5 Estimation of speed

In general, distance traveled by the same item across frames can be used to measure speed. The

pixels per meter traveled by the vehicle is then calculated using the customary way of employing Euclidean distance calculation.Here, we've begun our calculations by taking and adjusting the video's WIDTH and HEIGHT. We believed that these were crucial in estimating the vehicle's speed. Let Cn(a,b) and Cn+1(c,d) be the points of the object in frames n and n + 1, respectively, for the formal approach of the Euclidean distance that is used to determine the speed of the vehicle.

The distance d1 is calculated by Euclidean distance is mentioned below:

$$d = \sqrt{(a-c)^2 + (b-d)^2}$$

The phrase "pixels per meter" is used. The WIDTH and HEIGHT of the video, which differ in each video taken from the camera, are used to manually estimate this number. Therefore, values may differ from each movie and must be manually adjusted. These values can be found with the use of video processing.In determining the vehicle's speed, the ROAD WIDTH is also crucial. The distance traveled by the vehicle between frames is indicated by the Euclidean computation above with the variable d pixels. The distance in meters, say d meter, would then need to be computed for the standard conversion, which is done as follows.

D_meters = d_pixels / pixels per meter

The number of frames that may be collected in a second, or frames per second, is an additional intriguing factor. Therefore, the speed of the vehicle can be approximated in km/hr by knowing

these parameters.

$$Speed = D\_meters * frames\ per\ second * 3.6$$

which gives the speed of the vehicle in km/hr(3.6 is standard conversion of converting meter into kilometer).The bounding box is applied to the vehicle inside the ROI, and each object is tracked using the correlation tracker.Over the bounding box, the vehicle's average speed is shown. The speed is thus indicated and shown over each car in the produced video.

3.2.6 License Plate Recognition.

The principle of optical character recognition is used by license plate recognition to read each character on the number plate one at a time. Using only images of the car as input, ALPR uses machine learning algorithms to decipher the characters. The training method for this machine learning technique can increase accuracy. to discern As, Bs, etc., by mapping a character-like representation to its actual characters.

LPR is also called ALPR (Automatic License Plate Recognition) that consists of 3 major stages.

- 1. Detecting the License Plate.
- 2. Character Segmentation.
- 3. Character Recognition.

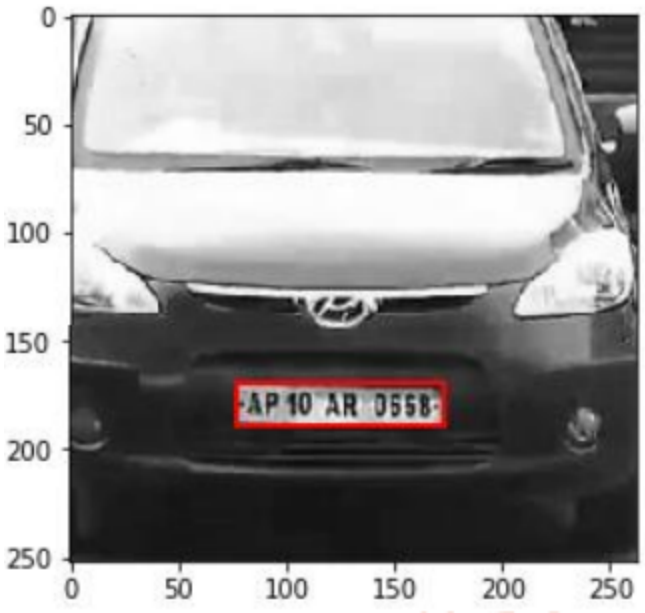
Approach as follows:

- 1. Detects the contours of the image.

- 2. Find the Bounding rectangle of all contours.
- 3. Compare the validate the height of the License plate ratio.
- 4. Apply the image segmentation to find the characters in it.
- 5. Using OCR method with KNN and SVC model under supervised learning, recognize the characters in the license plate.

3.2.7 Detecting License Plate

In this stage the License plate of the vehicle is detected. But it's particularly difficult to detect the position, too many styles and patterns.



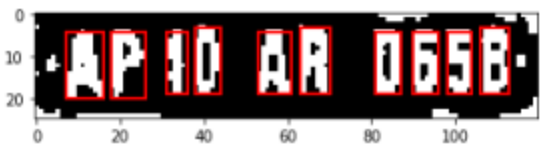
It is relatively possible to identify the license plate from the image of the vehicle using machine learning classification algorithms. Cropping is done with the intention of keeping just the

original image's regions with successful categorization and discarding the rest. As a result, the region detection and picture classification technique is crucial in identifying the vehicle's license plate. The connected region can be grouped and given a name using the connected component analysis approach.

3.2.8 Character segmentation.

Character segmentation is critical in the OCR Process. This step aids in segmenting the license plate characters into individual images using CCA. Characters are split from the license plate's cropped image, and each character is enlarged to 20x20 pixels as a distinct image. The list of regions that are comparable to the license plate region is used to refer to the plate like objects. Three areas of the photograph were picked out as potential locations for a license plate. This method aids in locating the actual information on the license plate image and removing any unnecessary areas. Then the license plate underwent a Connected component analysis, and each character's image was reduced in size to 20px * 20px. The system is able to recognise characters from segmented character images.

A column list is presented in which each image is identified sequentially in order to keep track of the order of the image, or segmented character. In order to print it, it can then be sorted according to how the License plate string can be identified.



3.2.9 Character Recognition.

It is the Final stage of License plate detection. . Here, machine learning methods are used to automatically identify the character on the licence plate. Simply said, it is the area of artificial intelligence that works with data and processes it to find patterns that can be applied to future predictions. Supervised, unsupervised, and reinforcement learning are the three main subcategories of machine learning. The best accuracy is obtained by supervised learning because the system is taught using a fixed dataset and predicted predictions. We therefore know how each character should be for character recognition. This is when supervised learning is helpful. Regression and classification are the two categories into which supervised learning may be subdivided. The K-Nearest Neighbor technique is used for character recognition in supervised character learning, which is a classification category.

Training data set is provided to choose a supervised learning classifier, train a model, test the model and see how accurate it is, then use the model for prediction.

4. FEASIBILITY STUDY

If resources and time were unlimited, all undertakings would be possible. But resource shortages and slow delivery times are major problems in software development. It is both wise and necessary to assess the project's viability as soon as possible.

If resources and time were unlimited, all undertakings would be possible. But resource shortages and slow delivery times are major problems in software development. It is both wise and necessary to assess the project's viability as soon as possible.

4.1.Key Considerations.

Three key considerations are involved in feasibility analysis.

4.1.1.Economic Feasibility

It decides if the project aim can be accomplished within the constraints of the resources allotted to it. Additionally, it decides whether the benefits of the new system outweigh the cost of processing the entire project. Nowadays computers are not considered an economic issue. Hence this technique is well within the budget of almost all the people

4.1.2.Economic Feasibility

Technical viability focuses on the hardware, software, etc. of the current computer system and how well it can handle the suggested edition. Technical limitations do not significantly affect the viability of this procedure. This system is feasible on the

following grounds:

- All necessary technology is available to those who want to develop an application.
- The existing resources are capable of holding all the necessary resources in an efficient way.
- The system is too flexible and can be expanded further whenever new modules are added.

4.1.3. Operational Feasibility

This establishes whether the suggested solution has met user objectives and can be integrated into existing operations. The suggested will undoubtedly meet user needs and improve their capabilities. It is most easily incorporated into ongoing processes. Additionally, system upkeep is quite simple and involves little work. The system is therefore operationally viable.

4.2. Feasibility Study of This Project

Feasibility studies are the first prerequisite of the engineering process. A brief description of the system serves as the feasibility study's input. A report outlining the results of the feasibility study makes recommendations regarding whether or not to move forward with requirements engineering and the system development process. Information assessment, gathering, and report writing are all required to complete the feasibility study.

5. EXPERIMENTAL ANALYSIS AND RESULT

5.1 System Configuration

The system configuration includes Software and Hardware requirement, which are provided below

5.1.1 Software Requirements

Operating System : Windows 7 or higher

Programming : Python 3.6 and related libraries

➔ **Python**

Python is a high-level, interpreted general-purpose programming language. Python, which was developed by Guido van Rossum and originally made available in 1991, stresses code readability and makes extensive use of whitespace. It offers building blocks that make it possible to programme clearly on both small and big scales.

Python has an autonomous memory management system and a dynamic type system. It includes a sizable and thorough standard library, supports a variety of programming paradigms, including imperative, functional, procedural, and object-oriented.

The community-based development model is shared by nearly all of Python's variant implementations, including CPython, the standard implementation. The nonprofit Python Software Foundation is in charge of running CPython.

5.1.2 Hardware Requirements

Processor : **Any Processor above 500 MHz.**

Ram : **4 GB**

Hard Disk : **4 GB**

Input device : **Standard Keyboard and Mouse.**

Output device : **VGA and High Resolution Monitor.**

5.2 Source Code:

0. Setup Paths

In [1]:
```
import os
```

In [2]:
```
import tensorflow as tf
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
tf.config.list_physical_devices('GPU')
```

```
Num GPUs Available:  0
```

Out[2]:
```
[]
```

In [3]:
```
tf.config.list_physical_devices('GPU')
```

Out[3]:
```
[]
```

In [11]:
```
!pip install cuda==11.2 cudnn==8.1.0
```

```
Looking in indexes: https://pypi.org/simple, https://pypi.ngc.nvidia.com
ERROR: Could not find a version that satisfies the requirement cuda==11.2 (from
versions: none)
ERROR: No matching distribution found for cuda==11.2
```

In [2]:
```
CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
PRETRAINED_MODEL_URL = 'http://download.tensorflow.org/models/object_detection/t
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'label_map.pbtxt'
```

In [3]:
```
paths = {
    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
    'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),
    'APIMODEL_PATH': os.path.join('Tensorflow','models'),
    'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace','annotations'),
    'IMAGE_PATH': os.path.join('Tensorflow', 'workspace','images'),
    'MODEL_PATH': os.path.join('Tensorflow', 'workspace','models'),
    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow', 'workspace','pre-trainec
    'CHECKPOINT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MC
    'OUTPUT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_
    'TFJS_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAM
    'TFLITE_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_N
    'PROTOC_PATH':os.path.join('Tensorflow','protoc')
 }
```

In [4]:
```
files = {
    'PIPELINE_CONFIG':os.path.join('Tensorflow', 'workspace','models', CUSTOM_MC
    'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'], TF_RECORD_SCRIPT_NAM
    'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
}
```

In [5]:
```
for path in paths.values():
    if not os.path.exists(path):
        if os.name == 'posix':
            !mkdir -p {path}
```

```
        if os.name == 'nt':
            !mkdir {path}
```

1. Download TF Models Pretrained Models from Tensorflow Model Zoo and Install TFOD

In [6]:
```
# https://www.tensorflow.org/install/source_windows
```

In [8]:
```
if os.name=='nt':
    !pip install wget
    import wget
```

```
Looking in indexes: https://pypi.org/simple, https://pypi.ngc.nvidia.com
Requirement already satisfied: wget in d:\project2\asap\lib\site-packages (3.2)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
```

In [9]:
```
if not os.path.exists(os.path.join(paths['APIMODEL_PATH'], 'research', 'object_d
    !git clone https://github.com/tensorflow/models {paths['APIMODEL_PATH']}
```

In [10]:
```
# Install Tensorflow Object Detection
if os.name=='posix':
    !apt-get install protobuf-compiler
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --p

if os.name=='nt':
    url="https://github.com/protocolbuffers/protobuf/releases/download/v3.15.6/p
    wget.download(url)
    !move protoc-3.15.6-win64.zip {paths['PROTOC_PATH']}
    !cd {paths['PROTOC_PATH']} && tar -xf protoc-3.15.6-win64.zip
    os.environ['PATH'] += os.pathsep + os.path.abspath(os.path.join(paths['PROTC
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --p
    !cd Tensorflow/models/research/slim && pip install -e .
```

In [14]:
```
if os.name =='posix':
    !wget {PRETRAINED_MODEL_URL}
    !mv {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.t
if os.name == 'nt':
    wget.download(PRETRAINED_MODEL_URL)
    !move {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.t
```

```
100% [................................................................................]
```

2. Create Label Map

In [6]:
```
labels = [{'name':'licence', 'id':1}]

with open(files['LABELMAP'], 'w') as f:
    for label in labels:
        f.write('item { \n')
        f.write('\tname:\'{}\'\n'.format(label['name']))
        f.write('\tid:{}\n'.format(label['id']))
        f.write('}\n')
```

3. Create TF records

In [20]:
```
# OPTIONAL IF RUNNING ON COLAB
ARCHIVE_FILES = os.path.join(paths['IMAGE_PATH'], 'archive.tar.gz')
```

```
if os.path.exists(ARCHIVE_FILES):
  !tar -zxvf {ARCHIVE_FILES}
```

In [18]:
```
!pip install pytz
```

```
Looking in indexes: https://pypi.org/simple, https://pypi.ngc.nvidia.com
Requirement already satisfied: pytz in d:\project2\asap\lib\site-packages (202
2.7.1)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
```

In [16]:
```
!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'train
!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'test'
```

```
Successfully created the TFRecord file: Tensorflow\workspace\annotations\train.
record
Successfully created the TFRecord file: Tensorflow\workspace\annotations\test.r
ecord
```

4. Copy Model Config to Training Folder

In [17]:
```
if os.name =='posix':
    !cp {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'pi
if os.name == 'nt':
    !copy {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, '
```

```
        1 file(s) copied.
```

5. Update Config For Transfer Learning

In [8]:
```
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
```

In [9]:
```
config = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
```

In [20]:
```
config
```

In [21]:
```
pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "r") as f:
    proto_str = f.read()
    text_format.Merge(proto_str, pipeline_config)
```

In [22]:
```
pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint = os.path.join(paths['PRETRAIN
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= files['LABELMAP']
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] = [os.pa
pipeline_config.eval_input_reader[0].label_map_path = files['LABELMAP']
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] = [os.
```

In [23]:
```
config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "wb") as f:
    f.write(config_text)
```

6. Train the model

In [28]:
```
!pip install cv2
```

```
Looking in indexes: https://pypi.org/simple, https://pypi.ngc.nvidia.com
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-packages)
ERROR: Could not find a version that satisfies the requirement cv2 (from versions: none)
ERROR: No matching distribution found for cv2
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-packages)
```

In [29]:
```
TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detec
```

In [44]:
```
command = "python {} --model_dir={} --pipeline_config_path={} --num_train_steps=
```

In [45]:
```
print(command)
```

```
python Tensorflow\models\research\object_detection\model_main_tf2.py --model_dir=Tensorflow\workspace\models\my_ssd_mobnet --pipeline_config_path=Tensorflow\workspace\models\my_ssd_mobnet\pipeline.config --num_train_steps=10000
```

In [ ]:
```
!{command}
```

7. Evaluate the Model

In [20]:
```
command = "python {} --model_dir={} --pipeline_config_path={} --checkpoint_dir={
```

In [21]:
```
print(command)
```

```
python Tensorflow\models\research\object_detection\model_main_tf2.py --model_dir=Tensorflow\workspace\models\my_ssd_mobnet --pipeline_config_path=Tensorflow\workspace\models\my_ssd_mobnet\pipeline.config --checkpoint_dir=Tensorflow\workspace\models\my_ssd_mobnet
```

In [22]:
```
!{command}
```

```
^C
```

8. Load Train Model From Checkpoint

In [178…
```
import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util
```

In [179…
```
# Load pipeline config and build a detection model
configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'], is_training

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-12')).expect_partial()

@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections
```

9. Detect from an Image

In [45]:
```
!pip install --upgrade opencv-python
```

```
Requirement already satisfied: numpy>=1.17.0 in d:\project2\asap\lib\site-packa
ges (from opencv-python) (1.24.1)
Installing collected packages: opencv-python
  Attempting uninstall: opencv-python
    Found existing installation: opencv-python 4.5.5.64
    Uninstalling opencv-python-4.5.5.64:
      Successfully uninstalled opencv-python-4.5.5.64
Successfully installed opencv-python-4.7.0.68
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
    WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-
packages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -rotobuf (d:\project2\asap\lib\site-pack
ages)
```

In [180…
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

In [181…
```
category_index = label_map_util.create_category_index_from_labelmap(files['LABEL
```

In [182…
```
IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'test', '3_148.jpg')
```

In [183…

```
img = cv2.imread(IMAGE_PATH)
image_np = np.array(img)

input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float3
detections = detect_fn(input_tensor)

num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
              for key, value in detections.items()}
detections['num_detections'] = num_detections

# detection_classes should be ints.
detections['detection_classes'] = detections['detection_classes'].astype(np.int6

label_id_offset = 1
image_np_with_detections = image_np.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
            image_np_with_detections,
            detections['detection_boxes'],
            detections['detection_classes']+label_id_offset,
            detections['detection_scores'],
            category_index,
            use_normalized_coordinates=True,
            max_boxes_to_draw=5,
            min_score_thresh=.1,
            agnostic_mode=False)

plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
plt.show()
```

In [101… `!pip install easyocr`

```
Collecting easyocr
  Using cached easyocr-1.6.2-py3-none-any.whl (2.9 MB)
Requirement already satisfied: numpy in d:\project2\asap\lib\site-packages (fro
m easyocr) (1.24.1)
Collecting torchvision>=0.5
  Downloading torchvision-0.14.1-cp310-cp310-win_amd64.whl (1.1 MB)
     ---------------------------------------- 0.0/1.1 MB ? eta -:--:--
     ------------ -------------------------- 0.4/1.1 MB 11.6 MB/s eta 0:00:01
     ---------------------------------- ----- 0.9/1.1 MB 11.9 MB/s eta 0:00:01
     ---------------------------------------- 1.1/1.1 MB 9.9 MB/s eta 0:00:00
Collecting torch
  Downloading torch-1.13.1-cp310-cp310-win_amd64.whl (162.6 MB)
```

```
     ---------------------------------------- 2.3/2.3 MB 2.2 MB/s eta 0:00:00
Requirement already satisfied: typing-extensions in d:\project2\asap\lib\site-p
ackages (from torch) (4.4.0)
Requirement already satisfied: requests in d:\project2\asap\lib\site-packages
(from torchvision) (2.28.2)
Requirement already satisfied: numpy in d:\project2\asap\lib\site-packages (fro
m torchvision) (1.24.1)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in d:\project2\asap\lib\si
te-packages (from torchvision) (9.4.0)
Requirement already satisfied: charset-normalizer<4,>=2 in d:\project2\asap\lib
\site-packages (from requests->torchvision) (3.0.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in d:\project2\asap\lib\si
te-packages (from requests->torchvision) (1.26.14)
Requirement already satisfied: idna<4,>=2.5 in d:\project2\asap\lib\site-packag
es (from requests->torchvision) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in d:\project2\asap\lib\site-
packages (from requests->torchvision) (2022.12.7)
Installing collected packages: torchaudio
Successfully installed torchaudio-0.13.1+cu116
```

In [184…
```
import easyocr
```

In [185…
```
detection_threshold=0.6
```

In [186…
```
image=image_np_with_detections
scores=list(filter(lambda x:x>detection_threshold,detections['detection_scores']
boxes=detections['detection_boxes'][:len(scores)]
classes=detections['detection_classes'][:len(scores)]
```

In [187…
```
boxes
```

Out[187]:
```
array([[0.59731805, 0.25718862, 0.6636981 , 0.4709754 ]], dtype=float32)
```

In [188…
```
classes
```

Out[188]:
```
array([0], dtype=int64)
```

In [189…
```
scores
```

Out[189]:
```
[0.6805661]
```

In [190…
```
width=image.shape[1]
height=image.shape[0]
```

In [191…
```
height
```

Out[191]:
```
900
```

In [192…
```
width
```

Out[192]:
```
1200
```

In [193…
```
for idx,box in enumerate(boxes):
    roi = box*[height,width,height,width]
    region=image[int(roi[0]):int(roi[2]),int(roi[1]):int(roi[3])]
    reader=easyocr.Reader(['en'])
    ocr_result=reader.readtext(region)
    print(ocr_result)
    plt.imshow(cv2.cvtColor(region,cv2.COLOR_BGR2RGB))
```

```
[([[2, 4], [250, 4], [250, 58], [2, 58]], 'T0322UP4229B', 0.6839579537793872)]
```

In [199…
```
region_threshold=0.6
```

In [200…
```
def filter_text(region,ocr_result,region_threshold):
    rectangle_size=region.shape[0]*region.shape[1]
    plate=[]
    for result in ocr_result:
        length=np.sum(np.subtract(result[0][1], result[0][0]))
```

```
        width=np.sum(np.subtract(result[0][2], result[0][1]))

        if length*height/rectangle_size>region_threshold:
            plate.append(result[1])
    return plate
```

In [201…
```
filter_text(region,ocr_result,region_threshold)
```

Out[201]:
```
['T0322UP4229B']
```

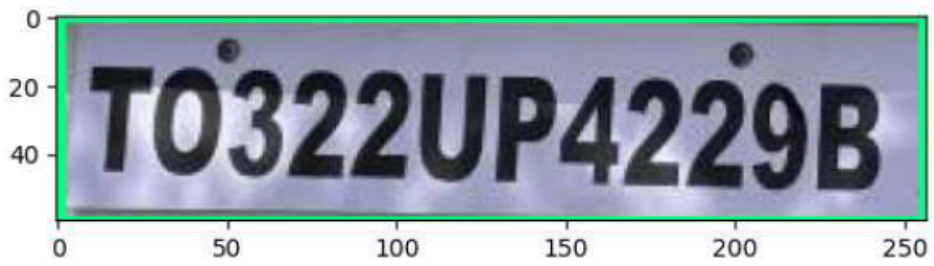
In [197…
```
region_threshold
```

Out[197]:
```
0.6
```

In [96]:
```
def ocr_it(image,detections,detection_threshold,region_threshold):
    scores=list(filter(lambda x:x>detection_threshold,detections['detection_scor
    boxes=detections['detection_boxes'][:len(scores)]
    classes=detections['detection_classes'][:len(scores)]

    width=image.shape[1]
    height=image.shape[0]
    for idx,box in enumerate(boxes):
        roi = box*[height,width,height,width]
        region=image[int(roi[0]):int(roi[2]),int(roi[1]):int(roi[3])]
        reader=easyocr.Reader(['en'])
        ocr_result=reader.readtext(region)
        text=filter_text(region,ocr_result,region_threshold)

        plt.imshow(cv2.cvtColor(region,cv2.COLOR_BGR2RGB))
        print(text)
        return text,region
```

In [97]:
```
text,region=ocr_it(image_np_with_detections,detections,detection_threshold,regic
```

```
['T0322UP4229B']
```



In [169…
```
import csv
import uuid
import io
```

In [170…
```
uuid.uuid1()
```

Out[170]:
```
UUID('8c7c252d-a6bc-11ed-8405-5414f3aa0e41')
```

In [175…
```
def save_results(text, region, csv_filename, folder_path):
    img_name = '{}.jpg'.format(uuid.uuid1())

    cv2.imwrite(os.path.join(folder_path, img_name), region)
```

In [176…
```
save_results(text, region, 'detection_results.csv', 'Detection_Images')
```

10. Real Time Detections from your Webcam

In [16]:
```
!pip uninstall opencv-python-headless -y
```

In [202…

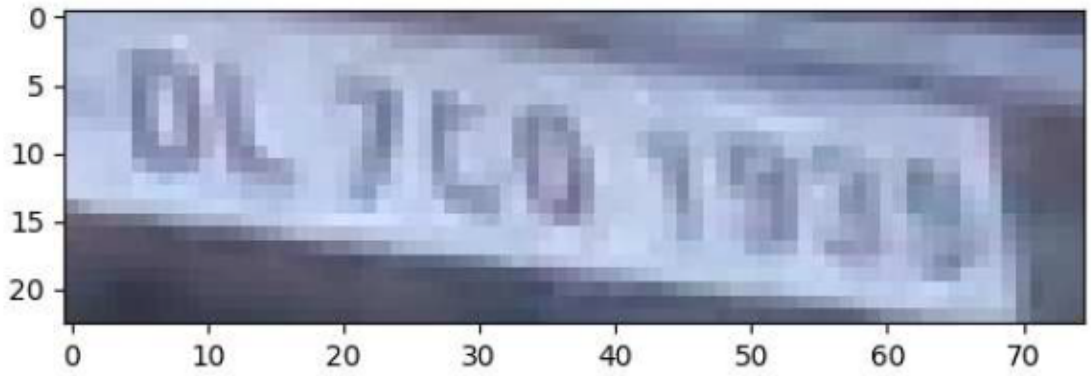
```
cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

while cap.isOpened():
    ret, frame = cap.read()
    image_np = np.array(frame)

    input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.fl
    detections = detect_fn(input_tensor)

    num_detections = int(detections.pop('num_detections'))
    detections = {key: value[0, :num_detections].numpy()
                  for key, value in detections.items()}
    detections['num_detections'] = num_detections

    # detection_classes should be ints.
    detections['detection_classes'] = detections['detection_classes'].astype(np.

    label_id_offset = 1
    image_np_with_detections = image_np.copy()

    viz_utils.visualize_boxes_and_labels_on_image_array(
                image_np_with_detections,
                detections['detection_boxes'],
                detections['detection_classes']+label_id_offset,
                detections['detection_scores'],
                category_index,
                use_normalized_coordinates=True,
                max_boxes_to_draw=5,
                min_score_thresh=.8,
                agnostic_mode=False)
    try:
        text,region=ocr_it(image_np_with_detections,detections,detection_threshc
        save_results(text, region, 'realtimeresults.csv', 'Detection_Images')
    except:
        pass
```

```
    cv2.imshow('object detection',  cv2.resize(image_np_with_detections, (800, 6

    if cv2.waitKey(10) & 0xFF == ord('q'):
        cap.release()
        cv2.destroyAllWindows()
        break
```

```
CUDA not available - defaulting to CPU. Note: This module is much faster with a
GPU.
CUDA not available - defaulting to CPU. Note: This module is much faster with a
GPU.
['DL 7C0 1939']
CUDA not available - defaulting to CPU. Note: This module is much faster with a
GPU.
['dL 7C0', '1939']
CUDA not available - defaulting to CPU. Note: This module is much faster with a
GPU.
['DL7CO 1939']
CUDA not available - defaulting to CPU. Note: This module is much faster with a
GPU.
['OL ZC0 1939]']
CUDA not available - defaulting to CPU. Note: This module is much faster with a
GPU.
[]
['DL 7C0 1939']
CUDA not available - defaulting to CPU. Note: This module is much faster with a
GPU.
CUDA not available - defaulting to CPU. Note: This module is much faster with a
GPU.
['Ibladcontont']
CUDA not available - defaulting to CPU. Note: This module is much faster with a
GPU.
['OL 7c0 1939']
CUDA not available - defaulting to CPU. Note: This module is much faster with a
GPU.
['OLZCO 1939']
['dL 7c0 !939]']
```

```
GPU.
[]
CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
['1', '1']
CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
['DL 7C0 1939']
CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
["~'4Y"]
CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
[]
CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
```

```
['oL 7c0 1939']
CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
['DL 7C0 1939']
CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
['DL 7C0 1939']
CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
['DL7C0 1939']
['DL 7C0 1939']
CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
['0LZL0']
```



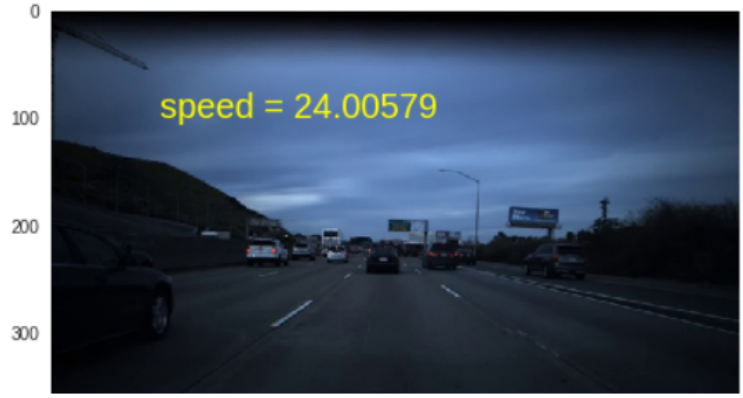
Video frames to images

In [0]:
```
import cv2

vidcap = cv2.VideoCapture('/content/data/train.mp4')
success,image = vidcap.read()
count = 0
success = True
while success:
  cv2.imwrite("/content/data/frames/%d.jpg" % count, image)
  success,image = vidcap.read()
  count += 1
```

In [0]:
```
# utils
import matplotlib.pyplot as plt
import matplotlib.image as mpimg


# get image index with path as input
def get_image_idx(path):
  idx = path.split('/')[4].split('.')[0]
  return int(idx)

# get speed from txt with idx
data_df = pd.read_csv('/content/data/train.txt', header=None)
def get_speed(idx):
  return data_df.iloc[idx][0]

def view_img_speed(img_path):
#   img_path = '/content/data/frames/'+frames_list[0]
  print(img_path)
  img = mpimg.imread(img_path)
  print(img.shape)
  plt.text(100,100,'speed = '+str(get_speed(get_image_idx(img_path))), fontsiz
  plt.imshow(img)
  plt.grid(False)
  plt.show()
```

In [6]:
```
for i in range(200,205):
  img_path = '/content/data/frames/'+str(i)+'.jpg'
  view_img_speed(img_path)
```


```
/content/data/frames/201.jpg
(480, 640, 3)
```



Raw Data to Numpy to Tensors

In [7]:
```
for i, row in data_df.iterrows():
  print(i, row[0])
  img_path = "/content/data/frames/{}.jpg".format(i)
  view_img_speed(img_path)
  if i>=5:break
```

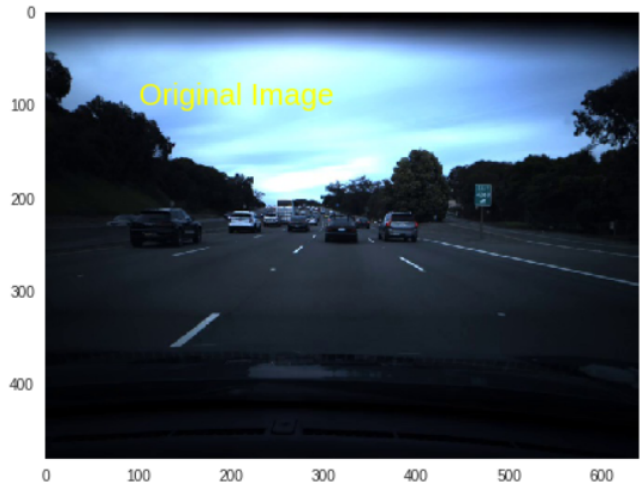

```
1 28.105569
/content/data/frames/1.jpg
(480, 640, 3)
```

480 x 640 is computationally expensive(atleast for GPU i am using) and also the sky and road doesn't give much information regarding the speed.

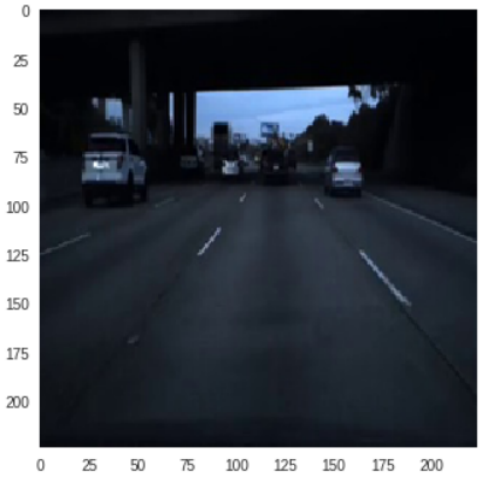
The speed of the vehicle will be calculated by the relative distance of the vehicle form other, so its better to remove unnecessary data from the image.

I am cropping 220x220 from center which will contain most of the necessary data. Range can be increased for better performance but due to my GPU Memory , i will stick to 240x240 , will increase it if possible later.

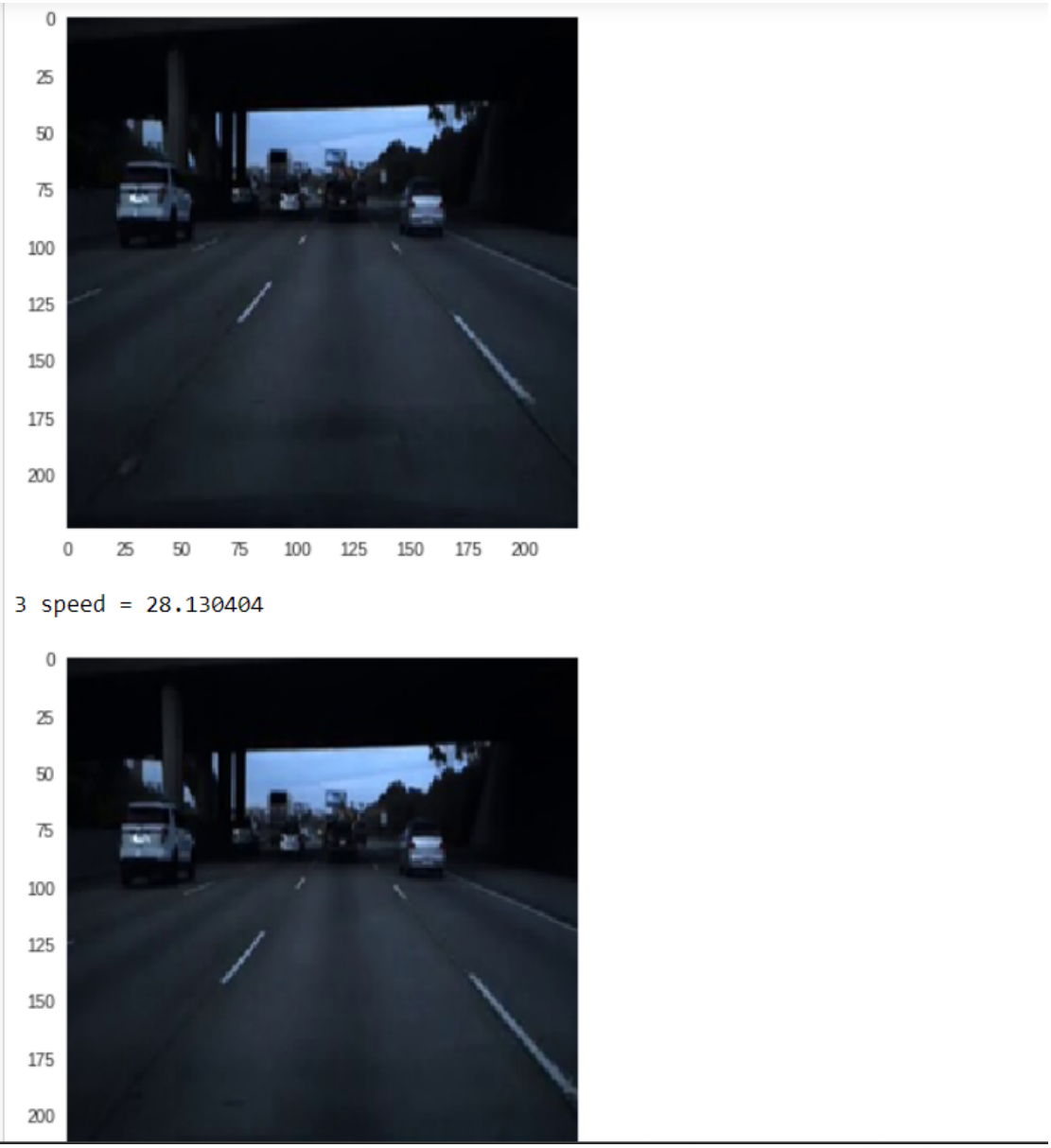
In [2]:
```
from PIL import Image


img = Image.open("/content/data/frames/1000.jpg")


img1 = img.crop((90,150,500,370))
# [130:350,210:450]
plt.imshow(img)
plt.text(100,100,'Original Image', fontsize=20, color='yellow')
plt.grid(False)
plt.show()

plt.imshow(img1)
plt.text(100,100,'Cropped Image', fontsize=20, color='yellow')
plt.grid(False)
plt.show()

import numpy as np
img2 = img1.resize((224,224), Image.ANTIALIAS)

plt.imshow(img2)
plt.text(100,100,'Resized Image', fontsize=20, color='yellow')
plt.grid(False)
plt.show()



def crop_resize_img(img):
  img = img.crop((90,150,500,370))
  img = img.resize((224,224), Image.ANTIALIAS)
  return img
```




```
1 speed = 28.105569
```




```
2 speed = 28.106527000000003
```


0
25
50
75
100
125
150
175
200
0 25 50 75 100 125 150 175 200
3 speed = 28.130404
0
25
50
75
100
125
150
175
200

5.3 Pipeline

```
model {
  ssd {
    num_classes: 2
    image_resizer {
      fixed_shape_resizer {
        height: 512
        width: 512
      }
    }
    feature_extractor {
      type: "ssd_resnet101_v1_fpn_keras"
      depth_multiplier: 1.0
      min_depth: 16
      conv_hyperparams {
        regularizer {
          l2_regularizer {
            weight: 0.00039999998989515007
          }
        }
        initializer {
          truncated_normal_initializer {
            mean: 0.0
            stddev: 0.029999999329447746
          }
        }
        activation: RELU_6
        batch_norm {
          decay: 0.996999979019165
          scale: true
          epsilon: 0.0010000000474974513
        }
      }
      override_base_feature_extractor_hyperparams: true
      fpn {
        min_level: 3
        max_level: 7
      }
    }
```

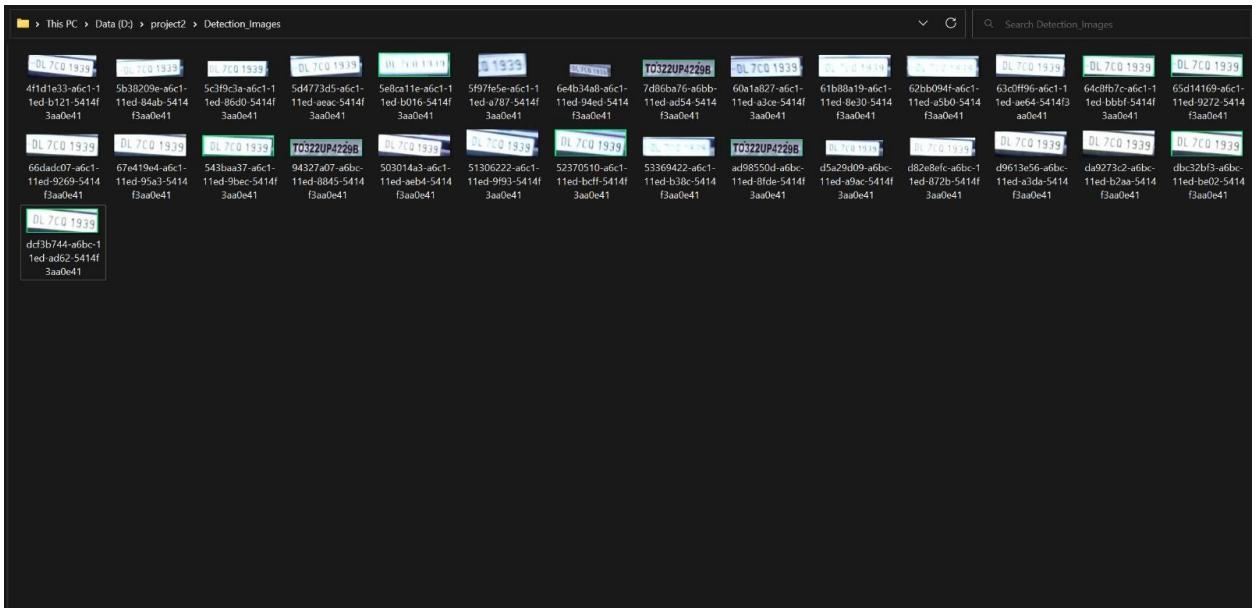
```
  }
  box_coder {
    faster_rcnn_box_coder {
      y_scale: 10.0
      x_scale: 10.0
      height_scale: 5.0
      width_scale: 5.0
    }
  }
  matcher {
    argmax_matcher {
      matched_threshold: 0.5
      unmatched_threshold: 0.5
      ignore_thresholds: false
      negatives_lower_than_unmatched: true
      force_match_for_each_row: true
      use_matmul_gather: true
    }
  }
  similarity_calculator {
    iou_similarity {
    }
  }
  box_predictor {
    weight_shared_convolutional_box_predictor {
      conv_hyperparams {
        regularizer {
          l2_regularizer {
            weight: 0.00039999998989515007
          }
        }
        initializer {
          random_normal_initializer {
            mean: 0.0
            stddev: 0.009999999776482582
          }
        }
```

```
        activation: RELU_6
        batch_norm {
          decay: 0.996999979019165
          scale: true
          epsilon: 0.0010000000474974513
        }
      }
      depth: 256
      num_layers_before_predictor: 4
      kernel_size: 3
      class_prediction_bias_init: -4.599999904632568
    }
  }
  anchor_generator {
    multiscale_anchor_generator {
      min_level: 3
      max_level: 7
      anchor_scale: 4.0
      aspect_ratios: 1.0
      aspect_ratios: 2.0
      aspect_ratios: 0.5
      scales_per_octave: 2
    }
  }
  post_processing {
    batch_non_max_suppression {
      score_threshold: 9.99999993922529e-09
      iou_threshold: 0.6000000238418579
      max_detections_per_class: 100
      max_total_detections: 100
      use_static_shapes: false
    }
    score_converter: SIGMOID
  }
  normalize_loss_by_num_matches: true
  loss {
    localization_loss {
```

```
112           weighted_smooth_l1 {
113           }
114         }
115         classification_loss {
116           weighted_sigmoid_focal {
117             gamma: 2.0
118             alpha: 0.25
119           }
120         }
121         classification_weight: 1.0
122         localization_weight: 1.0
123       }
124       encode_background_as_zeros: true
125       normalize_loc_loss_by_codesize: true
126       inplace_batchnorm_update: true
127       freeze_batchnorm: false
128     }
129   }
130   train_config {
131     batch_size: 1
132     data_augmentation_options {
133       random_horizontal_flip {
134       }
135     }
136     data_augmentation_options {
137       random_crop_image {
138         min_object_covered: 0.0
139         min_aspect_ratio: 0.75
140         max_aspect_ratio: 3.0
141         min_area: 0.75
142         max_area: 1.0
143         overlap_thresh: 0.0
144       }
145     }
146     sync_replicas: true
147     optimizer {
148       momentum_optimizer {
149         learning_rate {
```

```
          cosine_decay_learning_rate {
            learning_rate_base: 0.03999999910593033
            total_steps: 2000
            warmup_learning_rate: 0.013333000242710114
            warmup_steps: 2000
          }
        }
        momentum_optimizer_value: 0.8999999761581421
      }
      use_moving_average: false
    }
    fine_tune_checkpoint: "pre-trained-models/ssd_resnet101_v1_fpn_640x640_coco17_tpu-8/checkpoint/ckpt-0"
    num_steps: 2000
    startup_delay_steps: 0.0
    replicas_to_aggregate: 8
    max_number_of_boxes: 100
    unpad_groundtruth_tensors: false
    fine_tune_checkpoint_type: "detection"
    use_bfloat16: false
    fine_tune_checkpoint_version: V2
  }
  train_input_reader {
    label_map_path: "annotations/label_map.pbtxt"
    tf_record_input_reader {
      input_path: "annotations/train.record"
    }
  }
  eval_config {
    metrics_set: "coco_detection_metrics"
    use_moving_averages: false
  }
  eval_input_reader {
    label_map_path: "annotations/label_map.pbtxt"
    shuffle: false
    num_epochs: 1
    tf_record_input_reader {
      input_path: "annotations/test.record"
```

5.4 Database:



6. CONCLUSION & FUTURE SCOPE

6.1 Conclusion

One of the many crucial components of an Intelligent Traffic System (ITS) that may be performed using machine learning algorithms is speed estimation and license plate identification. Compared to traditional methods without image processing, including employing speed radar and manually inspecting license plates, it is more accurate and economical. This study shows that there is a relationship between speed estimation, camera angle, and ROI selection when using Euclidean distance. Later, license plate has been detected automatically using machine learning algorithms that gives its best accuracy, which is an interesting study for other researches that are discussing the same subject.In order for this study to serve as a guide for future investigations into how to estimate a moving vehicle's speed utilizing image processing, Euclidean distance, and license plate recognition.

6.2 Future Scope

The future work will involve recognizing the individual characters from the plate with other colors backgrounds with the standard templates issued by the government. This is mainly used in high traffic areas to control the traffic without the help of humans. It can be used for monitoring Parking Areas. In order to punish the one who violates the traffic rules such as crossing the speed limits, signal jumping and unauthorized vehicles.

REFERENCES

[1] Digital image processing by Rafacl C.Gonzalez and Richard E. Woods published by pearson education.

[2] R.Radha1 and C.P.Sumathi2, “A Novel approach to extract text from license plate of vehicle”, Signal & Image Processing: An International Journal (SIPIJ) Vol.3, No.4, August 2012

[3] Kumar Parasuraman and P.Vasantha Kumar, “An Efficient Method for Indian Vehicle License Plate Extraction and Character Segmentation”, IEEE International Conference on Computational Intelligence and Computing Research,2010.

[4] Lekhana G.C, R.Srikantaswamy ,“Real time license plate recognition system”, International Journal of Advanced Technology & Engineering Research (IJATER), National Conference on Emerging Trends in Technology (NCETTech) ISSN, Volume2,Issue4,ISSN No: 2250-3536, July 2012.

[5] Sarthak Babbar; Saommya Kesarwani; Navroz Dewanf; Kartik Shangle; Sanjeev Pate, “A New Approach for Vehicle Number Plate Detection”, 2018 Eleventh International Conference on Contemporary Computing (IC3)

[6] K. B. Sathya; V. Vaidehi; G. KavithaLiu, “Vehicle License Plate Recognition (VLPR),” 2017 Trends in Industrial Measurement and Automation (TIMA)

[7] Anumol Sasi ; Swapnil Sharma ; Alice N. Cheeran, “Automatic car number plate recognition” 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)

[8] Zusheng Zhang, Tiezhu Zhao, Xin Ao, and Huaqiang Yuan A Vehicle Speed Estimation Algorithm Based on DTW Approach IEEE SENSORS JOURNAL, VOL. 17, NO. 8, APRIL 15, 2017

[9] Mahmoud Famouri, Zohreh Azimifar, Member, IEEE, and Alexander Wong , Senior Member, IEEE A Novel Motion Plane-Based Approach to Vehicle Speed Estimation IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS

[10] Cheng-Hung Lin; Yong-Sin Lin; Wei-Chen Lu , An efficient license plate recognition system using convolution neural networks- 2018 IEEE International Conference on Applied System Invention (ICASI).