International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-7, Issue-6, March 2019

Flag Semaphore Detection using Tensorflow and Opencv

Athul Motty, Yogitha A, R Nandakumar

Abstract*: This paper studies the automatic recognition of Flag Semaphores. We consider both static semaphores wherein the flags are held by the signaler in fixed positions and also dynamic signaling with flags (used internationally for aircraft marshalling and also by mariners). Reading Static semaphores such as those used by mariners are our main focus. We suggest the use of image processing and machine learning techniques to recognize and detect the flags and the signaler. OpenCV technology was used to capture the images and the TensorFlow API to detect the static semaphores. We could achieve promising results in the detection of static flag semaphores - a confidence level of 99%. We conclude that for deciphering signals where the flags are in motion, more sophisticated machine learning methods would be needed.

***Index Terms*: Image Processing, Flag Semaphore signals, Machine learning, Tensorflow API, OpenCV**

I. INTRODUCTION

Image processing is concerned with converting an image into digital form and performing some operations on it, in order to extract some useful information from it [2]. Image processing comprises three general phases, that all types of data have to undergo: a) Pre- processing b) enhancement and display c) information extraction.

Semaphores are a means of communication using gestures to convey some useful information from a distance. Flag semaphores are a purely visual communication system for giving information a distance employing signals made with hand-held flags [8]. The flag positions and orientation encode the information. In the flag semaphore system currently used all over the world, the signaller holds two short poles with square flags in different fixed positions to indicate letters of the Roman alphabet. In each hand the signaller holds one pole; and each arm is extended in one of eight possible directions as shown in figure 1 [19]. Except for the rest position, the flags do not overlap. For maritime and land-based signalling, flags of different colours are used. In what follows, we refer to this system as consisting of static semaphores.

Revised Manuscript Received on 30 March 2019.
***** Correspondence Author
Athul Motty, Department of Computer Science and IT, Amrita School of Arts & Sciences, Kochi, Amrita Vishwa Vidyapeetham, India.
Yogitha A, Department of Computer Science and IT, Amrita School of Arts & Sciences, Kochi, Amrita Vishwa Vidyapeetham, India.
R Nandakumar, Department of Computer Science and IT, Amrita School of Arts & Sciences, Kochi, Amrita Vishwa Vidyapeetham, India.

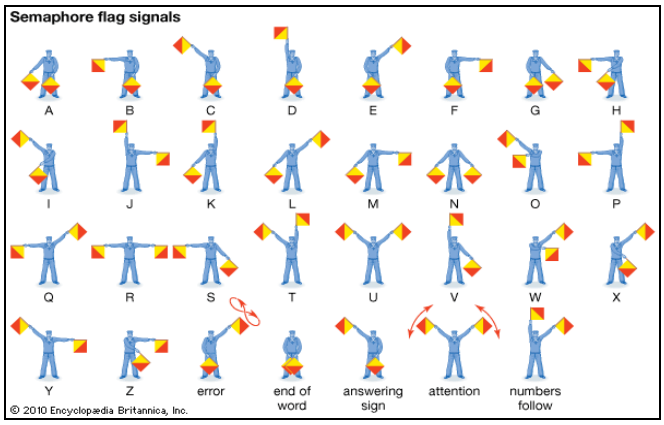

Fig.1. 31 variations of semaphore signals

More sophisticated signals using flags and batons are employed by aircraft marshals and mariners. Aircraft Marshalling is part of aircraft ground handling at airports. The marshal signals the pilot to keep turning or slow down or stop or shut down engines, thus leading the aircraft to its parking stand or the runway. Most of these signals involve movements of two orange batons. We call such signals, dynamic semaphores. A mariner on one ship communicates with his counterpart on another ship using an international signalling system that uses two flags – for some signals, the flags are moved but in others, they are held in fixed positions.

In this paper, we concentrate the static semaphores. We first describe conceptually our approach towards detecting them and the tools that we employ - the OpenCV technology and TensorFlow API.

II. STATIC SEMAPHORES DETECTION – CONCEPT

In the system of signalling with static semaphores, as shown in figure 1, there are only finitely many allowed positions for each flag, so there are only a limited number of signals that can be made. And all these signals may be detected by calculating the angle between the arms holding the flags. For this angle calculation, we need a mechanism to visually spot the flags and the signaller and their relative positions. Calculating the angles between oriented features of various objects is a well-studied area - for example, in decoding hand gestures, the angles between the fingers of a hand need to be calculated with some accuracy [3].However, on closer examination of the static semaphore system (figure 1), we see that we don't need to deploy this angle detection functionality owing to the possible orientations of the flag holding hands being at least 45 degrees apart.

Indeed, if we consider only the right hand of the signaller,

a) *If the flag position has only an x coordinate* (with Y coordinate nearly zero), it is taken to be necessarily horizontal. If the flag has only a positive Y coordinate and nearly zero X coordinate, it is taken as pointing up.
b) *If the flag has both x and y coordinates positive and nearly equal*, it is taken to be 45 degrees above horizontal and
c) *If x is positive and y negative and both have nearly equal magnitude*, it is taken to point at a direction 45 degrees below horizontal.

So we infer that only an approximate measurement of the flag positions with respect to the signaller is sufficient to decipher the precise signal being made. We don't need accurate determination of the angles.

III. TOOLS - TENSORFLOW AND OPENCV

The primary concern of Machine learning (ML) is the development of algorithms and statistical models that enable computer systems to successively improve their performance on a specific task. Machine learning algorithms construct a mathematical model of sample data, known as "training data", and they then employ this model to arrive at decisions or predictions without it being explicitly programmed to perform the task [6]. Deep learning (which also goes by the names of 'deep structured learning' and 'hierarchical learning') is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. There are various categories of Learning - supervised, semi-supervised or unsupervised [7]. Among all machine learning algorithms, Neural networks are, at present, the best - they outperform all other families of machine learning algorithms. Neural networks can efficiently learn from the datasets and create models of the data. A convolutional neural network (CNN) is the type of network most commonly used for image classification whereas an R-CNN (R for 'region'), is the network of choice for object detection. In our study, we use TensorFlow (TF), a framework for deep learning to recognize flag semaphore. TensorFlow gives us an open source API set that has functionality for the detection of multiple objects featured in real-time video streams. Figure-2 shows the overall architecture of how an image is being recognized using TensorFlow. We use Faster-RCNN-Inception-V2 model, a predefined model offered by TensorFlow as the base. To improve the accuracy and the range of objects that can be detected, the model can be fine-tuned. It can also be trained to recognize any custom object.

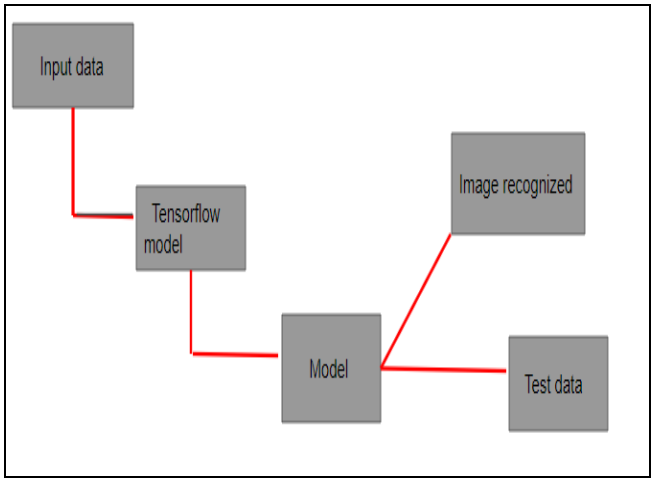

Fig.2. Tensorflow – overview

Once the system has been trained to detect objects, it is further trained to track the object as long as the latter stays within the range of the camera. Our implementation focuses on recognizing the semaphore flag signals using the TensorFlow API - signals shown by the marshal are captured by webcam installed on the flight and interpreted.

OpenCV is an image and video processing library; it has bindings in C, C++, Java and Python. Facial recognition and detection, optical character recognition, advanced robotic vision, license plate reading, photo editing - in all these domains of image and video analysis, OpenCV finds use. In our implementation, we deploy OpenCV with python binding to capture images from a video input received from our optical device.

IV. DETECTION OF STATIC SEMAPHORES-PROCESS

Our system captures image signals, translates and classifies them into a form which the receiver can understand. We use TensorFlow API + deep neural network (DNN) module in OpenCV and train the model for detecting the semaphore gestures. To do this, we need the Images of semaphore gestures, matching TF Records for the training and testing data, and then we need to set up the configuration of the model, then we can train. Once the training is done, Tensorflow can map each semaphore signal with those it has already learned. Figure 3 shows the detailed workflow of generating a static semaphore detection model. The system also overcomes the limitations faced by the traditional methods by detecting multiple semaphores at once. The receiver can use this system to monitor semaphore signals shown from the sender to quickly make any decisions or actions.

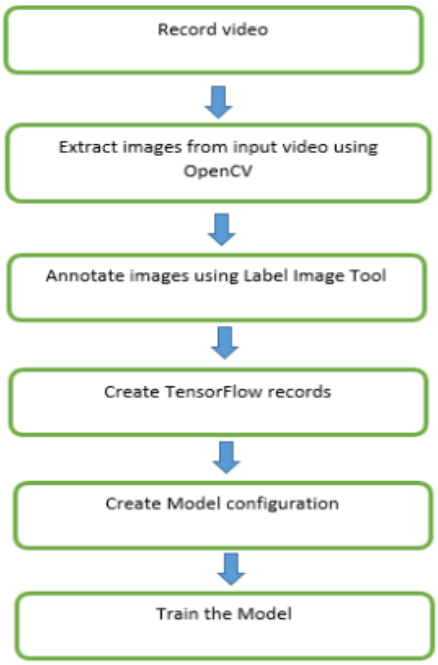

Fig.3. Steps involved in generating static semaphore detection model.

The goal of our implementation is to train our neural network object detection classifier for recognizing flag semaphore signals and the output is shown by drawing boxes around the input images recognized as representing semaphores.

System requirements:

- Linux OS (4GB RAM, Intel 3rd generation processor)

Further requirements:

- Python3 installation
- Install TensorFlow- TensorFlow the open source API used for recognizing and matching flag semaphore. TensorFlow reads input in 3 different formats such as a) images b) video c) live recording. In our experiment we use images of flag semaphores as the input.
- Pip installation- Pipenv is a dependency manager for Python projects. Pipenv enables users and developers of applications to easily set up a working environment.
- Virtualenv installation- Virtualenv is a tool to create an isolated Python environment [17].
- Tensorflow model- Faster-RCNN-Inception-V2-COCO model. TensorFlow provides several pre-trained classifiers with specific neural network architecture models. Some of these models (for example, the SSD-MobileNet model) has faster detection but accuracy is compromised. On the other hand, some models (such as the Faster-RCNN model) give slower detection but enhanced accuracy. In our implementation we use Faster RCNN model which can be deployed on a decent powered laptop or desktop. We perform training for the detector on the Faster-RCNN-Inception-V2 model, and it is found that the detection showed considerable improvement, albeit at a noticeably slower speed [15].
- labelImg tool- python program is used for annotating the semaphore images collected online.
- XML to CSV conversion program
- Python Dependency packages (pillow, lxml, cython, jupyter, matplotlib, pandas, opencv-python)

V. STATIC SEMAPHORES DETECTION: SUMMARY OF OUT EXPERIMENTS

We collected images of semaphores from google. The training database was then generated by manually tagging the objects (semaphores) in the images using Label Img (Fig 4). LabelImg saves the annotations as .XML files in a PASCAL VOC format. A ready-made script is available for creating TFRecords (TensorFlow record format). All the images collected are of the dimensions below 640 X 480 to increase model training efficiency. The training directory consisted of a total of 130 pictures along with its xml annotations where in 80 percent is contained in train folder and the remaining 20 per cent in the test folder. The .XML file is used to generate the TFRecord which is the input to the TensorFlow trainer. TFRecord is being served as the input to the TensorFlow training model. The .XML data is used to create a CSV file that contains all the data for training and testing.

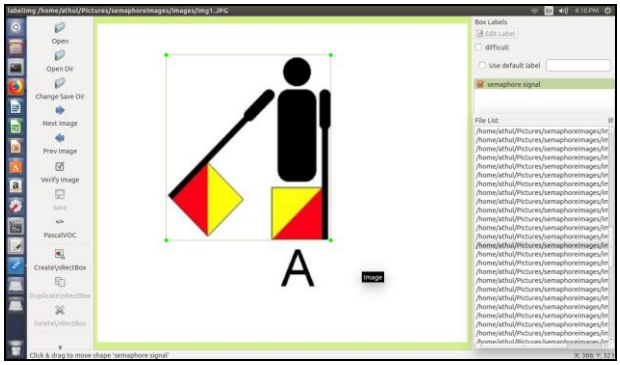

Fig.4. LabelImg tool for annotating the images

Creating LabelMap- Before training begins, it is necessary to create a LabelMap that helps the trainer to understand what each object is by defining a mapping from class name to class ID. We need to create a label.pbtxt file that is used to convert the label name to a numeric id. In our experiment, it is:

item {
id: 1
name: 'semaphore signal'
}

Once the TFR datasets are created, we can either use an existing model and fine tune it or build from scratch. In this work we have tested on an existing model (faster_rcnn_inception_v2_coco_2018_01_28, Base model of an object detector trained on different datasets) we reuse some of its parameters to initialize our new model. since most of the features that are learnt by CNNs are often object-agnostic and fine tuning an existing model is usually an easy and accurate process [15].

Training:

Within the supervised learning paradigm, a machine learning algorithm puts together a model by considering many examples and attempting to find a model that minimizes loss. Loss is a parameter that indicates a penalty for a bad prediction; it is a number quantifying how off the mark the model's prediction was on a single example [18]. Loss has the value 0 if the model's prediction is perfect; else, the loss has a positive value. The intent of training a model is to find a set of weights and biases that lead to low loss, averaged over all examples. The training routine periodically saves checkpoints (a checkpoint is a Binary file that saves values of the weights, biases, gradients and all the other variables). In our implementation, checkpoints typically were saved every five minutes. We waited until just after a checkpoint has been saved to terminate the training. We can terminate training and start it later, and it will restart from the last saved checkpoint. The checkpoint at the highest number of steps will be used to generate the frozen inference graph. The frozen inference graph is normally loaded for inference, in a TensorFlow Python API session during which weights from the checkpoint files are inserted into the Variable in the graph. To keep all the values of the variables and the Graph in a single file, we freeze the graphs. The training process was carried out for a period of 3 to 4 hours. Figure 5 shows how training is progressing along with its loss. We terminate the training process when the loss is consistently below 0.05.

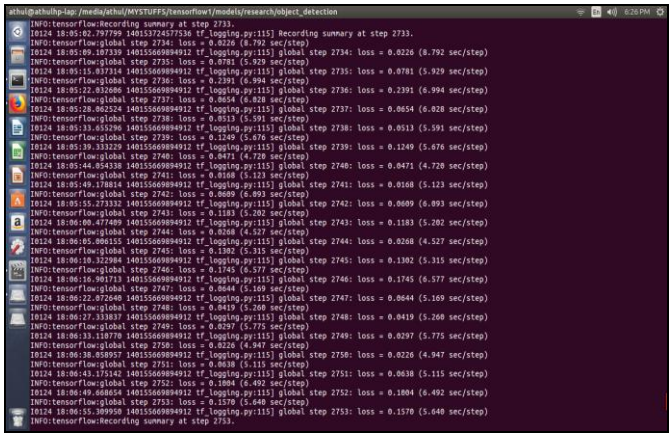

Fig.5. Training progress

The TensorBoard page provides information and graphs that show how the training is progressing. One important graph is the Loss graph, in which the overall loss of the image classifier has been shown against training time on the X axis. See figure 6.

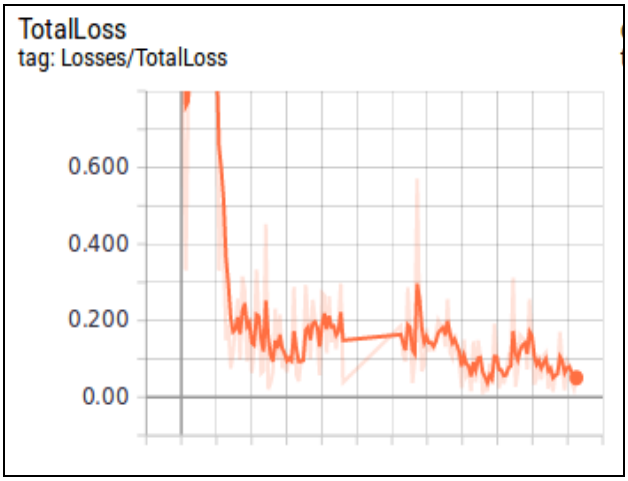

Fig.6. Total loss occurred while training (we ended at the checkpoint at step 2814; the last loss=0.041)

Display of Test Results:

To test the model, we select a model checkpoint (usually the latest) and export into a frozen inference graph. The following figure shows that a typical semaphore image could be recognized with a confidence of 99%.

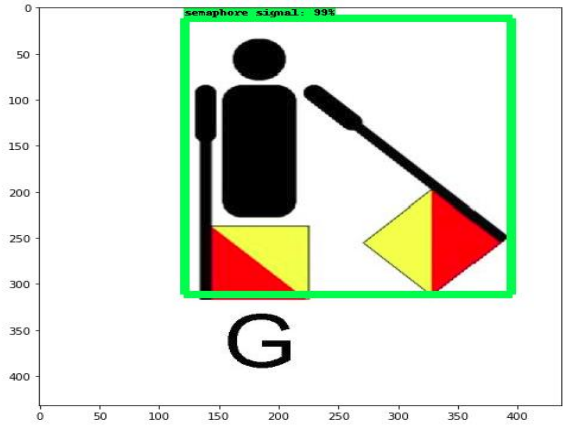

Fig.7. Semaphore Image is recognized with a confidence level of 99%

VI. CONCLUSION

We have proposed a system that attempts to automate the recognition of static flag semaphore signals. We used TensorFlow Object Detection API, an open source framework for tasks pertaining to object detection, for training and testing an R-CNN object detection model with a view to detecting semaphore flag signals. We understand that decrypting the very limited system of static flag semaphores needs only very limited image processing functionality - and not even angle detection functionality - because of there being only finitely many and discretely separated positions of the flags. For dynamic semaphores, we observe that there is a need for greater sophistication and we suggest a more evolved use of the Tensorflow machine learning API system.

REFERENCE

1. Fuad, M (2015). Semaphore gesture detection using Kinect sensor. Seminar Nasional Aplikasi Dan Pengembangan Teknologi Informasi.
2. https://en.wikipedia.org/wiki/Digital_image_processing
3. S. P. More and A. Sattar, "Hand gesture recognition system using image processing," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, 2016, pp. 671-675J. Wang, "Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style—Submitted for publication)," *IEEE J. Quantum Electron.*, submitted for publication.
4. https://www.tensorflow.org/tutorials/images/image_recognition
5. P. R. V. Chowdary, M. N. Babu, T. V. Subbareddy, B. M. Reddy and V. Elamaran, "Image processing algorithms for gesture recognition using MATLAB," 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, Ramanathapuram, 2014, pp. 1511-1514.
6. https://en.wikipedia.org/wiki/Machine_learning
7. https://en.wikipedia.org/wiki/Deep_learning
8. https://en.wikipedia.org/wiki/Flag_semaphore
9. https://www.cs.ubbcluj.ro/~gabis/ml/ml-books/McGrawHill%20-%20Machine%20Learning%20-Tom%20Mitchell.pdf
10. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html
11. https://towardsdatascience.com/building-a-toy-detector-with-tensorflow-object-detection-api-6 3c0fdf2ac95 - Priya Dwivedi
12. Object detection in sports: TensorFlow Object Detection API case study Bachelor's Thesis Degree Programme in Mathematical Sciences January 2018-Pirkko Mustamo`
13. Q. Zhao, Y. Li, N. Yang, Y. Yang and M. Zhu, "A convolutional neural network approach for semaphore flag signaling recognition," 2016 IEEE International Conference on Signal and Image Processing (ICSIP), Beijing, 2016, pp. 466-470.
14. R. Phadnis, J. Mishra and S. Bendale, "Objects Talk - Object Detection and Pattern Tracking Using TensorFlow," 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, 2018, pp. 1216-1219
15. https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10
16. S. Hayat, S. Kun, Z. Tengtao, Y. Yu, T. Tu and Y. Du, "A Deep Learning Framework Using Convolutional Neural Network for Multi-Class Object Recognition," 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), Chongqing, 2018, pp. 194-198
17. https://docs.python-guide.org/dev/virtualenvs/
18. https://www.tensorflow.org/guide/summaries_and_tensorboard
19. https://www.google.co.in/search?q=flag+semaphores+images&oq=flag+semaphores+images&aqs=chrome..69i57j0l3.6863j0j4&client=ms-android-oppo&sourceid=chrome-mobile&ie=UTF-8#imgrc=sBr9X05sndaf_M:

AUTHORS PROFILE

Athul Motty PG Student, Master of Computer Application, Department of Computer Science & IT, Amrita School of Arts & Sciences, Kochi, Amrita Vishwa Vidyapeetham, India.

Yogitha A PG Student, Master of Computer Application, Department of Computer Science & IT, Amrita School of Arts & Sciences, Kochi, Amrita Vishwa Vidyapeetham, India.

R Nandakumar Assistant Professor, M.Tech.(CS), M.Sc. (Physics), Department of Computer Science & IT, Amrita School of Arts & Sciences, Kochi, Amrita Vishwa Vidyapeetham, India.