DETECTION AND TRACKING IN AERIAL VIDEOS

PROJECT REPORT

submitted by

AJOY BRITTO REX (TVE16CS006)

AKSHAY SEBASTIAN (TVE16CS008)

HARISANKAR R (TVE16CS032)

MATHEW GEORGE (TVE16CS040)

to

the APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology

in

Computer Science and Engineering

Department of Computer Science and Engineering

College of Engineering, Trivandrum

Kerala

April 7, 2022

DECLARATION

We undersigned hereby declare that the project report **Detection and Tracking in Aerial Videos** , submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of **Dr. Saritha R**. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Trivandrum

Date: April 7, 2022

Ajoy Britto Rex

Akshay Sebastain

Harisankar R

Mathew George

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING, TRIVANDRUM

CERTIFICATE

This is to certify that the report entitled **"DETECTION AND TRACKING IN AERIAL VIDEOS ",** submitted by **Ajoy Britto Rex**, **Akshay Sebastain**, **Harisankar R**, **Mathew George** to the **APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY** in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the project presented by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Dr. Saritha R	**Salim A**
Assistant Professor	Head of Department
Department of CSE	Department of CSE
(Guide and Coordinator)	(Head of Department)

ACKNOWLEDGEMENT

It is with extreme respect that we remember the names of all who had been a great help and guidance throughout our sessions. Firstly, we would like to thank the Almighty for giving me the wisdom and grace for completing the project. With a profound sense of gratitude, we would like to express our heartfelt thanks to my guide, **Dr. Saritha R**, Assistant Professor, Department of Computer Science and Engineering for her expert guidance, co-operation and immense encouragement in pursuing this project. We are extending our gratitude to the project coordinator, **Piyoosh P**, Assistant Professor, Department of Computer Science and Engineering for their co-operation and support. We are very much thankful to **Dr. Salim A**, Head of the Department of Computer Science and Engineering, for providing necessary facilities and his sincere co-operation. Our sincere thanks is extended to all the teachers of the Department of CSE and to all our friends for their help and support.

Ajoy Britto Rex
Akshay Sebastian
Harisankar R
Mathew George

ABSTRACT

Cameras make ideal sensors for drones as they are lightweight, power-efficient and an enormously rich source of information about the environment in numerous applications. Although lots of information can be derived from camera images using the newest computer vision algorithms, the use of them on-board of UAVs poses unique challenges. Their computational complexity often conflicts with the need for real-time operation and the extreme resource limitations of the platform.

The detection of vehicles driving on busy urban streets in videos acquired by airborne cameras is challenging due to the large distance between cameras and vehicles, simultaneous vehicle and camera motion, shadows, or low contrast due to weak illumination. However, it is an important processing step for applications such as automatic traffic monitoring, detection of abnormal behaviour, border protection, or surveillance of restricted areas.

In our project we have built a system capable of detecting and tracking objects in drone video footage which is optimized to run on the on board computer of the UAV and maximizing the detection and tracking accuracy.

Contents

List of Figures

List of Tables

ABBREVIATIONS

(List in the alphabetical order)

FPS Frames per second

IoU Intersection over Union

mAP mean Average Precision

NMS Non Max Suppresion

SORT Simple Online and Realtime Tracking

Chapter 1

Introduction

Unmanned aerial vehicles (UAVs) played a vital role in modern wars and industries as technology developed. Moving object detection in aerial video is essential for UAV intelligence as the foundation for higher targets, such as tracking and object recognition. Unlike applications with fixed cameras, such as traffic control and building surveillance, aerial surveillance has the benefits of improved mobility and broader scope of surveillance. Meanwhile, aerial video presents more challenges, such as changing background and low resolution. Much attention has therefore been paid to the moving detection of objects in aerial video.

Numerous UAV studies have attempted to detect and track those Material forms such as cars, people including Moving pedestrians, and autonomous landmarks Real-time Navigation and Landing. However there are only a handful who consider multiple object detection. Given the fact multiple target objects identification is relevant for a lot of UAV applications. In our view, the main reasons for this gap between application needs and technical capabilities are due to two practical but critical limitations: (1) object recognition algorithms often need to be hand-tuned to particular object and context types; (2) it is difficult to build and store a variety of target object models, especially when the objects are diverse in appearance, and (3) real-time object detection demands high computing power even to detect single objects, much less when many target objects are involved.

Detection of objects can be divided into static object detection and moving object detection, in which the detection of static objects is achieved through image segmentation and colour, gray value, edge and texture-based clustering and moving object detection, using

the continuity of moving information to partition areas in images and associating similar moving areas in target objects. Detection of moving objects can be divided into three categories including methods for optical flow, different methods for the background and different methods for frames.

Through our project we propose to build a system capable of detecting objects in drone video footage which is optimized to run on the on board computer of the UAV and maximizing the detection and tracking accuracy.

Chapter 2

Existing Methods

This section provides an analytical review of the existing techniques of object detection and tracking. By this literature study we hope to find and systematically review these state-of-the-art object detection and tracking techniques in a structured and scientifically valid way.

2.1 DETECTION

There are several detection methods existing. Each of them having different methods for region proposal and classification. Accordingly there are mainly two classification. One, which does region proposal and classification in two stages and the other which combines both of them and does detection in a single pass of the input image.

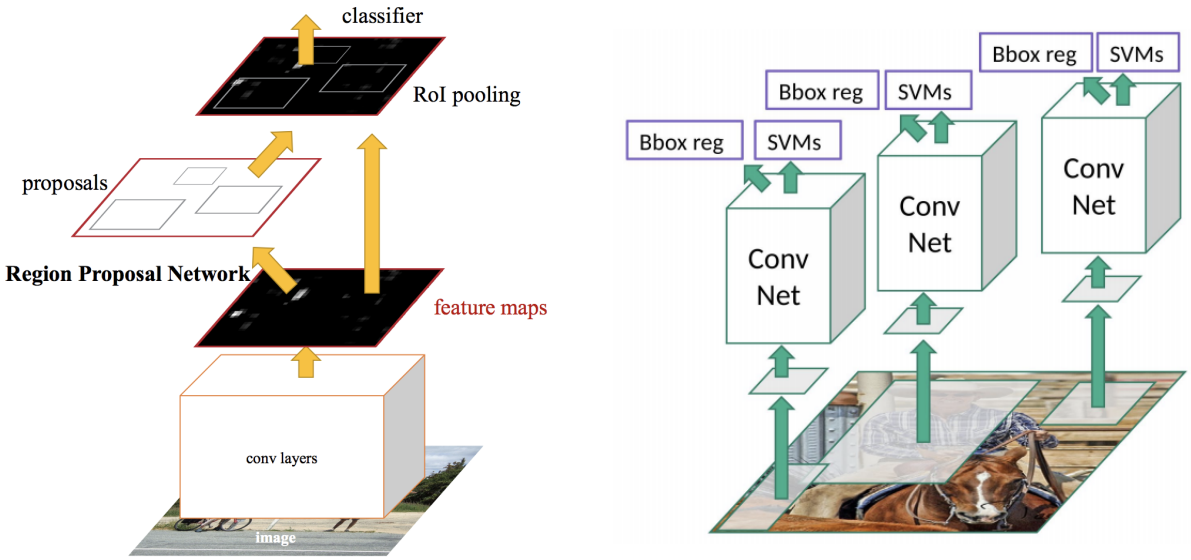
2.1.1 Two stage Detectors

The two stage Detectors in the first stage uses some method to extract regions of possible objects from the input. The extracted regions, in the next stage, are fed to the classification network, generally a fully connected layer to produce the class of the object.

Region proposal can be done in a variety of methods including sliding window, Selective search using segmentation, region proposal networks etc. The stage gives possible sections in the image where there is a chance of occurrence of object.

The output of the region proposal stage will contain regions in image where an object may be present. These regions are then fed to the classifier which is a fully connected layer.

The classifier gives the probability of the region belonging to each of the predefined class. The class with the highest probability will be assigned as the detection for the region.
The most widely used two stage detectors are R-CNN, Fast R-CNN, Faster R-CNN. Each differ in the way the produce the region proposals.



(a) Faster-RCNN

(b) Fast RCNN

Figure 2.1: One Stage Detectors

2.1.2 One stage Detectors

One stage detectors produce detections with only one pass of the input image. Object localization and identification are done by a single convolutional network. A general design is to use the earlier stages of the network for region proposal and the later stages, usually fully connected layers for region classification.

Since detection only include a single network, one stage detectors are more likely to be used in real time applications. Popular examples of this class of detectors are YOLO, single-shot detectors etc.

2.2 TRACKING

Tracking involves labeling objects in a frame and keeping track of the labelled objects in the subsequent frames. There are tracking methods either using deep neural network or traditional image properties or both.

2.2.1 Optical Flow

The objects are tracked using the spatio-temporal image brightness variations at pixel level. A displacement vector for the object to be tracked is obtained based on the:

- Brightness Consistency - The brightness around a small region is assumed to be nearly constant.
- Spatial Coherence - Neighbouring points generally belong to the same surface and hence have similar motions.
- Temporal Persistence - Motion of a patch has a gradual change
- Limited motion - Points do not move very far or in a haphazard manner.

Once these conditions are satisfied, the velocity of certain pixels are modelled. Tracking is carried out using this velocity model and some prediction techniques [10].

2.2.2 Kalman filter

Kalman filter uses detections from $(t-1)^{th}$ frame to predict the positions of the tracks in the t^{th} frame. Kalman filter assumes a constant velocity model and later weighs in the noises associated with the velocity model and in the detections based on available measurement. Kalman filter works recursively by making predictions and then updating the predictions based on the noise measurement [11].

2.2.3 Deep Regression Networks

This is deep neural network based tracking method. A model is trained on a dataset consisting of videos with labelled target frames. The model tracks a given object from the given image

crop. A two-frame CNN architecture which uses both the current and previous frame to regress on the object is used. The crop from the previous frame is taken based on the predictions and define a search region in the current frame based on that crop. The network is trained to regress for the object in this search region. The network architecture is simple with CNN's followed by Fully connected layers that directly give us the bounding box coordinates [10].

Chapter 3

Detection and Tracking in Aerial videos

It has already been mentioned in the previous section that Detection and Tracking are subsisted by various methods or processes. What we intend to propose here is a system capable of detecting drone images via on-board processing subject to certain constraints or requirements consecutively catering to various needs or applications that can range from traffic surveillance to monitoring and helping people from calamity affected areas.

3.1 Problem Definition

With on-board camera and low-power embedded system, we aim to implement deep learning algorithms for the drones to perform real-time object detection and tracking. We propose a handshake mechanism between object detection and object tracking that takes advantage of the run-time algorithm for object tracking and the accuracy of the algorithm for object detection.

3.2 Aim

The aim of this project is to:-

- Detect the various classes with high accuracy as per the requirements.
- Cater to various applications such as automatic surveillance, traffic monitoring etc.
- To foster a generation of commercial or military drones that are expected to be aware of surrounding objects while flying autonomously in different terrains and conditions.

Chapter 4

Design and Implementation

The project aims to perform object detection in drone images on the on-board computer of the drone at a minimum of 5 FPS. Thus the computation power available is limited. We propose a design which aims at maximizing the accuracy subject to the limited hardware constraint, at the same time achieving the real time 5 FPS requirement.

4.1 DATASET

To train our detector we used the following two datasets:

4.1.1 VisDrone

This is the primary dataset used. It consists of high definition images of different image sizes, all captured using UAV, with annotations of 10 classes which include pedestrians, people, car, bus etc. As the dataset is captured over urban as well as suburban areas of 14 different cities across China from north to south, the dataset comprises a large diversity of different scenarios. The dataset was available with a partitioning of images in the ratio of approximately 12:1 for training and validation purposes [8].

4.1.2 UAVision

This dataset also consists of sequences captured from an aerial viewpoint. The image quality ranges from 720p to 4K captured at a frame rate of 30fps. It contains a wide variety of scenarios

like urban,landscape, road, buildings with targets including persons, cars, trucks etc. under various illumination conditions.

Many of the objects in these two datasets do not satisfy the DORI criteria for object size for successful detection and further processing is needed.

4.2 STATIC TILING

The images captured using a UAV differ from normal images used for training and detection in mainly two aspects.

Image quality: The images used in normal detectors range from 256x256 pixels (or less) to 720p images whereas UAV uses high definition cameras resulting in high definition images ranging from 720p to 4K images.

Object size: The objects in normal detection images are large in the sense that the ratio of the object size compared to the image size is significantly high, making detection of these objects easier. But the objects in the UAV captured images do not satisfy this property. The Detection, Observation, Recognition, and Identification (DORI) criteria requires that objects should have a minimum of 10% pixel height compared to that of the image for detections. This criteria is not met for many of the objects in the dataset images.

Normally every detector resizes the input image before passing to the detection model. Since UAV captured images consist of many very small objects, downscaling of these images will result in losing the object information required for detection. This will result in great drop in accuracy[3]. In order to alleviate small object problems, we decrease the effect of image down-sampling. Static tiling is employed for this and is used during both training and inference phases[1]. Tiling is that we crop the image into a number of smaller images. For the detection phase, we are using 2x2 tiling. If the size of the input image is above a predefined threshold then the image is divided into four smaller images. Each smaller image shares their boundary with the adjacent smaller image thus we have overlapping regions between two images. Overlapping is included to ensure that the objects that are truncated by the boundary of a cropped segment are not lost. These cropped image segments are fed into the detection network. This will result in a higher ratio of object size to image size during downscaling

leading to reduced drop in accuracy [1].

After detection of objects in all the image segments, they are combined together to obtain the detections in the original image. The detection box coordinates are adjusted such as to form valid detection in the original image. Since there is an overlap between adjacent segments, objects in the original image that are present in this common region are detected multiple times. Non-maximum suppression is used to select one of the many detections of the same object based on the detection score. Static tiling is shown in Figure 4.3

This process increases the number of objects detected in the images. The following figure shows the result of study on the effect of tiling. As shown in the graph, objects have be enlarged which increases the chances of them being detected.

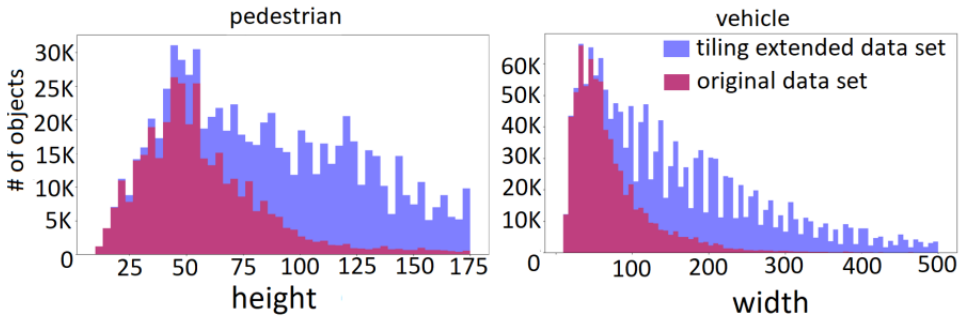


Figure 4.1: Height and width histograms of objects for original and tile extended data set

4.3 YOLOv3

The performance of object detectors are measured by its Frames per second (FPS) which is the number of frames it can process in a second and mean Average Precision (mAP) at different Intersection over Union (IoU) thresholds. This is the accuracy of the predicted bounding box and classification. There is always a trade off between FPS and mAP. More FPS leads to reduced precision and vice-versa.

The comparison of FPS, mAP trade off of various detectors in the trained with the COCO dataset are given in the following table [7].

Model	Test set	mAP	FPS
SSD300	COCO test-dev	41.2	46
SSD500	COCO test-dev	46.5	19
YOLOv2 608x608	COCO test-dev	48.1	40
Tiny YOLO	COCO test-dev	23.7	244
SSD513	COCO test-dev	50.4	8
FPN FRCN	COCO test-dev	59.1	6
Retinanet-101-500	COCO test-dev	53.1	11
Retinanet-101-800	COCO test-dev	57.5	5
YOLOv3-320	COCO test-dev	51.5	45
YOLOv3-416	COCO test-dev	55.3	35
YOLOv3-608	COCO test-dev	57.9	20
YOLOv3-tiny	COCO test-dev	33.1	220

Table 4.1: Comparison of various Object Detection Models

The static tiling preprocessing splits one image into 4 pieces, i.e, in effect the 5 FPS requirement changes to 20 FPS and we need a model with minimum 20 FPS and a high mAP measure. Also considering other overheads like crop area detections, cropping, post processing to aggregate results etc, we chose the YOLOv3-416 model as our object detection model. The YOLOv3 is a single stage object detection model. YOLOv3 uses a variant of Darknet with 53 layers as backbone. For detection, 53 more layers are stacked, giving a 106 layer fully convolutional underlying architecture.

4.3.1 Architecture

YOLO works by partitioning the image into grids and associating anchor boxes to each gird. For every anchor box-grid pair, a classification is carried out. If an object exists in that pair, that is output as a detection. To improve object detection, the architecture includes residual skip connections and upsampling. The most important feature of this model is that it makes detections at three different scales. YOLO is a fully convolutional network and its outputs are generated by applying a 1x1 filter on a feature map. In the v3 version of YOLO, the detections are obtained by applying the 1x1 detection filter on feature map of three different sizes at three different places(layer 79, layer 91 and layer 106) in the network [3][4].

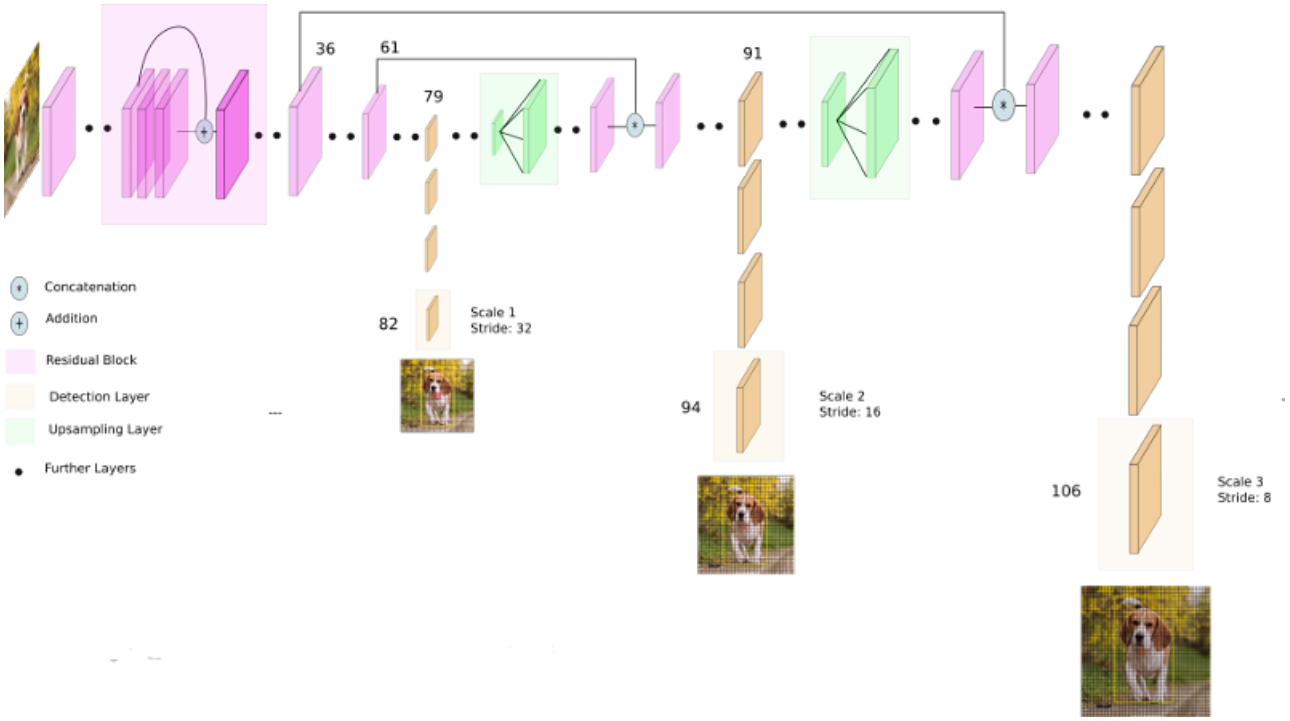


Figure 4.2: YOLOv3 Architecture

The detections at different layers aims to identify objects of different sizes. This is aided by the fact that each of these layers use different anchor boxes to help detect object of that particular size. The initial detections at layer 79 identifies large sized objects from the feature map available at that stage. After some convolutions and concatenation with previous layers feature maps, a detection is done at layer 91 which aims to find objects of medium size. The process is repeated again and a final detection is done at layer 106 to identify the smaller objects.The upsampled layers concatenated with the previous layers help preserve fine grained features which help in detecting small objects [3][4].

Since our dataset contains drone images, most of the objects present will be very small, due to the wide perspective from which the images are taken. YOLOv3, inherently in its architecture, contains features for small object detection which is an added advantage.

Unlike previous versions of YOLO, YOLOv3 uses logistic regression to predict output classes than a softmax layer. Softmax only allows an anchor box in one grid to be associated to one of the class whereas the logistic regression approach associates every anchor box, cell pair to be associated to classes for which the logistic regression unit outputs one. Thus, for

example if an anchor box within a cell detects a woman, that detection can be associated to both the class women and person. To allow for this change, YOLOv3 has also changed the loss function from a squared error loss to a cross entropy loss.

4.4 TRAINING

A pre trained YoloV3 model was trained using our dataset for better detection of the aerial objects. The model was trained to detect the 10 classes mentioned in the VisDrone dataset.

Objects marked to be ignored were masked in the images to avoid false positives during the training. Objects with higher truncation and occlusion were also masked.

The input images of the training dataset are high definition images. If the model input size is set to match the image size of the training dataset, then the time for training will increase drastically. Hence increasing the model input size is not an acceptable solution. But having low model input size will result in reduced detection accuracy. Hence the model input size was set to 608x608. Further small randomization of model input size was allowed during training to increase the accuracy.

Later the input images were tiled to form smaller images. The reason is that:

Since the model input size is set as 608x608, downscaling of images occurs before the image is passed for training. If the model is thus trained, objects with very low aspect ratio will not be detectable for the model. This will result in reduced accuracy. Further, this may lead to errors during training since areas specified by the annotations might not give any meaning to the model.

Hence the images in the training set were tiled before training the model. All images were cropped into segments of size approximately 608x608. Depending on the image size, the image may be sliced into 4 or 8 segments. The annotations of the original image were changed to match the coordinate system of the smaller images. These smaller images form the new training set for the model. This process increased the ratio of object size to image size resulting in better training of the model [1].

The model was trained using the new training set. Once the change in loss started to decrease, mAP was checked at intervals of 500 iterations. The weights providing better

accuracy was chosen for the detection model.

4.5 DETECTION

The input images are subjected to the static tiling preprocessing where there are cropped into predefined number of slices with predefined height and width overlap. These parameter, i.e, the number of slices, height offset and width offset may be considered as hyperparameters and can be varied based on available hardware, model input size etc.

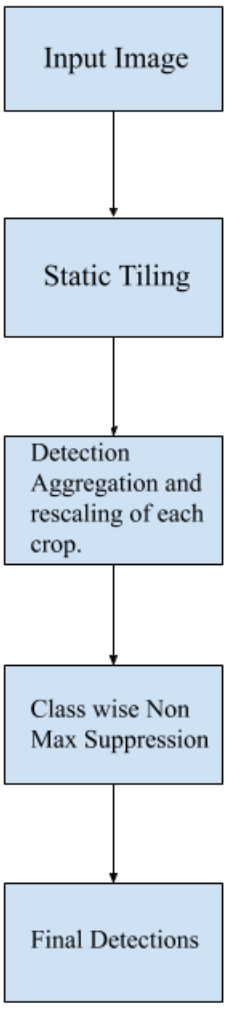


Figure 4.3: Flowchart: Detection pipeline

After tiling each cropped piece is batched together into an array of dimensions *[number of crops, crop height, crop width, color channel]* and fed to the custom trained YOLOv3 model for object detections. The detector returns a list of dimension *[number of crops]* which contains detections of each crop. The elements of the output list are dictionaries with key as class id's of detected objects and values as a list where each element in that list contains coordinates of a detected bounding box along with the corresponding confidence score.

The detections results from each crop need to be aggregated to get the final result of the input image. Detections from each crop will be given with that crops top left point as origin. They should be re-scaled so that all the bounding box coordinates will be with respect to the input images top left corner. Since we use overlapping crops, there are possibility of detection of same object in multiple crops. These redundant detections are removed by using a class wise Non Max Suppresion (NMS). The class wise NMS first rejects overlapping bounding boxes of same class based on a confidence threshold value. The IoU of the remaining boxes are calculated and for boxes with IoU value above a specified IoU threshold, the one with the best confidence score is retained.

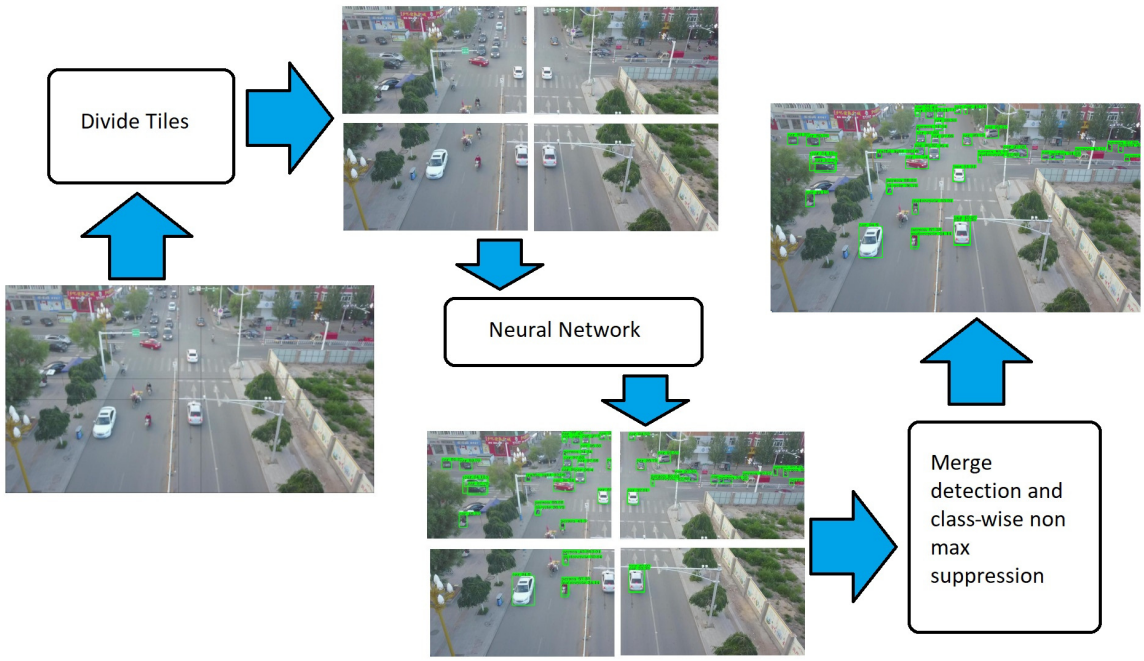


Figure 4.4: Flow of image through detection pipeline.

4.6 TRACKING: DEEP SORT

Simple Online and Realtime Tracking (SORT) is a simple tracking algorithm which uses a kalman filter to predict bounding box position in $(t+1)^{th}$ frame based on bounding boxes in the t^{th} frame. It then uses an association metric based on distance and the Hungarian Algorithm to link the predictions to detections. Deep Sort is a modified version of SORT, which adds in a feature based association metric along with the distance based metric. This helps associate detections with predictions more efficiently and in scenarios with occlusion and truncations [10].

4.6.1 Kalman Filter

Kalman filter is an algorithm that uses a series of measurement observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe [11].

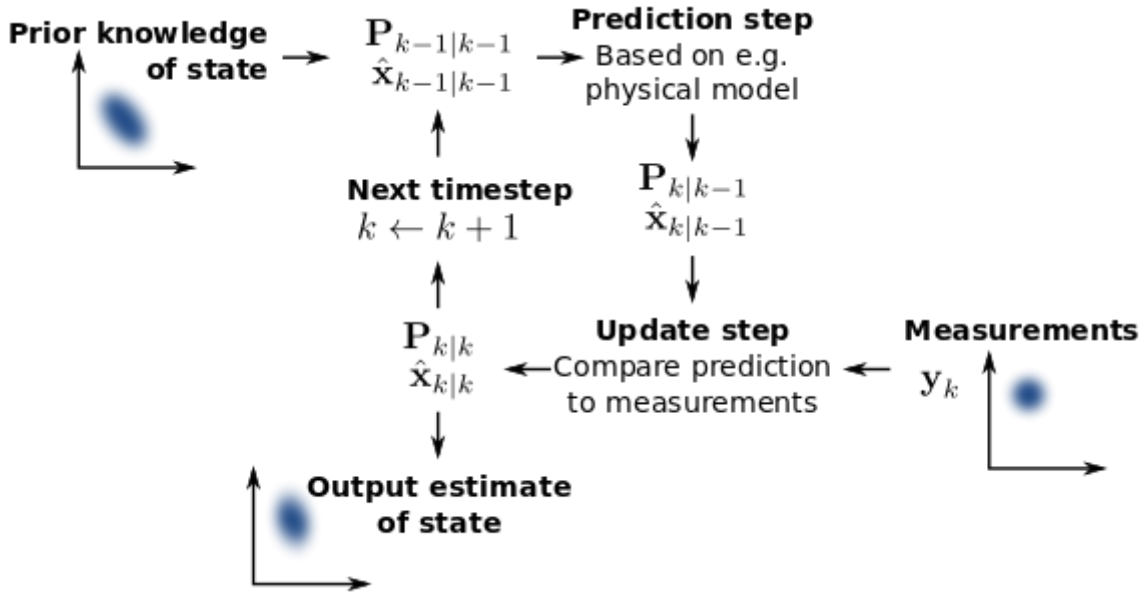


Figure 4.5: Kalman Filter

In Deep SORT, for every t^{th} frame, bounding boxes are predicted using information from the $(t-1)^{th}$ frame. When the detections for the t^{th} arrive, the noise in the predicions are taken into consideration and the calculated values are modified. This is a recursive process, where predictions are made and them modified to model the noise characteristics.Thus for object tracking for every identified track in the $(t-1)^{th}$ frame a prediction of where it will be in the t^{th} frame is made using kalman filter [11].

4.6.2 Association

The Association between the predicted Kalman state and the newly arrived detections is solved by using the Hungarian algorithm. Into this problem formulation, motion and appearance informations are integrated by appropriate metrics. To include motion information, the squared Mahalanobis distance(measure of the distance between a point P and a distribution D) between the predicted Kalman state and detections are measured. This takes state estimation uncertainty into account by measuring how many standard deviations the detection is away from the mean track location.The Mahalanobis distance is a suitable association metric when motion uncertainty is low.However, unaccounted camera motion can introduce rapid displacements in the image plane, making the Mahalanobis distance a rather uninformed metric for tracking through occlusions. Therefore, we integrate a second metric into the assignment problem.

For each bounding box detection we compute an appearance descriptor.A collection of last few associated appearance descriptors for each track k are also kept. Then, the second metric(appearance metric) measures the smallest cosine distance between the track and detection. In practice, a pre-trained CNN is used as feature extractor.For the project we use a siamese network as the feature extractor trained on people and vehicle images. Each bounding box detection is fed to the siamese network to find the feature vector and the cosine distance is calculated with the previously stored feature vectors [5].

For each track we count the number of frames since the last successful measurement association. This counter is incremented during Kalman filter prediction and reset to 0 when the track has been associated with a measurement. If no associations are made after

a specified threshold the track is supposed to leave the frame and is ignored. Similarly for newly entering objects, if no tracks are associated with them even after a threshold and the object still continue to exist in the frame, it is considered as a new object [5][10].

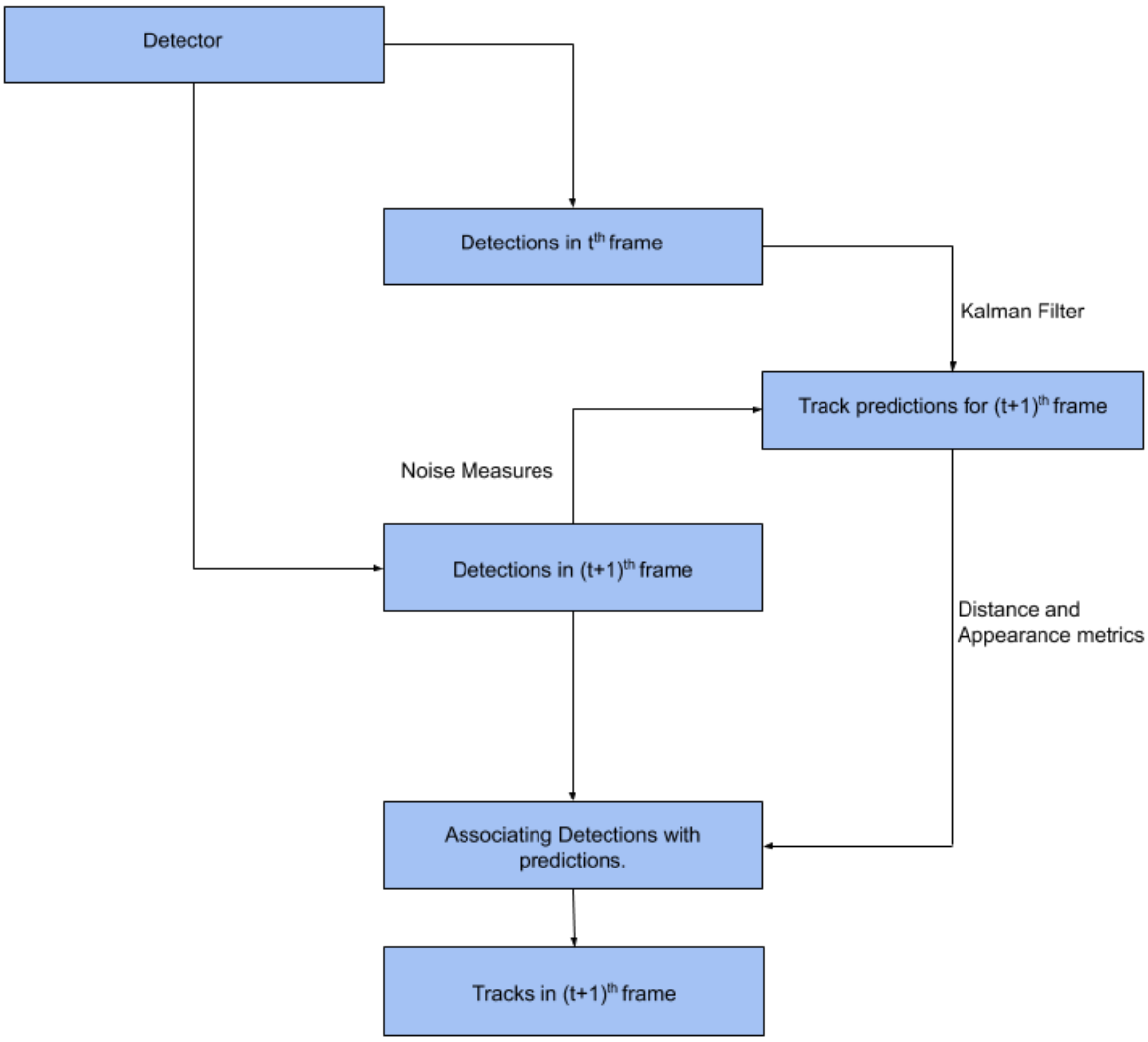


Figure 4.6: Flowchart: Tracking

4.7 TensorRT

TensorRT is a high-performance neural network inference optimizer and runtime engine for production deployment.TensorRT optimizes the network by combining layers and optimizing

kernel selection for improved latency, throughput, power efficiency, and memory consumption. If the application specifies, it will additionally optimize the network to run in lower precision, further increasing performance and reducing memory requirements.TensorRT based applications perform up to 40x faster than CPU-only platforms during inference.You can import trained models from every deep learning framework into TensorRT. After applying optimizations, TensorRT selects platform specific kernels to maximize performance on Tesla GPUs in the data center, Jetson embedded platforms etc [12].

After the neural network is trained, TensorRT enables the network to be compressed, optimized and deployed as a runtime without the overhead of a framework.TensorRT combines layers, optimizes kernel selection, and also performs normalization and conversion to optimized matrix math depending on the specified precision (FP32, FP16 or INT8) for improved latency, throughput, and efficiency. The optimizations includes,

- Elimination of layers whose outputs are not used
- Elimination of operations which are equivalent to no output
- The fusion of convolution, bias and ReLU operations
- Aggregation of operations with sufficiently similar parameters and the same source tensor
- Merging of concatenation layers by directing layer outputs to the correct eventual destination.

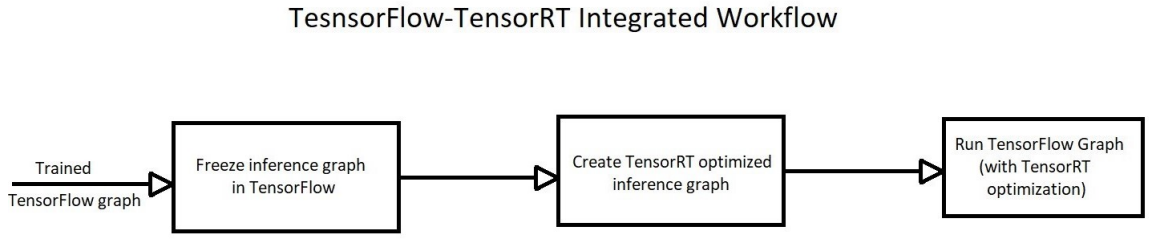


Figure 4.7: Workflow Diagram when using TensorRT within TensorFlow [12].

Chapter 5

Results and Discussion

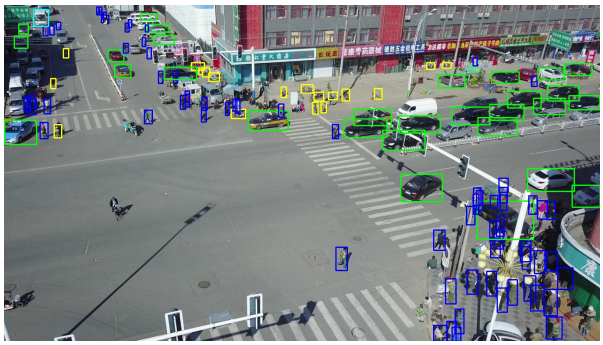
The detection pipeline incorporating static tiling have considerably improved detection performance. The number of misclassifications tend to reduce and the number of true positives increases. This is further enhanced by using tiling in training stage also. Tiling the training images remove the information loss due to down sampling at the input layer. Hence object features are retained in training and this helps during inference time. Figure 5.1 shows the difference in detections when using tiling and not.

We trained the custom YOLOv3 model with the Visdrone Train dataset which contains around 6300 images for 4 classes namely Pedestrian, Car, Bus and Motor. The images for training has been cropped as given in the design section. The training was done for 11000 iterations with an error of around 1.8. The performance of the model was tested on the Visdrone validation dataset which contains 548 images and the mean average precision at different IoU are as given below:

	Pedestrian	Car	Bus	Motor	mAP
$AP_{0.3}$	36.24%	75.68%	42.67%	26.33%	44.73%
$AP_{0.5}$	32.07%	72.79%	42.10%	23.18%	42.54%
$AP_{0.75}$	6.90%	52.01%	20.93%	3.92%	20.94%
$AP_{[0.5:0.95]}$	12.68%	45.42%	22.17%	8.64%	22.28%

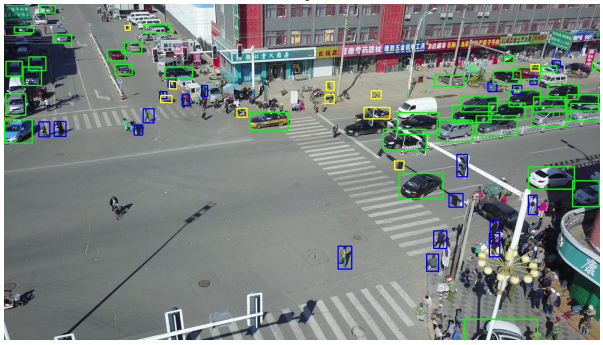
Table 5.1: Average Precisions on Visdrone validation set with tiling

The model was run on the same validation dataset without tiling to study the effect of the preprocessing in enhancing detections. The mean average precision at different IoU without tiling are as given below:



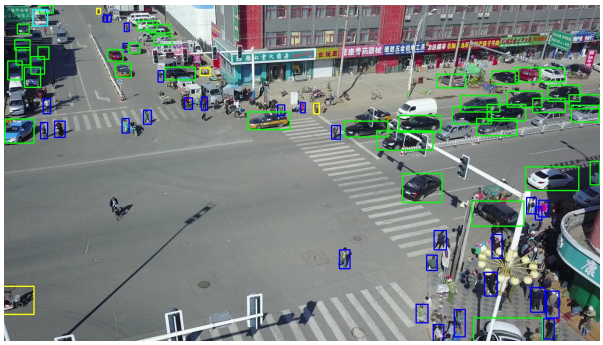
(a) Total objects = 111

(b) Total objects=134



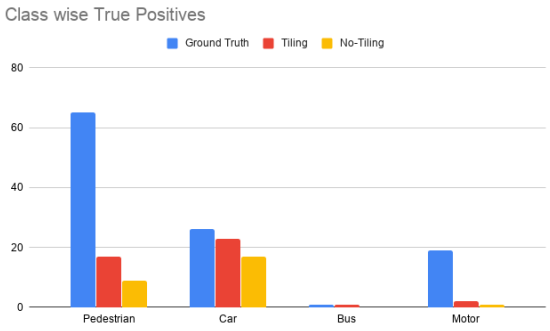
(c) Number of detections=67

(d) Number of detections=72

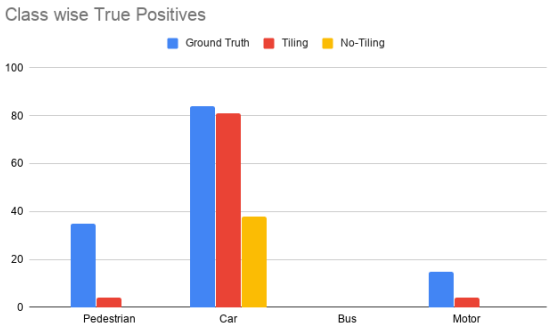


(e) Number of detections=72

(f) Number of detections=129



(g) True positives



(h) True positives

Figure 5.1: Comparison of detections with and without tiling-Top to bottom:Ground Truth,Without Tiling, With Tiling, True positive.

	Pedestrian	Car	Bus	Motor	mAP
$AP_{0.3}$	20.64%	64.37%	20.55%	18.71%	30.71%
$AP_{0.5}$	15.06%	58.35%	20.07%	13.0%	26.62%
$AP_{0.75}$	1.78%	30.72%	11.56%	0.88%	11.23%
$AP_{[0.5:0.95]}$	4.8%	31.315%	11.54%	3.736%	12.85%

Table 5.2: Average Precisions on Visdrone validation set without tiling

Using the tiling preprocessing has a considerable change in class wise mAP as well as in total mAP especially for the classes pedestrian and motor which are very tiny in the input images considered to the images size.

YOLOv3 is a one stage detector. One stage detectors in general have larger bounding box error, i.e, the error with which the bounding box has been predicted, compared to two stage detectors. This can be to a large extend decreased by training the model for a large number of iterations. This is the reason for reduced mAP at high IoU values.

For tracking, we use a probabilistic method with a neural network used only for feature extraction. The accuracy of the tracker depends primarily on the performance of the detector. If the detector can output correct detections,i.e, accurate bounding box and correct class, for each frame the kalman filter could predict boxes in subsequent frame and the association algorithm with the feature vector extractor could correctly associate previous tracks with current detections.

Chapter 6

Conclusion and Future Scope

The performance of the detector in UAV images has improved significantly. After training the model for one third of the required number of iterations, it's accuracy has increased greatly such that the number of true positives increased and also the error in bounding box prediction has decreased. The detector shows promising performance when compared to the results of top performers in the VisDrone'19 challenge. This performance is achieved by using only one of the two chosen datasets. Completing training of the model using the proposed method should further improve the detector performance.

The detector is performing at a satisfactory speed in traditional CPU systems. Comparing GPU and CPU performances of YoloV3, the detector should provide real-time detections on the Jetson TX2 device. Proper pipelining of the images for pre and post processing of the images and for detections can be applied to increase the average speed of detections.

The detector can be applied in any UAV scenarios that require image processing which include but not limited to drone based surveillance, rescue operations etc. The ability to detect even small objects in images that are below the standard detection thresholds enables surveillance using UAVs from great height or long distances.

In future, the performance of the detector can be further increased by using a custom variation of the model, adding facility for recurrent layers. This is due to the fact that the successive images that are processed hold high temporal correlation. Thus using recurrent models for detections and employing tracking techniques in the detection model should increase both the accuracy and speed of the detector.

References

[1] F. Özge Ünel,Burak O. Özkalaycı,Cevahir Çığla. The Power of Tiling for Small Object Detection, *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*

[2] VisDrone-DET2019: The Vision Meets Drone Object Detection in Image Challenge Results *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*

[3] Joseph Redmon, Santosh Divvala , Ross Girshick , Ali Farhadi. You Only Look Once:Unified, Real-Time Object Detection

[4] Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement

[5] Nicolai Wojke, Alex Bewley, Dietrich Paulus. SIMPLE ONLINE AND REALTIME TRACKING WITH A DEEP ASSOCIATION METRIC

[6] J. Cartucho and R. Ventura and M. Veloso. Robust Object Recognition Through Symbiotic Deep Learning In Mobile Robots *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*

[7] YOLOv3. *https://pjreddie.com/darknet/yolo.*

[8] Visdrone Challenge. *http://aiskyeye.com/*

[9] R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms - *https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e*

[10] DeepSORT: Deep Learning to Track Custom Objects in a Video *https://nanonets.com/blog/object-tracking-deepsort/*

[11] Kalman filter *https://en.wikipedia.org/wiki/Kalman_filter*

[12] TensorRT. *https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/index.html*

[13] Tensorflow. *https://www.tensorflow.org/versions/r2.0/api_docs/python/tf*