Modelling and Implementation of Real-time Animal Detection Module for Mobility Assistant for Visually Impaired (MAVI) System

by

Richa Verma

Under the Supervision of

Dr. Chetan Arora, IIIT Delhi
Prof. M. Balakrishnan, IIT Delhi

Indraprastha Institute of Information Technology, Delhi

July, 2017

Keywords: Computer Vision, Object Detection, Deep Learning, HOG

Modelling and Implementation of Real-time Animal Detection Module for Mobility Assistant for Visually Impaired (MAVI) System.

Richa Verma
IIIT-D-MTech-CS-DE-15-054

Submitted in partial fulfillment of the requirements
for the Degree of M.Tech. in Computer Science

to

Indraprastha Institute of Information Technology Delhi

July, 2017

Certificate

This is to certify that the thesis titled **"Modelling and Implementation of Real-time Animal Detection for Mobility Assistant for Visually Impaired (MAVI) System"** submitted by **Richa Verma** to the Indraprastha Institute of Information Technology Delhi, for the award of the Master of Technology, is an original research work carried out by her under our supervision. In our opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree. The results contained in this thesis have not been submitted in part or full to any other university or Institute for the award of any degree/diploma.

Dr. Chetan Arora

Dept. of Computer Science

IIIT Delhi

Prof. M. Balakrishnan

Dept. of Computer Science

IIT Delhi

Abstract

India is currently home to around 12 million blind people out of 39 million globally - one-third of the world's blind population resides in India. People with low vision or complete blindness find it hard to navigate themselves outside familiar environments. Tasks like simply walking down a crowded street may pose a great challenge. Because of this, many people with low vision will bring a sighted friend or family member to help navigate unknown environments. It calls for the need for a device to pose as a guide for such users. An interesting task that is necessary to be performed is safety from animals in an outdoor environment.

This thesis presents the design and the implementation of an animal detection module for MAVI (Mobility Assistant for Visually Impaired), an outdoor navigation system. The goal is to let the user be aware of animals (cows and dogs) present around them. The work involves measuring accuracy, performance and power/energy on Zedboard (embedded system) by varying configuration of various parameters of OpenCV functions. It includes cross-compilation and building of environment for Zedboard arm development and profiling algorithms. Both traditional Computer Vision techniques and Deep Learning methods for animal detection have been applied.

The exploration has been done while being mindful of other processes and applications that will be running alongside with this application. The speed and memory requirements of the proposed methods are also checked. The module provides useful parameters to a controller for monitoring performance and energy consumption according to the circumstances and needs of a VI person. The complete flow to animal detection is presented. A novel dataset of Indian cows with proper annotations that are suitable for evaluation of object detection techniques has also been released. A major challenge for such a module is to perform well with the diversity and complexities that arise in the Indian scenario due to non-standard practices.

Acknowledgements

I sincerely thank my supervisor Dr. Chetan Arora, IIIT Delhi for giving me the opportunity to work on this project and also his constant supervision and valuable guidance during the course of the thesis. I would also like to thank my co-supervisor Prof. M. Balakrishnan, IIT Delhi for his valuable insights and assistance. I extend my gratitude to Mr. Rajesh Kedia, IIT Delhi for providing us with his ideas and motivation.
A special thanks to my fellows, Sakti Saurav, Prerna Agarwal and Saurabh Agarwal for making my thesis journey fun and memorable and also for their help in cross-compilation.
I also thank my parents and my brother for always supporting me through my endeavours.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Physical movement is one of the biggest challenges for blind people. Traveling or simply walking down a crowded street may pose great difficulty. Because of this, many people with low vision will bring a sighted friend or family member to help navigate unknown environments. Blindness causes considerable social challenges, usually in relation to the activities in which a blind person cannot participate. Many of these social challenges limit a blind person's ability to meet people, and this only adds to low self esteem.

1.1 MAVI: Overview

The objective of MAVI System is to conceptualize and design a smart-camera based vision system, capable of extracting useful information, for example, faces, texture, animals and signboard information from captured images.

It consists of different modules integrated onto one platform, communicating with the user through a mobile interface and using cloud to translate the coordinates sent by Localization module into navigational information in a frame captured by the camera. The modules in MAVI System consist of Face Detection and Recognition, Texture Detection, Signboard Detection, Animal Detection and Localization [60,65].

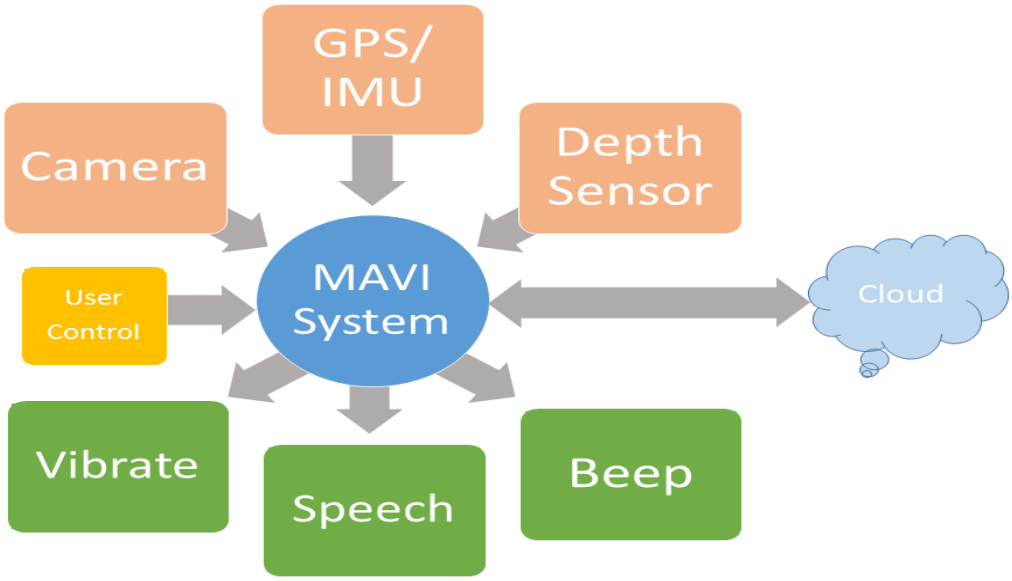

Figure 1.1: MAVI system

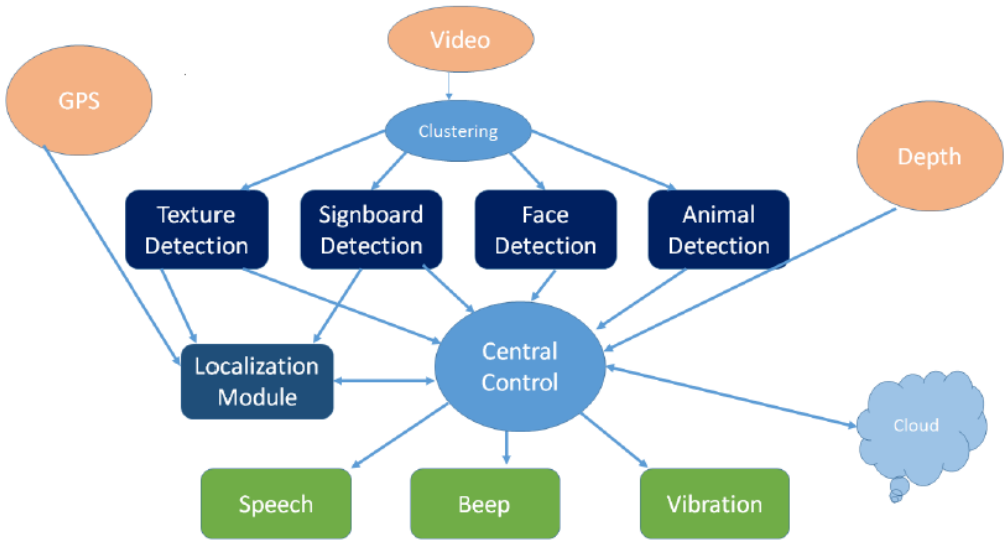

Figure 1.2: MAVI System and its modules

1.2 Objective

Presence of commonly found animals (cows and dogs) on Indian roads poses danger to a VI person. Such a person can unknowingly venture into close proximity of animals. This may annoy them and cause them to harm the VI person. Being fully aware of the animals in one's surroundings is, therefore, necessary.

The objective is to design and implement a real-time, online, energy-optimal algorithm for animal detection by using Computer Vision techniques and further explore deep neural network techniques for the same. The scope of this problem is limited to detecting cows in uncontrolled, outdoor environment. Animal detection module is indispensable for MAVI because of the unpredictability of animal actions in response to the stimuli in their environment. It is implemented along with the other modules to function as an integral unit of MAVI.

1.3 Challenges

These aspects should be considered:

- *Position/pose invariance*: There can be variation in poses and positions of animals.
- *Occlusions*: Truncations and occlusions result in the invisibility of important parts of an animal in images/videos, which makes it difficult to recognize them.
- *Illumination invariance*: Lighting conditions of the environment have a profound influence on the appearance of a scene/object. There can also be shadows depending on the position of light sources, which changes the appearance of the scene/object. These differences in appearance of the scenes/objects due to change in lighting conditions makes the task of visual categorization/retrieval difficult.
- *Robustness w.r.t. parameters*: Tuning the algorithm parameters (if

applicable) such that the algorithm can be used for a given set of constraints.

- *Computational complexity*: It should be kept under bounds permissible by the hardware of the system.
- *Proximity to animals*: The algorithm should be able to detect the an animal before the user is too near to an animal.

1.4 Hardware: ZedBoard

ZedBoard is based on Zynq All-Programmable SoC (AP SoC). This product integrates a feature-rich dual-core ARMCortex- A9 MPCore based processing system (PS) and Xilinx programmable logic (PL) in a single device, built on a state-of- the-art, high-performance, low- power process technology. It is a joint venture by Xilinx (Zynq AP SoC), Digilent (board manufacturer) and Avnet (distributor) .

- Xilinx Zynq XC7Z020-1CSG484 EPP (extensible processor platform) containing Dual-core ARM Cortex-A9 (PS) running at 666 MHz and Artix-7 FPGA (PL)
- Memory: 512 MB DDR3 and 256 Mb QSPI Flash
- On-board oscillators: 33.333 MHz (PS), 100 MHz (PL)
- Interfaces like USB-JTAG Programming, Ethernet, USB OTG, USB-UART,SD Card, Pmod, LEDs, Dip switches, Push buttons etc.
- Display: HDMI output, VGA (12-bit colour)
- Power: 12 V @ 5A AC-DC regulator

1.4.1 Operating System for Zedboard: Linaro

For the software-only implementation, Linaro Ubuntu distribution was used. Linaro is an open-source complete Linux distribution based on Ubuntu. It supports graphical desktop through on-board HDMI port. For booting from SD card, Linaro less system has to be placed in different partition other

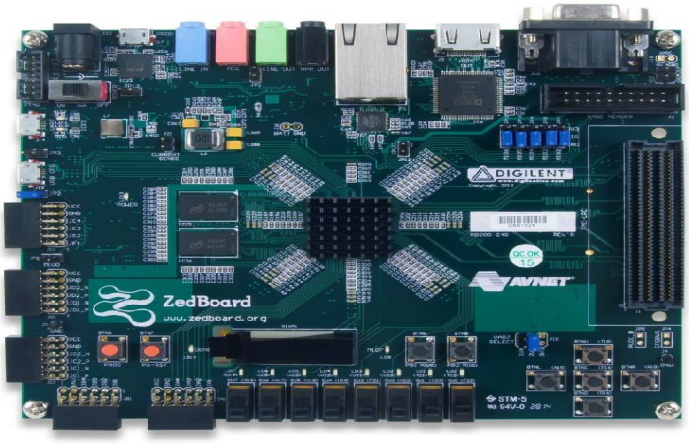

Figure 1.3: ZedBoard

than the partition where Kernel image,devicetree reside. It is a persistent OS, i.e. all changes are written to memory and it saves les after reboot or shutdown. OpenCV(version 3.1) was built on top of it. For booting Linux, 8 GB, class-10 SD card was used.

1.4.2 Cross-compilation for ZedBoard

Arm toolchain provided for linux ARM-linux-gnueabihf was installed and OpenCV 3.1 was source compiled using this tool chain for Linaro based systems. OpenCV was compiled both as a shared library and static libraries , stripped version of OpenCV was also compiled . The same tool chain is use to compile the application for Zedboard.

1.4.3 Pivothead

Pivothead is a wearable camera device, which can be used to record live video data and store it. That data can further be used as an input to be processed on a ZedBoard to produce an output.

Figure 1.4: Pivothead Camera

1.5 Thesis Outline

The rest of the thesis is organized as follows:

Chapter 2 gives an overview of object detection using Computer Vision techniques and deep neural networks.

Chapter 3 presents a new Indian cow dataset and the implementation of the animal detection module of MAVI.

Chapter 4 presents the findings of exploring the usage of deep neural networks for animal detection in Indian enviornment.

In chapter 5, conclusion along with the future work are presented.

Chapter 2

Background

Animal detection problem is a subset of object detection. This section presents an overview of the related work done in this domain.

2.1 Object Detection Using Traditional Computer Vision Techniques

Research in visual object detection has gathered considerable momentum in the past decade because it has finally reached the stage where many practical applications appear to be within reach. Object detectors have to cope with the effects of varying illumination, cluttered backgrounds, and widely varying image positions and scales. To adapt to these difficulties, researchers have developed increasingly informative descriptors and powerfull classifiers and training mechanisms.

Whole objects are typically too complex to generate reliable responses from generic feature detectors, so object detection approaches usually scan a dedicated object detection window densely across the image at multiple positions and scales. The simplest methods of this type are known as sliding window detectors.

Fig 2.1 illustrates the idea behind them. A vector of highly discriminative local visual features is calculated at each location of an image pyramid over the input image. A rectangular detection window is then scanned more or less densely over the feature pyramid, at each location evaluating an object/non-

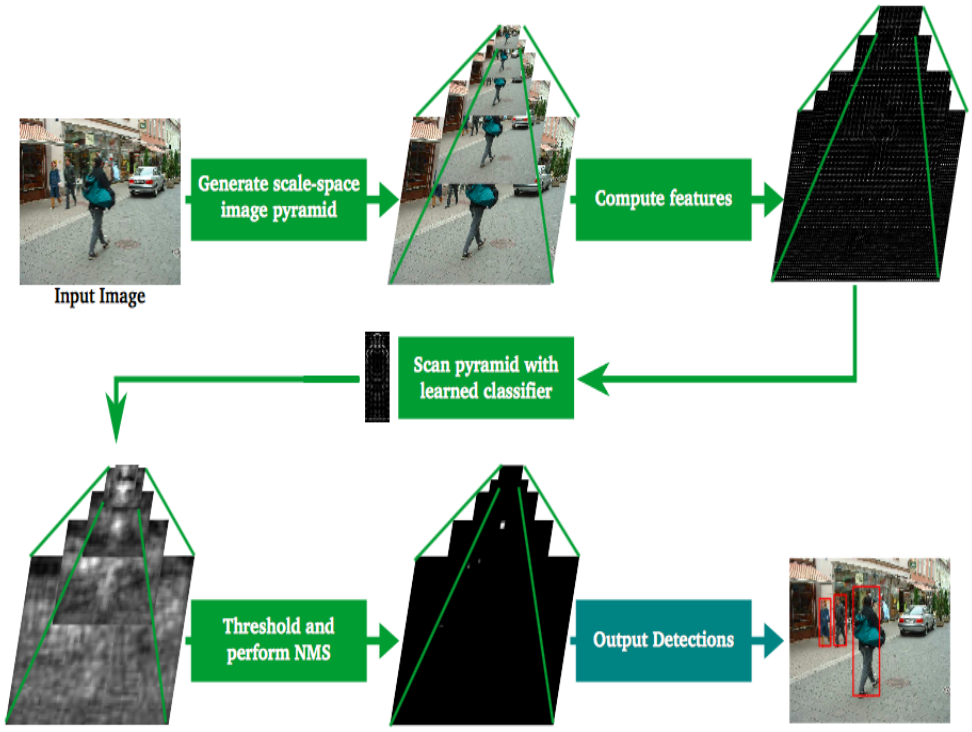

Figure 2.1: Sliding Window Detection (borrowed from [33])

object classifier using the features in the window. The resulting pyramid of classifier scores is then post-processed with thresholding and non-maximum suppression to produce the final detections. The window level classifier is essentially a modern variant of the traditional rigid object template.

Visual descriptors are a basic component of any object detection system. They must capture sufficient information to distinguish class instances from non-class ones while being resistant to photometric and geometric changes. People have tried many different kinds of descriptors including: edges and gradients [4,5,6], convolutional net filters [7,8], wavelets and Haar rectangles [9,10,11,12,13], Bag-of-Words descriptors [14,15], feature covariance matrices [16,17], local pattern texture features [18], etc. Recent approaches typically interpose some form of local spatial pooling between the pixels and the template matching to provide greater resistance to small spatial deformations of the object. For example, Scale Invariant Feature Transform (SIFT) [19]

and Histogram of Oriented Gradient (HOG) descriptors [20] are based on histogramming pixel-level oriented gradient descriptors into the cells of a regular spatial grid. Local Binary Pattern (LBP) descriptors [21] do the same with pixel-level local pattern texture codes. HOG and LBP are currently among the most popular descriptors for object recognition. Their main advantage over simpler features like Haar wavelets is the high discriminative power produced by locally-invariant pooling of features with high spatial resolution, fine orientation resolution and strong resistance to illumination variations.

Part-based models have proved successful on difficult object detection datasets recently. Deformable part models (DPM) by Felzenszwalb et al. [22] is an example successful method. It performs well on challenging datasets and handles variations in objects due to illumination differences. But it is slow, and speed proves to be a bottleneck for applications like MAVI where speed is as important as accuracy. When combined with HOG computation, DPM slows down further.

During the detection process, the classifier is scanned across the image at multiple scales and positions. For a give real class instance, this often produces several candidate detections that overlap closely in scale and space. As a post-processing step, the goal of the detection fusion or Non-Maximum Suppression (NMS) stage is to integrate or cluster these candidates to produce a single final detection at the appropriate position and scale without suppressing detections associated with other nearby objects. Several different methods of merging detections have been developed. We also use NMS in our algorithm to avoid redundant output.

2.2 Object detection Using Deep Learning

The annual ILSVRC[25] challenge uses images from the huge ImageNet dataset (more than 14 million images) that include up to 1000 different classes for the classification task and 200 classes for detection. In 2012, the challenge saw a significant improvement in performance, when Krizhevsky et al. [26] entered a convolutional neural network (CNN) for the first time. They were able to bring down the top-5 classification error from 25.2 % (for the second best entry) to 15.3 % and the localization error from 50 % to 34.3 %. Since

then, the ILSVRC has been dominated by convolutional neural networks of increasing depth.

The sliding window approach described in the previous subsection is extremely costly due to the vast search space. To reduce the number of times that the classifier (a CNN) needs to run, it could be applied on a coarser grid, but this has the risk of missing the ideal bounding box in some cases. Ideally, one would like to have an initial set of somehow proposed regions that need to be evaluated. It should be minimal while still containing all ground truth bounding boxes of objects in the image. This approach was successfully introduced by Girshick et al. [27] in their work called R-CNN. To generate the region proposals, they employed the Selective Search algorithm. Since then, they have improved the approach in [28,29], with the latest one presented as Faster-RCNN. The main idea is that use the last convolution layers to infer region proposals. Faster-RCNN consists of two modules: RPN (Region Proposal Network) and Fast-RCNN. RPN gives a set of rectangles based on deep convolution layer. Fast-RCNN and Roi Pooling layer classify each proposal and refine proposal location.

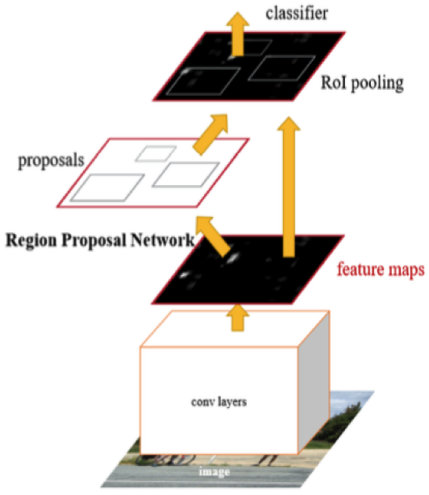

Figure 2.2: Faster-RCNN: Detection

YOLO [30] is a little bit less precise, but it is a high-speed detector. The idea of this detector is that a CNN model takes an image as input

and performs detection in a single pass. Firstly, the image is resized to 448x448, then fed to the network and finally the output is filtered by a non-maximum suppression algorithm. The coordinates of bounding boxes directly predicted in YOLO by using a convolutional feature extractor topped by fully connected layers.

Faster R-CNN predicts bounding boxes using hand-picked priors instead of predicting coordinates directly. Faster R-CNN predicts confidences and offsets for anchor boxes of varying dimensions by using only convolutional layers the region proposal network (RPN). Since the prediction layer is convolutional, these offsets are predicted by the RPN for a feature map's entire set of locations. The prediction problem gets simplified by choosing to predict offsets instead of coordinates. It also makes it easier for the network to learn.

SSD [31] detector differs from others popular detectors because it provides better accuracy on objects with varying scales due to the usage of multiple layers (each deeper layer sees bigger objects). SSD typically starts with a VGG or Resnet pre-trained model that is converted to a fully convolutional neural network. Then some extra convolutional layers are attached, which help to handle bigger objects. One important point to notice is that after the image is passed to VGG network, some convolution layers are added producing feature maps of sizes 19x19, 10x10, 5x5, 3x3, 1x1. These, together with the 38x38 feature map provided by VGG's conv4_3, are the feature maps which are used to predict bounding boxes. SSD can be trained with relatively less number of training images, and it is tested for MAVI.

For animal detection in MAVI, using such popular object detection methods to run on an embedded system is challenging. Since it is a real-time system, high detection speed is desirable. Accuracy is of importance, too. A recent work [32] by Google compares speed/accuracy trade-off for such methods. MobileNets [34] present a class of models that are suitable for embedded vision and mobile-based applications. MobileNets are light weight deep neural networks that use a streamlined architecture that consists of depth-wise separable convolutions. They are further explored in the last section.

Chapter 3

Implementation of Animal Detection Module

This chapter discusses the algorithm and the software implementation of the animal detection module for MAVI. Section 3.1 explains the overall approach used for animal detection. A new dataset of Indian cow images and the implementations of the detection algorithms for animal detection, along with their limitations, are presented in section 3.2. The results and the outputs of the algorithms are shown in section 3.3. Section 3.4 consists of the details of hardware porting of the algorithm. MAVI as an android app is explained in section 3.5.

3.1 Implementation Approach

This section covers the overall implementation approach for the animal detection module.
Histogram of Oriented Gradients feature descriptor is used to provide features for classification by SVM. The algorithm is implemented in C++ by using OpenCV [1]. OpenCV comes with a number of optimized algorithms to provide a common infrastructure for efficiently developing computer vision applications. OpenCV provides detectMultiscale function to be used along with HOG descriptor to perform object detection with multi-scale window. It has many parameters like window stride, hit threshold, window size, scale factor etc. [2]

As mentioned earlier, MAVI is composed of different modules along with control system. Control system plays an important role in scheduling different modules and extracting results from them. It also controls different modes of operation of different modules. Figure 3.1 represents a basic implementation approach followed for animal detection. As shown in the figure, control system accepts a live stream as input from the pivothead camera and extracts frames from it. These frames are then passed to the animal detection module.
The animal detection module receives an input image. On the received image it runs animal detection algorithm, which detects the cows (if present) in the image and outputs their coordinates . The coordinates of identified cow faces and bodies in the image along with their respective labels are then returned to the control system.

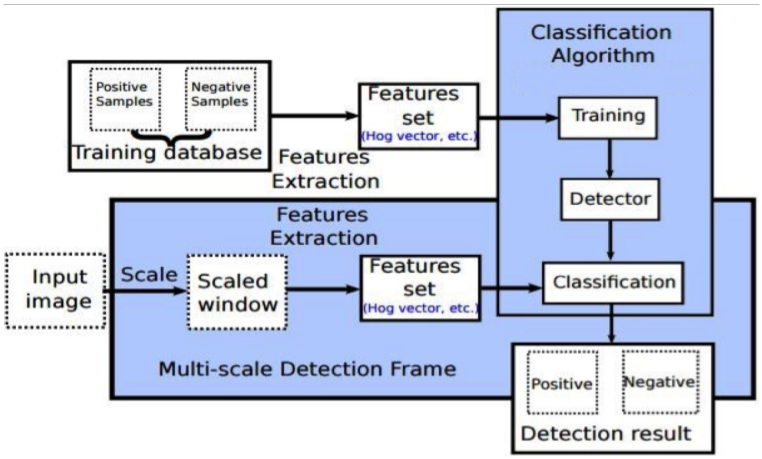

Figure 3.1: Animal Detection Module Design

3.2 Cow Dataset-1.0

For the task of training and testing a classifier for animal (cow) detection and to further build a multi- scale detection pipeline, a database of images of Indian cows is created. The quality of database and accuracy of recognition depends on the camera used to capture the database. The images are captured in different lighting and background conditions. Various poses,

positions and colours of cows are also included. Some images in the database have been annotated with a bounding box around each cow present in an image and annotations are saved as XML files in the PASCAL VOC [3] format.

Figure 3.2: Images from Cow Dataset-1.0

Number of Images with Annotations	1603
Front view	218
Side view	1493
Back view	335
Standing pose	1958
Sitting pose	63
Illumination Conditions	Sunny, Shadowy
Colors	Black, Brown, White, Grey

Table 3.1: Cow Database Details

The above table gives a brief detail of our cow database for the annotated images.
To train the classifiers, 2,600 cropped images of cow faces (64x64), 2,000 cropped images of cow bodies (128x64) and 13,000 cropped patches of other possible background objects (64x64), labeled as negative, are used to train the classifiers needed for performing detection. The images in the dataset have been collected by using Intel RealSense camera to shoot videos, while mounted on chest. Some of the data has also been taken from videos on YouTube.

3.3 Histogram of Oriented Gradients Feature Descriptor

A feature descriptor is a means of representing an image to extract useful information from it and discard the extraneous information from it.

The histogram of oriented gradients (HOG) is a feature descriptor used for the task of object detection the fields of computer vision and image processing. The technique counts occurrences of gradient orientation in localized portions of an image. The idea behind the histogram of oriented gradients feature descriptor is that histograms of intensity gradients or directions of edges in an image can be used to represent the appearance of an object and its shape. The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. The descriptor is the concatenation of these histograms. Contrast normalization is performed over a larger group of cells in an image, called a block, for improved accuracy. This normalization results in better invariance to changes in illumination and shadowing.

The HOG descriptor has a few key advantages over other descriptors. Since it operates on local cells, it is robust to illumination changes and geometric transformations. It is not invariant to object orientation.

- **Computation of HOG features**

1. An image is divided into 88 cells and a histogram of gradients is calculated for each of those cells.
2. For every cell, a histogram of 9 bins is computed. 1 bin equals 20 degrees. The histogram is essentially a vector of such 9 bins corresponding to angles 0, 20, 40, 60 ... 160.
3. Each pixel inside an 8x8 cell votes for its bin.
4. Block normalization is the next step. It makes HOG the robust to illumination changes.
5. 1 Block = 2 cells x 2 cells. Blocks have 50% overlap.
6. Block normalization is performed by concatenating the histograms of the four cells within the block into a vector with 36 components (4 histograms x 9 bins per histogram). This vector is divided by its magnitude to normalize it.
7. For a 64x128 window, there are 7x15 = 105 blocks. So the size of the vector =105x4x9 = 3780.

3.4 SVM

Support vector machines (SVMs) are supervised learning models that are used with certain learning algorithms for classification and regression analysis in machine learning. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model provides a representation of the example data points as points in space, such that the data points belonging to different classes are separated by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on on which side of the gap they fall.

It is a discriminative classifier essentially defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

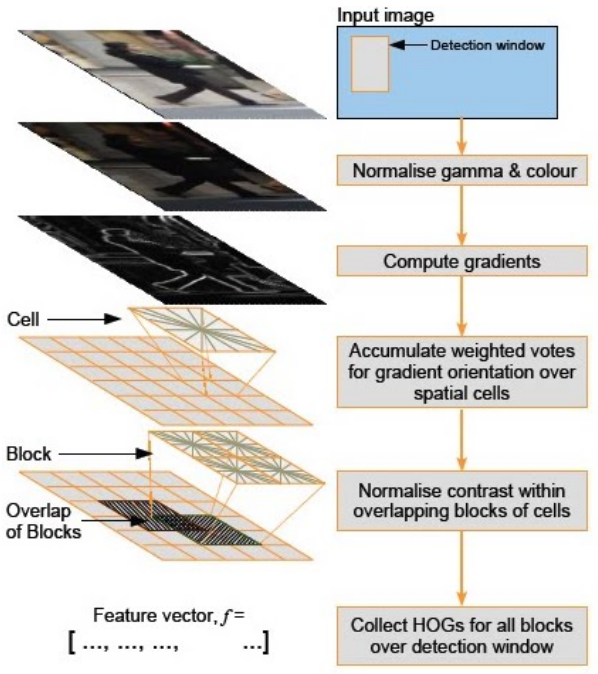

Figure 3.3: An overview of HOG feature extraction

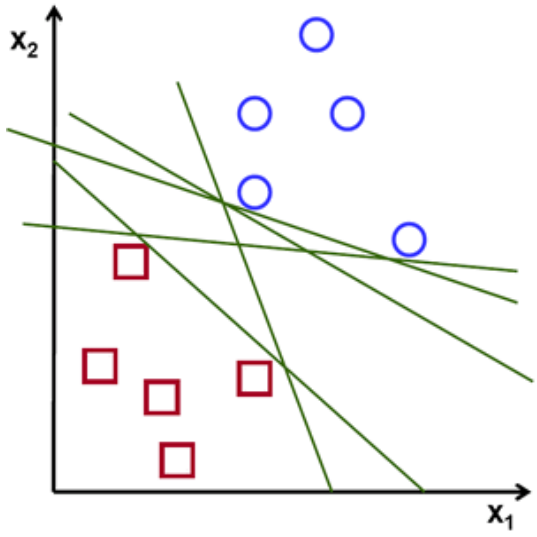

Figure 3.4: Possible decision lines

In the above picture, multiple lines are shown that offer a solution to the problem. A line is considered as bad if its distance is less from the points because it is sensitive to noise and it does not generalize correctly. Therefore, the goal is to find the line passing as far as possible from all points. Then, the operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples. Twice this distance receives the important name of margin within SVM's theory. Therefore, the optimal separating hyperplane maximizes the margin of the training data.

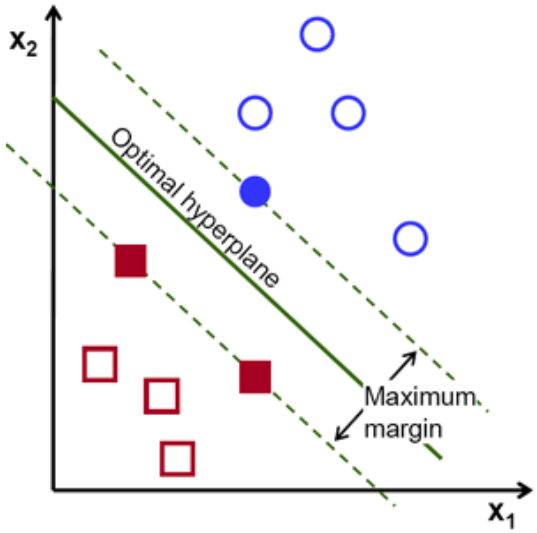

Figure 3.5: Optimal separating hyperplane

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces

3.5 Cow Detection Algorithm

This algorithm presents the training and testing procedure followed for cow detection.

- **Dataset:Cow Dataset-1.0**
- **Training**:
    1. Compute HOG features for an image from training set that consists of cropped faces (for front-view classifier) of size 64x64 and cropped bodies (for side-view classifier) of size 64x128.
    2. Train SVM classifier using the HOG features.
    3. Test the classifier on the training data.
    4. Add false positives as 64x64 patches to the training set to perform hard-negative mining.
    5. Repeat 1 and 2.

    The same algorithm is repeated for training a side-view classifier for full-body detection of cows.

- **Testing:**
    1. Image of size 640*480 is used as input.
    2. It is divided into several scales, scale is coefficient for detection window increase.
    3. For each scale, the image is resized at each layer of the image pyramid by that factor.
    4. For each layer of the pyramid, a sliding window with 'winStride' steps is moved across the entire layer.
    5. HOG features are extracted from the window and classifiers are run, front-view classifier is run first, followed by the side-view classifier.
    6. Non-maximal suppression is used to combine overlapping detections with more that 60% overlap in bounding-boxes.

3.5.1 Limitations

- Lot of false positive detections.
- HOG feature descriptors of many different objects look very similar.

3.5.2 Observations

- False positive detections rarely exist in 3 or more consecutive frames.
- True positive detections are not present in consecutive frames even when not much of the scene has changed.

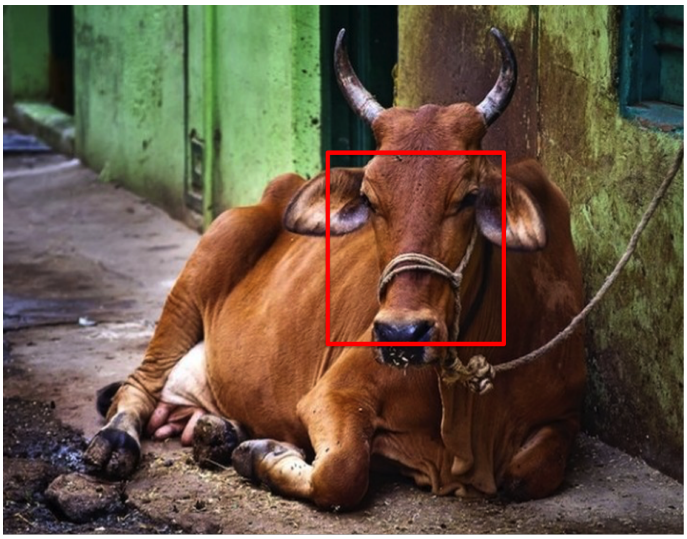

Figure 3.6: Output of the algorithm on an image from Cow Dataset

3.5.3 Cow Detection Algorithm: Improved

This algorithm presents an improvement over the algorithm described earlier. False positive detections are a major limitation of the previous algorithm due to highly similar HOG features of cows and various other objects. It is observed that false positives mostly don't occur in the same image area for 3 or more consecutive frames. To build upon this observation, this improved algorithm uses temporal pooling to calculate the IoU overlap among the detected bounding boxes in three consecutive frames. It only counts a detection as a cow if there is at least 50% overlap among the detected boxes. Then, tracking is applied on the detected cows over the subsequent frames. Once a detection is made, it is tracked by using Lucas-Kanade algorithm [] to compute sparse flow (and its corresponding function [] in OpenCV). Detection algorithm is run on every 5th frame of the incoming video feed.

- **Dataset:Cow Dataset-1.0**
- **Training**:
    1. Compute HOG features for an image from training set that consists of cropped faces (for front-view classifier) of size 64x64 and cropped bodies (for side-view classifier) of size 64x128.
    2. Train SVM classifier using the HOG features.
    3. Test the classifier on the training data.
    4. Add false positives as 64x64 patches to the training set to perform hard-negative mining.
    5. Repeat 1 and 2.
       The same algorithm is repeated for training a side-view classifier for full-body detection of cows.

- **Testing:**
    1. Image of size 640*480 is used as input.
    2. It is divided into several scales, scale is coefficient for detection window increase.

3. For each scale, the image is resized at each layer of the image pyramid by that factor.
4. For each layer of the pyramid, a sliding window with 'winStride' steps is moved across the entire layer.
5. HOG features are extracted from the window and classifiers are run, front-view classifier is run first, followed by the side-view classifier.

6. Non-maximal suppression is used to combine overlapping detections with more that 60% overlap in bounding-boxes.
7. The detected bounding boxes in the current image are compared with those in the previous and the next image.
8. The bounding boxes with an IoU of at least 50% are counted towards detections.
9. Tracking (using Lucas-Kanade algorithm) is applied on the detections that qualify in the previous step.

3.5.4 Limitations

- False positive detections are considerably reduced but they still exist.
- Detection time is increased.
- Recall drops.

3.6 Results and Output

The results and observations of a software implementation of the proposed algorithms are presented here.

3.6.1 Results: Algorithm 1

The first algorithm consists of running front view cow classifier followed by the side view classifier on an input image. To analyze the performance of

the software implementation of the cow detection algorithm, the distribution curves for different scaling factors are plotted.

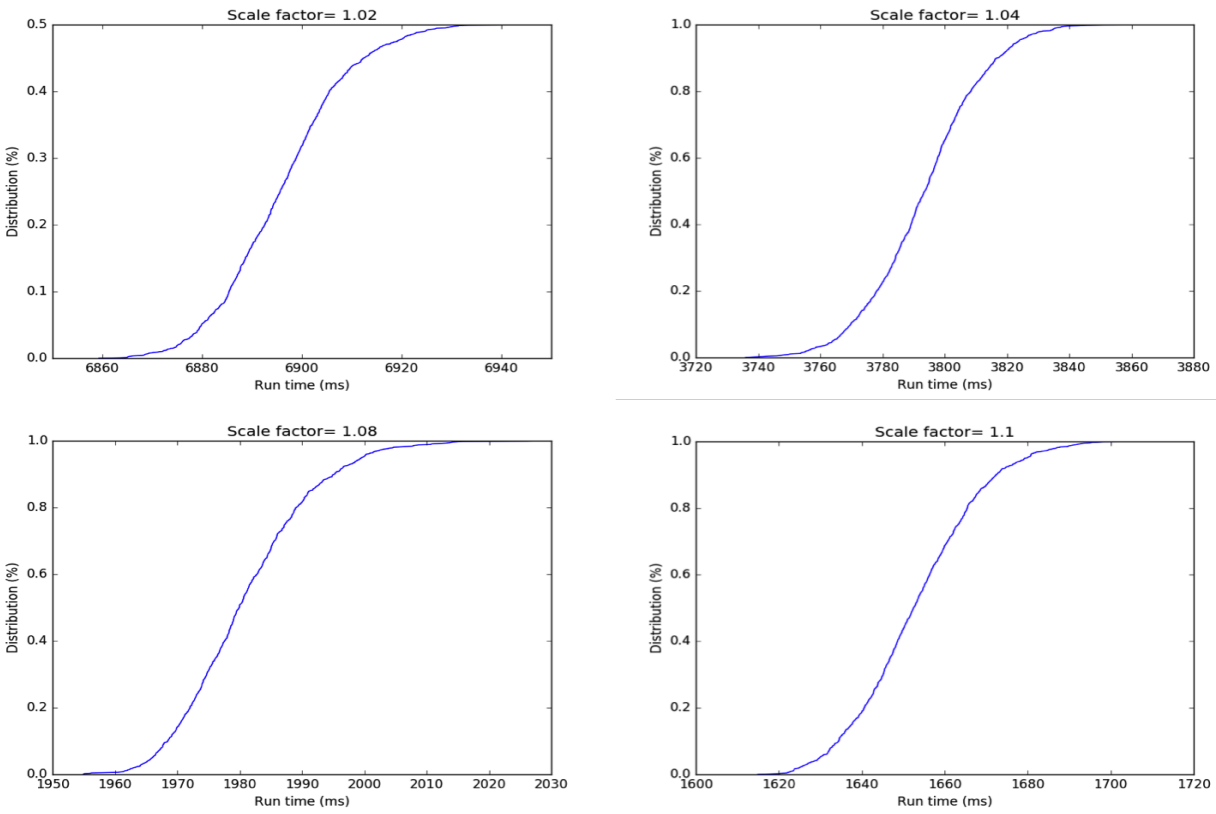

Figure 3.7: Distribution curves for performance at different scaling factors

The accuracy details of the classifiers are reported below.

	Front View		Side View	
Scale factor	Precision (%)	Recall (%)	Precision (%)	Recall (%)
1.02	64.5	89	63.2	81.5
1.04	60	80.1	60.5	78.4
1.06	55	69	53	68.2
1.1	35.2	48.4	40.5	58.5
1.2	18.1	34	29	33.2

Table 3.2: Precision and Recall: Algorithm 1

3.6.2 Results: Algorithm 2

The second algorithm consists of running front view cow classifier followed by the side view classifier on an input image, followed by temporal pooling and tracking. The improved algorithm leads to better precision but it comes

	Front View		Side View	
Scale factor	Precision (%)	Recall (%)	Precision (%)	Recall (%)
1.02	79.3	82.5	77.5	78.8
1.04	68	72.4	66.5	73

Table 3.3: Precision and Recall: Algorithm 2

with a loss of recall, for both classifiers. Tracking helps in increasing recall values. It is because it was observed in the output of the first algorithm that true positive detections are not present in consecutive frames even when not much of the scene has changed. However, false positives occurring in frame get tracked, too.

3.6.3 Output

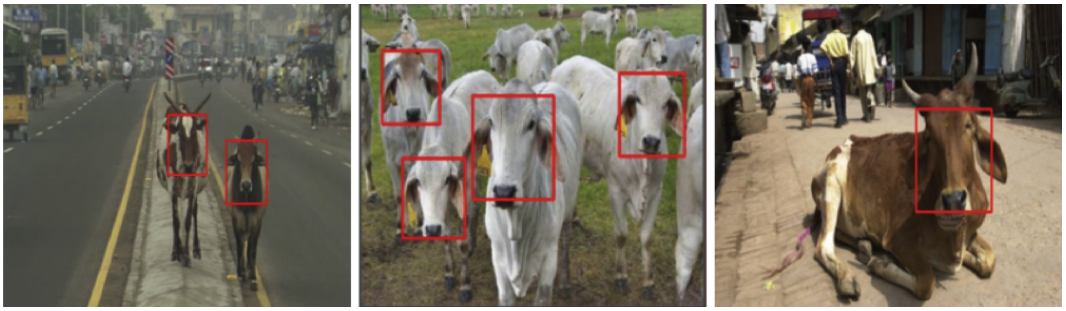

Figure 3.8: True Positive Detections from Front View Classifier

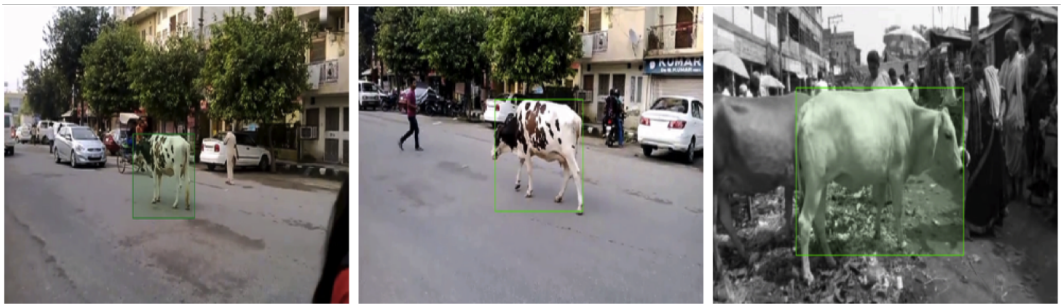

Figure 3.9: True Positive Detections from Side View Classifier

Figure 3.10: False Positive Detections

Figure 3.11: False Negative Examples

Figure 3.12: Reduced false positives from Algorithm 2

3.7 Porting to Hardware: ZedBoard

The algorithms for cow detection are initially developed and tested on Intel x86 processor. Further, they are ported to the embedded platform for our application's purpose. The embedded platform that is used packs in a Cortex-A9 ARM processor. Thus, all the OpenCV libraries used in algorithm are cross-compiled for ARM processor to port the algorithms to the embedded platform. While using OpenCV libraries on the Intel processor, shared libraries are built as memory is not a constraint. But, for using on the embedded platform, optimum usage of the limited available memory should be made. Thus, all the libraries are cross-compiled and built as static libraries. Subsequently, the code is compiled on the computer itself using arm-based toolchain and static OpenCV libraries.
The toolchain used is "arm-linux-gnueabihf-g++" for compiling the code. One needs to be careful about the sequence of libraries while compiling the code using static libraries. The following command is used.

3.8 Porting to Android

Since, the ZedBoard Hardware is difficult to carry due to space, energy consumption, an app was created which processed real time video through phone's camera. It processes frames at a normal walking speed of 1 frame/sec and gives the color coded output to guide the user.

Figure 3.13: Output on Android app (without NMS)

Chapter 4

Animal Detection Using Deep Neural Networks

4.1 Single Shot Multibox Detector

Faster RCNN and most of the other current state of the art techniques for object detection suffer from similar problems, and SSD aims to deal with them. All the object detection algorithms have same methodology:

- Train 2 different networks - Region Proposal Net (RPN) and an advanced classifier to detect the bounding box and the class of an object separately.
- During inference, run the test image at different scales to detect an object to account for invariance.

Such a methodology makes the nets perform very slow. Faster RCNN could result in 73.2% mAP at 7 FPS while 74.3% mAP was achieved by SSD at 59 FPS on VOC 2007 dataset.

4.1.1 SSD: Methodology

A key feature that differentiates SSD from the other state of the art methods for object detection is that it uses a single network to predict both object class and bounding box. It deploys a technique to pick ROIs. End-to-end

training is performed to predict a class and compute the boundary shift for a particular ROI.

Selecting ROI

Like Faster RCNN, SSD works on the same concept of using anchor boxes to create ROIs. All this is done by using the feature maps of the last layer of shared convolution layer. In a layer of such feature maps, k anchor boxes which have varying aspect ratios are picked to be around each pixel. So if a feature map has a resolution of m x n then, that amounts to m x n x k ROIs for that layer. To deal with detecting objects of various sizes, SSD uses many such feature layers that have different dimensions to generate the ROIs. Earlier layers in a deep convolutional network capture only low-level features, it utilizes features maps from after certain levels and layers.

Labelling ROIs

After going through some specific transformation, If an ROI matches with the Ground Truth for a class and has 0.5 or more Jaccard overlap value then it is labeled as positive. Different resolutions of feature maps and varying aspect ratios of each box make the task more challenging. To handle that, SSD performs scaling and works with different aspect ratios to be close to the ground truth dimensions. This facilitates the calculation of Jaccard overlap for default boxes for each pixel at a particular resolution.

Classification of ROIs

SSD uses a small kernel of dimensions 3*3*p (p channels) to predict the value of four offsets (cx, cy , h, w) for each candidate ROI from the Ground Truth box with a confidence score for every class.

So for each box out of k at a given pixel location, c class scores and four offset values relative to actual default box shape are computed. ROIs are generated by using multiple feature maps with varying dimensions. The ROIs are labeled as positive or negative based on the value of the Jaccard overlap after the ground box has scaled appropriately to deal with the differences in

the resolutions of the input image and feature map.

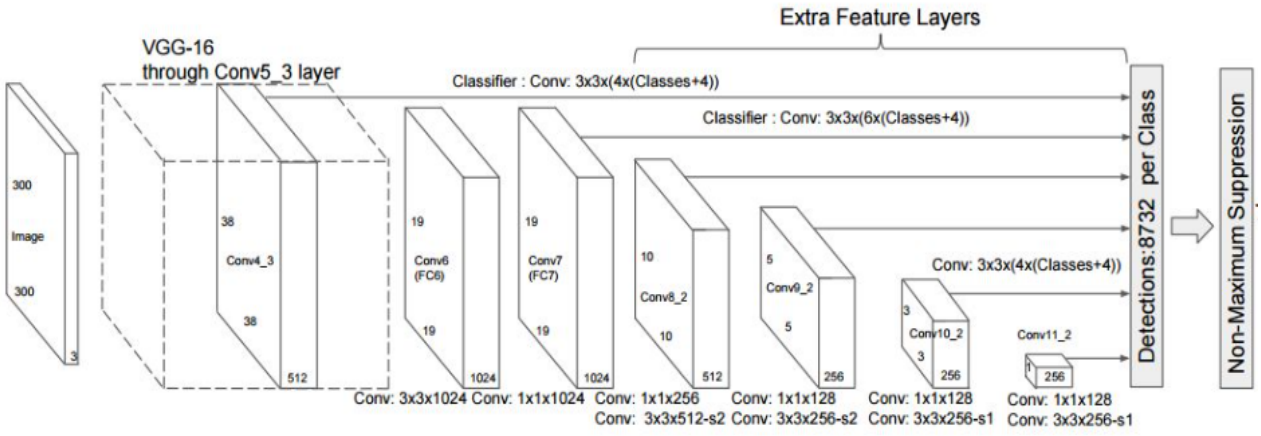

Figure 4.1: SSD Architecture (borrowed from [31])

Training

For each ROI, loss is computed which is a weighted sum of localization error and classification error (confidence):

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{4.1}$$

Inference

For each ROI, irrelevant predictions are identified and removed by using a small threshold on the classification score. On the remaining predictions, Non Maximum Suppression (nms) with Jaccard overlap of 0.45 per class is applied and based on that, the top 200 detections per image are picked.

4.2 Observations

The following table provides the accuracy statistics of testing SSD with VGG Net on 1436 images from Cow Dataset-1.0.

True Positives	984
False Positives	30
False Negatives	470
Precision	97%
Recall	67%

Table 4.1: Testing SSD on Cow Dataset 1.0

SSD missed quite a lot of cows in the dataset. Some of the false negative examples are shown below:

Figure 4.2: Some examples of False Negatives

4.3 SSD using MobileNets

MobileNets [34] are the latest offering from Google. They are suitable for embedded and mobile devices because they have a small size, low latency and need low power. They can be used on platforms with resource constraints for various use cases. They can be used like some popular, but large scale, models such as Inception Net for the tasks of classification, detection and segmentation. Using TensorFlow Mobile [35], mobile devices running Android [36] and embedded devices like Raspberry Pi [37] can be used efficiently to run MobileNets.

MobileNets have achieved an accuracy of the level of VGG-16 on Imagenet with only 1/30 of the model size and computational cost. This is possible because of performing factorization of a standard convolution operation by using depthwise separable convolutions. A single layer filter is used for each channel of the input and then the channel information is combined by using pointwise 1x1 filter, as shown in the figure below. This significantly reduces the number of parameters and the model size.

Before considering porting MobileNets to Raspberry Pi, some tests are run on a MacBook Air with 1.6GHz Intel Core i5 and 4GB memory to check the performance. The model runs at 8fps and takes roughly 340MB memory. It takes less than 1GB memory in all settings tested, which makes it suitable for a Raspberry Pi 3. The following table provides the accuracy statistics of testing SSD MobileNet model (trained on COCO dataset) on the same 1436 images from Cow Dataset-1.0 that were used to test SSD.

Precision	99.29%
Recall	86.4%
True Positives	1275
False Positives	9
False Negatives	205

Table 4.2: Testing MobileNet SSD model on Cow Dataset 1.0

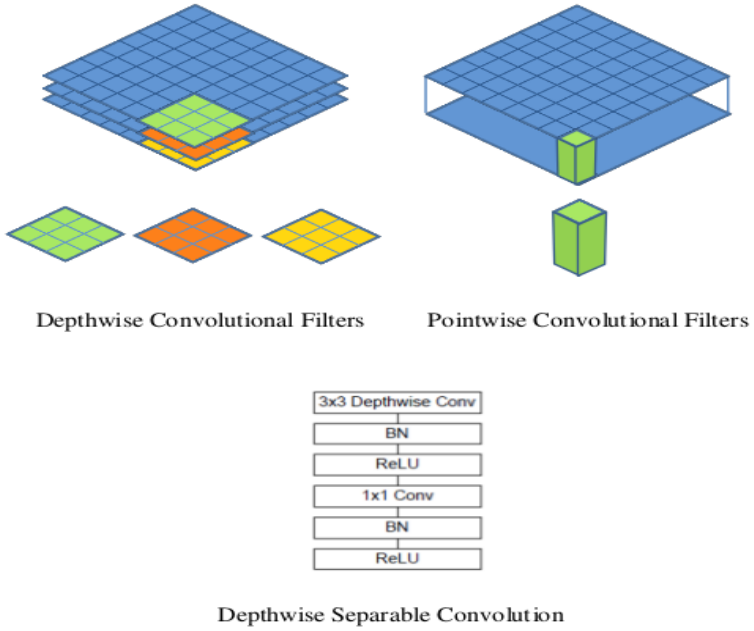

Figure 4.3: Depthwise and Pointwise Convolution in MobileNet (borrowed from [34])

For a marginal loss in accuracy, MobileNets perform well with much less latency and size as compared to the popular models from the literature. It is expected that object detectors like SSD can be utilized for animal detection task in MAVI when coupled with MobileNets on a Raspberry Pi 3 board.

Chapter 5

Conclusion and Future Work

This thesis presents the design and implementation of the animal detection module of Mobility Assistant for Visually Impaired (MAVI) system. Both traditional Computer Vision techniques and deep neural network based approaches are tested. For this purpose, a novel dataset consisting of 1,603 images of Indian cows is released. Each image in the dataset comes with its own XML annotation file. Apart from these pictures, there are cropped patches of faces and bodies of cows, too. This dataset is used to train models for cow detection.

Firstly, two classifiers are trained for cow detection, one for the face and the other one for full-body detection. Training data is formed by cropped patches of cow faces and bodies. Training is performed on HOG feature descriptors of such patches and classification is performed using linear SVM. During the testing phase, multi-scale pyramid based detection is performed while varying the scale factor. To make the detector more robust to false positives, hard negative mining is performed twice. However, this detector suffers from low precision due to a high number of false positive detections.

It is observed that false positive detections don't typically occur in consecutive frames over nearly the same location. As an improvement to the first algorithm, a detection is only reported during the testing phase if it overlaps with a detection in the previous and the next frame, considerably. This reduces the number of false positives. It increases precision, but it brings down recall, as well. This happens because true positive detections are not consistent in consecutive frames, either. As a corrective measure, tracking

is performed on the regions detected as cows. This algorithm gives better results than the first one, but it is slower than the first one.

Usage of deep neural networks for cow detection is also explored while keeping in mind the issue of porting the model to Zedboard or Raspberry Pi. Single Shot Multibox Detector (SSD) is also tested on the same dataset. It gives around 97% precision, but recall is low. The number of false positives reported is much less than that in traditional Computer Vision approaches explained before. But, the model needs a lot of memory to run.

MobileNets are the latest offering from TensorFlow [23] that is suitable to be run on embedded and mobile devices. TensorFlow object detection API [24] can be used to train and test such models with reduced memory requirements. It is shown that object detectors like SSD can be utilized for animal detection task in MAVI when coupled with MobileNets. Porting MobileNet model to a platform such as Raspberry Pi is the next logical step in that direction. This task is work in progress.

Bibliography

[1] OpenCV: http://opencv.org/opencv-3-1.html.

[2] HogDetectMultiscale: http://docs.opencv.org/2.4/modules/gpu/doc/object_detection.html.

[3] Pascal VOC 2012: http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[4] DM Gavrila, V Philomin, *Real-time object detection for" smart" vehicles*, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Fort Collins, USA, pages 87–93. 1999. pages 12, 13.

[5] N. Dalal and B. Triggs. *Histograms of oriented gradients for human detection*, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego, USA, pages 886–893. 2005.

[6] P. Felzenszwalb et al. *Object detection with discriminatively trained part based models*. In IEEE Transactions on Pattern Analysis and Machine Intelligence, 2009.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. *Gradient-based learning applied to document recognition*, In Proceedings of the IEEE, 86(11):2278–2324, 1998.

[8] C. Garcia and M. Delakis. *Convolutional face finder: A neural architecture for fast and robust face detection*, In IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(11):1408–1423, 2004.

[9] A. Mohan, C. Papageorgiou, and T. Poggio. *Example-based object detection in images by components*, In IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(4):349–361, 2001.

[10] P. Viola and M. J. Jones. *Robust real-time face detection*, In International Journal of Computer Vision, 57(2):137–154, 2004.

[11] C. Papageorgiou and T. Poggio. *A trainable system for object detection*, In International Journal of Computer Vision, 38(1):15–33, 2000.

[12] H. Schneiderman and T. Kanade. *Object detection using the statistics of parts*, In International Journal of Computer Vision, 56(3):151–177, 2004.

[13] P. Dollàr et al. *Multiple component learning for object detection*, In Proceedings of the 9th European Conference on Computer Vision, Marseille, France. 2008.

[14] M. Blaschko and C. Lampert. *Learning to localize objects with structured output regression*, In Proceedings of the 9th European Conference on Computer Vision, Marseille, France, pages 2–15, 2008.

[15] H. Harzallah, F. Jurie, and C. Schmid. *Combining efficient object localization and image classification*, In Proceedings of the 12th IEEE International Conference on Computer Vision, Kyoto, Japan, pages 237–244. IEEE, 2009.

[16] O. Tuzel, F. Porikli, and P. Meer. *Region covariance: A fast descriptor for detection and classification*, In Proceedings of the 8th European Conference on Computer Vision, Graz, Austria, pages 589–600. Springer, 2006.

[17] O. Tuzel, F. Porikli, and P. Meer. *Pedestrian detection via classification on riemannian manifolds*, In IEEE Transactions on Pattern Analysis and Machine Intelligence, page 1713–1727, 2008.

[18] X. Wang, T. Han, and S. Yan. *An HOG-LBP human detector with partial occlusion handling*, In Proceedings of the 12th IEEE International Conference on Computer Vision, Kyoto, Japan. 2009.

[19] D. G. Lowe. *Distinctive image features from scale-invariant keypoints*, In International Journal of Computer Vision, 60(2):91–110, 2004.

[20] N. Dalal. *Finding People in Images and Videos*, Ph.D. thesis, Institut Polytechnique de Grenoble, 2006.

[21] X. Wang, X. Zhou, T. X. Han, S. Tang, G. Chen, K. Yu, and T. S. Huang. *Liblinear SVM with HOG-LBP and bag of words (DHOG) features*, VOC Object Detection Challenge, 2010.

[22] P. Felzenszwalb, R. Girshick, and D. McAllester. *Cascade object detection with de- formable part models*, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, USA, pages 2241–2248. IEEE, 2010.

[23] TensorFlow, `https://www.tensorflow.org`.

[24] TensorFlow Object Detection API, `https://github.com/tensorflow/models/tree/master/object_detection`.

[25] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei,*ImageNet Large Scale Visual Recognition Challenge*, IJCV, 2015.

[26] A Krizhevsky, I Sutskever, GE Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in neural information processing systems, 1097-1105, 2012.

[27] Girshick, Ross and Donahue, Jeff and Darrell, Trevor and Malik, Jitendra, *Rich feature hierarchies for accurate object detection and semantic segmentation*, In proceedings of Computer Vision and Pattern Recognition, 2014.

[28] R. Girshick. *Fast R-CNN*, arXiv:1504.08083, 2015.

[29] S Ren, K He, R Girshick, J Sun, *Faster R-CNN: Towards real-time object detection with region proposal networks*, Advances in neural information processing systems, 2015.

[30] Redmon, Joseph and Farhadi, Ali, *YOLO9000: Better, Faster, Stronger*, arXiv:1612.08242, 2016.

[31] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, *SSD: Single Shot MultiBox Detector*, arXiv:1512.02325v5 [cs.CV], 2016.

[32] Jonathan Huang et al, *Speed/accuracy trade-offs for modern convolutional object detectors*, arXiv:1611.10012v3 [cs.CV], 2017.

[33] PhD Thesis: `https://tel.archives-ouvertes.fr/tel-00722632/document`.

[34] Andrew G. Howard et al, *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*, arXiv:1704.04861v1 [cs.CV], 2017.

[35] TensorFlow Mobile, `https://www.tensorflow.org/mobile/`.

[36] TensorFlow Mobile, `https://www.android.com/`.

[37] Raspberry Pi, `https://www.raspberrypi.org/`.