Real-time Animal Detection System for Intelligent Vehicles

by

Depu Zhou

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Animal and Vehicle Collisions (AVCs) have been a growing concern in North America since the abundant wildlife resources and increases of automobiles. Such problems cause hundreds of people deaths, thousands of human injuries, billions of dollars in property damage and countless of wildlife deaths every year. To address these challenges, smart cars have to be equipped with Advanced Driver Assistance Systems (ADAS) able to detect dangerous animals (e.g., moose, elk and cow), which cross the road, and warn the driver about the imminent accident. In this thesis, we explore the performance of different image features and classification algorithms in animal detection application, and design a real-time animal detection system following three criteria: detection accuracy, detection time and system energy consumption. In order to pursue high detection rate but low time and energy consumption, a double-stage detection system is proposed. In the first stage, we use the LBP adopting AdaBoost algorithm which provides the next stage by a set of region of interests containing target animals and other false positive targets. Afterward, the second stage rejects the false positive ROIs by two HOG-SVM based sub-classifiers. To build and evaluate the animal detector, we create our own database, which will be updated by adding new samples. Through an extensive set of evaluations, we note that the double-stage system is able to detect about 85% of target animals.

Acknowledgements

First of all I would like to offer my heartfelt gratitude to my thesis supervisor, **Professor Azzedine Boukerche**, who has supported me throughout my Master thesis with his kind heart, encouragement and knowledge. I also offer my sincere appreciation for his financial support throughout my research work. He is always a nice person who not only a helpful supervisor but also a best friend of my study life in the University of Ottawa, and I appreciate of his support again.

As a member of Mobile Vision Group in PARADISE laboratory, I have been extremely lucky to be aided by **Dr. Abdelhamid Mammeri**, a fine tutor and friend. He brought me to the wonderful and attractive Image Processing & Artificial Intelligence research area. He gave me useful guidance and great support in my whole research and responded to my questions and queries so promptly.

I would like to thank all **PARADISE** researchers, especially **Mobile Vision Group** members. They provided so many inspiration, help, support and friendship during these 2 years. I am indebted to them for their help.

Finally, I thank my parents for supporting me throughout all my studies at University of Ottawa.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and motivation

When referring to the Advanced Driver Assistant System (ADAS), the most well-known image processing applications are traffic sign recognition and pedestrian detection. Collisions between motor vehicles and large animals, such as moose, deer, and bear have been an increasingly serious and challenging problem since the invention of automobiles.

As we know, North American countries have abundant wildlife resources and a great number of highways along with forests, therefore the possibilities of animal-vehicle collisions (AVCs) is higher than other areas in the world. In turn, it causes hundreds of human deaths, thousands of injuries, billions of dollars in property damage and countless of wild animal deaths every year. An official report from Transport Canada [50] shows approximately between 4 and 8 large animal-vehicle collisions take place every hour in Canada. In the US, there are over 35,000 AVCs happening every year results in about 200 deaths and close to 4,000 property damages with the lose of over $1,000 in each case [10]. For example, the total annual loss of such collisions in a particular section of Highway 3 (50 kilometers) in southern Alberta was estimated to $\$909,000$. Similar situations are arising in Europe, Africa and Asia. For instance, before 1996 some of the European countries experienced more than $507,000$ collisions causing around 300 humans fatalities,

30,000 human injuries, and more than 1 billion dollars in damage every year [26], [44], and [16].

To cope with this problem, modern vehicles are managed to be equipped with smart driver assistance systems which are capable to detect dangerous animals which are crossing roadways or appear in front of vehicles, and warn drivers about the imminent danger. Moreover, this system can also function as roadside devices which are installed at the road borders around areas of interests, and wireless communication technology can help us transfer the warning information from the roadside devices to the passing drivers. In addition, With the development of artificial intelligence technology, unmanned vehicles (robot driving) might replace conventional cars, become a new form of transportation in future. This kind of animal detection system is indispensable for safe driving.

1.2 Outline of the thesis

This thesis is structured as follows.

In chapter 2, the literature review of different techniques in AVCs mitigation is listed.

In Chapter 3, we present the introduction of some basic knowledge of three main image descriptors which are widely used in objects detection technology. These descriptors are Haar-like features, Hog (Histogram of Oriented Gradients) features and LBP (Local binary pattern) features.

After that, two classification algorithms, AdaBoost and SVM (Support Vector Machine) are presented separately in Chapter 4. These two kind of classification methods help us to distinguish the positive targets from the negative samples.

Then, we present in Chapter 5 how to build our database which is used to train and test our final system. The created large animal database are mainly collected from Internet since no such public database exists. This database will be frequently updated by adding more new images.

Chapter 6 is the main part of this thesis. We create four different types classifiers

based on different image features and classification algorithms. Some results of evaluated experiments are illustrated, including detection accuracy and detection speed. Subsequently, we introduce the architecture design of double-stage animal detection system whose design is based on the experimental results. At the end of this chapter, we give the final evaluation results to show our system has a relatively excellent performance.

We present the energy consumption test in Chapter 7. We design a method to evaluate the energy consumption of different single classifiers. Initially, we transplant the four classifiers from PC platform to Android devices (smart phone). Then, we use the rear-camera of the device to detect animals from an input video. At last, an energy consumption software help us to record the energy consumption for each classifiers.

Finally, Chapter 8 includes the conclusion and the future work.

Chapter 2

Related Works

There are so many AVCs mitigation solutions have been proposed, these solutions range from the use of a simple traffic sign, fencing and reflectors, to a more refined methods such as intelligent cameras based on image processing technology. The present AVCs mitigation solutions are grouped into two main categories: passive methods (animals deterrence) which focus on keeping the dangerous animals away from the road, and active methods based on animals detection. For each category, these solutions are further divided into on-vehicle and roadside methods, see Table 2.1 [27].

2.1 Passive methods based on deterrence

The core idea of passive deterrence methods is to keep the large animals away from dangerous roadways (or prevent them from crossing the dangerous roads). We call these as passive methods, since they do not see around roads, and they activate a signal (alarm or warning) when animals are detected. These techniques can be grouped into roadside methods and on-vehicle methods, according to where the detection systems are quipped.

<table>
<tr><td>Detection method</td><td>Location</td><td colspan="2">Type of detection/warning</td></tr>
<tr><td rowspan="4">Passive deterrence method</td><td rowspan="2">Roadside</td><td>Conventional tools</td><td>Examples: reflectors, electronic mats, underpass and overpass construction</td></tr>
<tr><td>Prevention techniques</td><td>Examples: Removal of roadside vegetation</td></tr>
<tr><td rowspan="2">On-vehicle</td><td colspan="2">Audible warning such as whistles</td></tr>
<tr><td colspan="2">Visible warning such high-intensity discharge</td></tr>
<tr><td rowspan="4">Active detection method</td><td rowspan="3">Roadside</td><td colspan="2">Break the beam systems such as infrared sensors, ultrasonic sensors, laser or microwave radio</td></tr>
<tr><td rowspan="2">Area-cover systems</td><td>Passive using cameras</td></tr>
<tr><td>Active such microwave radar</td></tr>
<tr><td>On-vehicle</td><td colspan="2">Cameras: VS, IR, TIR</td></tr>
</table>

Table 2.1: Methods used to mitigate AVCs [27]

2.1.1 Roadside technique

The Passive roadside deterrence techniques are very traditional. They include conventional tools such as electronic mats, whistles, repellents, animal reflectors, and roadside reflectors (see Figure 2.1) [22]; and prevention techniques including removal of roadside vegetation and feeding interception are also used to keep away large animals from roadways. Figure 2.2 shows several examples of roadway crossings such as overpasses (i.e., bridges) and underpasses (e.g., culverts and tunnels). These structures keep the animals away from the dangerous road and help them pass the road in a safe way. It is shown that passive roadside methods are proved to be inefficient to reduce AVCs.

Figure 2.1: Example of roadside animal reflector [22].

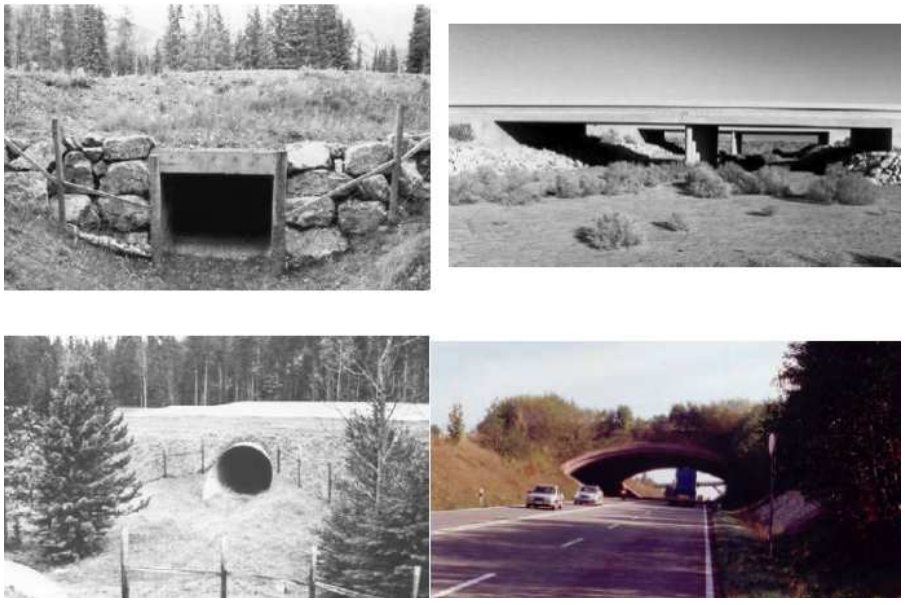


Figure 2.2: Example of underpasses and overpasses [22].

2.1.2 On-Vehicle technique

The primary effect of on-vehicle deterrence technology is to warn animals through animal-audible or animal-visible single emanated by specialized devices. Audible warning consists in generating an ultrasonic noise (mainly in the range of 2-6 KHz), which is inaudible to humans, to alert deer to the approach of a vehicle, see Figure 2.3 [45]. A typical example of these devices is whistle which produces noise as air passes through it. When vehicles exceed certain speed, whistles emit noise that can be heard a few hundreds of meters. We note the existence of some commercial whistles devices, which can be attached to the front of vehicles. For example, the Hornet V120, described in [45], which produces a warning signal to alert animals that cross roads. However, it is reported in the literature that some animals, such as deer, often do not respond to whistles, and the noise is often attenuated by roadway curvature, trees, and other objects.

Figure 2.3: An example of Ultrasonic animal repellent (right image) which can be equipped on vehicle.

Visible warning is another method to deterrent animals which performed by producing high intensity lights by vehicles. This method increases the distance of perceiving approaching vehicles by animals, which give them more time to leave away from the danger. In a recent study [3], it is observed that the combination of standard tungsten-

halogen lighting and different pulse frequencies of Xenarc high-intensity discharge (HID) lighting increases the reaction of animals to oncoming vehicles. Even the increase of the reaction distance by animals to up to 20 meters using HID technology, audible-in-vehicle deterrence systems have not yet proven their efficiency in AVCs mitigation.

2.2 Active techniques based on detection

A more effective strategy to mitigate the AVCs is based on animal detection rather than animal deterrence. We also introduce the Animal detection methods on two categories, based on whether the detection system is equipped on roadside or inside vehicles.

2.2.1 Roadside systems

Active roadside animal detection methods are usually used in conjunction with flashing signs installed on some strategic roadways, where the possibility of road crossing is relatively high. Regarding the technology used in detection, roadside techniques are classified into two categories: break-the-beam method (BTB) and area-cover method [14].

Break the beam systems: A beam is a signal formed between an emitter and a receiver. If the beam is interrupted or reduced, a flashing system issues a warning message to drivers. The beam can be formed using infrared signals, ultrasonic signals, lasers, or using microwave radio. Many traditional systems have adapted this technology to reduce AVCs, see [14]. Figure 2.4 shows a dynamic animal sign and sensor system which works with radar beam sensor device connected to amber lights. The main challenge of these systems is that they require a clear line of sight to establish beam connections. Another problem is related to their activation by non-large animals, such that small species, air, or humans, resulting in false alarms. The advantage of these systems is that they are relatively insensitive to changes in temperature.

Area-cover systems: Area-cover systems detect animals within the detection range

Figure 2.4: Roadside dynamic animal sensor system example (Photo courtesy of the Western Transportation Institute) [22]

covered by the sensors in use, see Figure 2.5. Area-cover systems can be passive or active, depending on the deployed sensors. Passive sensors are based on infrared or conventional digital video cameras, and receive a signal upon the detection of animals. The cameras, then, notifies drivers through flashing systems. Active sensors such as microwave radars send a signal over the region of interest and measure its reflection. Examples of such applications are given in [15], such as the system in Kootenay, British Columbia, Canada and the system in Box, Finland, Europe. Compared to BTB methods, area-cover systems are more accurate and robust.

2.2.2 On-Vehicle systems

In this category, see Figure 2.6, we regroup all on-vehicle cameras based systems including thermographic, infrared or conventional camera video. We focus our interest in this paper on camera-based systems since they are the most efficient and accurate way to see

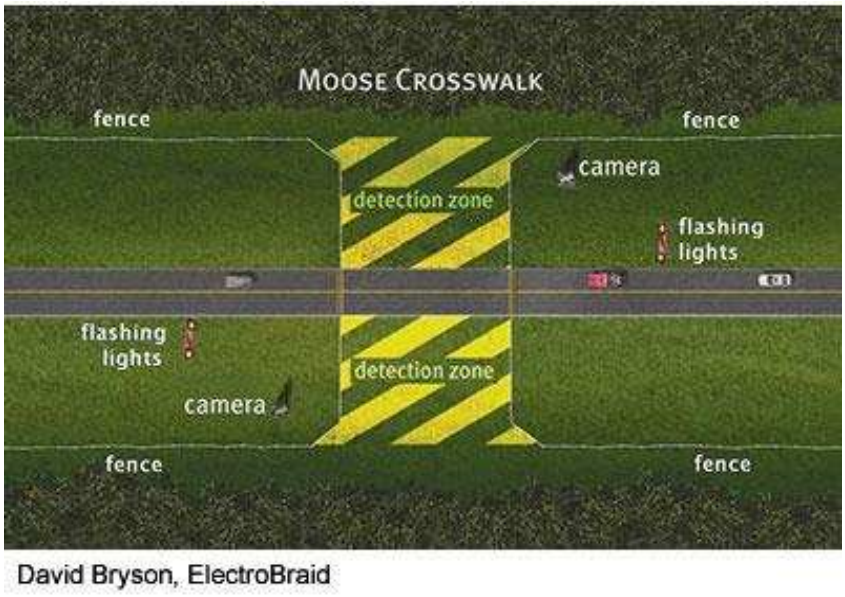


Figure 2.5: Area-cover system. On both side of the road are an animal detection area including detection camera, flashing light, and wildlife exclusion fencing [12].

around the regions under investigation, and to reduce AVCs. On the other hand, the disadvantages of camera based systems lies on the fact they focus on animals within the road, and ignore those outside. Also, these systems fail to detect animals on curve-lanes.

2.3 Camera based systems: detection and recognition stages

To efficiently detect road-animals before their collisions with vehicles, camera-based systems seem to be the best option, compared to the aforementioned solutions [22]. For this purpose, we present in this section the main idea behind the detection and recognition of animals either by roadside cameras or in-vehicle cameras.

Animal detection, especially in nighttime, is an extremely challenging problem due to the surrounding conditions such as illumination changes and cluttered background; and,

Figure 2.6: On-vehicle animal detection system. Camera captures the dangerous animals which are crossing the road, and the system recognizes the target and warns drivers about the imminent danger.

on the other hand, the large intra-class variability between different types of animals, and between animals of the same category. Surprisingly, animal detection systems for collision mitigation have not received high interest by computer-vision community. Even the existence of some camera-based systems for AVC mitigation, almost all of them do not detail their stages of detection and recognition. This does not help to improve the detection accuracy and, hence, do not protect drivers from collisions with animals. In this section we review some schemes related to the detection and recognition of animal in, which might help to design animal detection and recognition schemes. We focus only on algorithms applied on large animal, while research-works dedicated to small animals like bird are not assessed since they are not considered as a danger for vehicles drivers. We summarize in Table 2.3 the methods found in both detection and recognition stages.

Stage	Techniques		Examples
Detection	Texture	Example: LBP, Haar, PCA	[39] [5] [28] [20]
	Shape	Example: HOG	[28] [57] [58]
	Color		[55]
Recognition	AI and ML techniques SVM, NN, AdaBoost		[28] [57] [5]

Table 2.2: Some techniques used by detection and recognition stages

2.3.1 Detection

A natural way to detect animals using cameras is through one of the existing detection schemes, especially those applied for pedestrian detection such as detection through texture features, color features or gradient features (which are robust to illumination changes and hence more applicable to nighttime conditions). Unfortunately, directly applying one of these schemes, developed initially to detect other object such as pedestrian or TS, has apparent difficulties. This is due to the fact that animals have their specific properties including height, shape, texture and different views (rear-view, front view and side-view), which distinguish them from other objects, such as pedestrian. That is, an adaptation of the current detection schemes is mandatory. In what follows, we review algorithms used to detect animals through textures, gradient or color features.

Texture features based detection. Some texture features such as Haar-like features, Local Binary Patterns (LBP) features, and Principal Component Analysis (PCA) are widely used in object detection.

Historically, one of the most famous and pioneering works to human-face detection is based on Haar-like features adopts a kind of cascade AdaBoost classifier [52]. The authors also present a very fast method to calculate the Haar features called integral image.

Another algorithm that explores Haar features to detect and track lion faces scenario

is presented in [5]. An adapted version of Haar features and AdaBoost classification algorithm is investigated. But they only focus on animal faces instead of the animal body. Figure 2.7 lists some examples of lion face detection. In addition, a combination of the Kanade-Lucas-TOmasi method and a specific interest model applied to the detected face, is used to track the animal faces. This allows achieves reliable detection and temporally smooth tracking of animal faces.

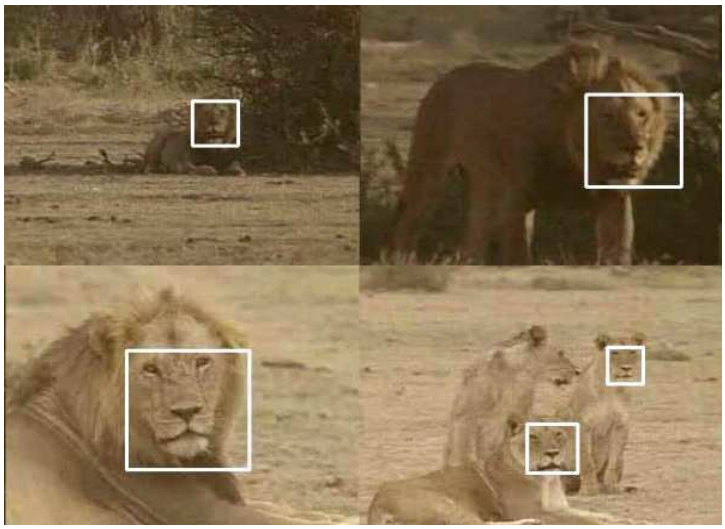


Figure 2.7: Examples of lion face detections: images show scale invariance, slight posture invariance and multiple detections of the system [5].

The next technique used in object detection concerns LBP features. The authors in [56] presents a novel and efficient model to achieve human face detection based on LBP features. They extend the traditional pixel LBP features to multi-block regions (MB-LBP) and apply it in an adapted AdaBoost algorithm. Their experiment result shows the MB-LBP based human face classifier is more powerful than Haar-like classifiers.

Gradient features based detection. Gradient features, such as Scale-invariant Feature Transform (SIFT) or Histogram of Oriented Gradient (HOG), can describe the object's edge and contour appropriately. One major breakthrough in object detection is occurred in a gradient feature domain, where a HOG detection strategy is presented in

[8]. The detection method starts by converting the input image into grayscale version, which is then divided into a set of blocks and cells, and for each cell a histogram of gradient orientation is performed. HOG adopt SVM algorithm has been successfully applied to detect pedestrian, traffic sign and animals.

The authors design a deer detection device (see Figure 2.8) based on thermographic cameras. They apply a new Contour-Based HOG algorithm (CNT-HOG) to detect the deer from the thermal images. This device has a excellent performance in nighttime deer detection [58], as shown in Figure 2.9.

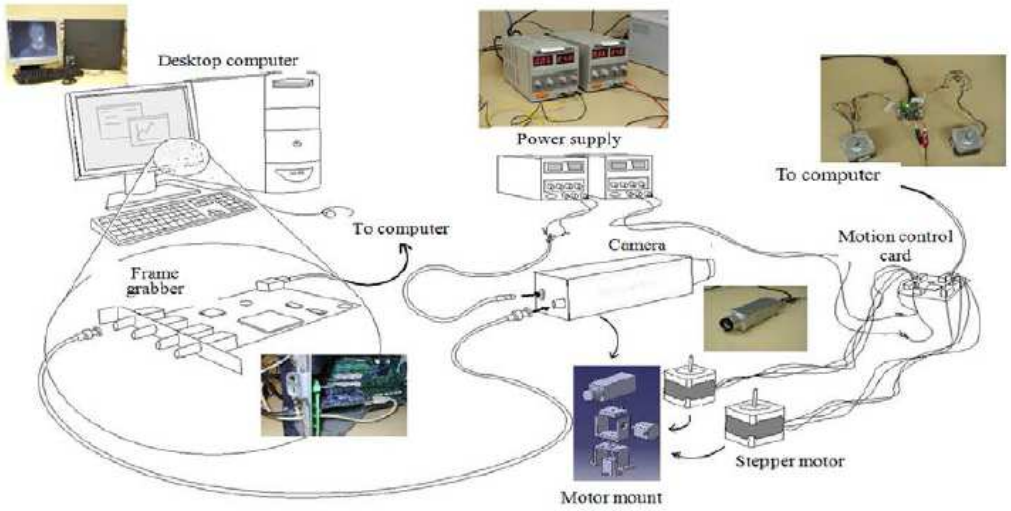


Figure 2.8: Connection of components in infrared camera based deer detection system [58].

Colour based detection. Besides texture and shape features, color descriptor can be used to detect animals especially in day time. Color extraction can be performed using segmentation methods, which allows to intensify certain colors, and to ignore unsuitable regions or noise. This step is preferably headed by the selection of a suitable color space such as RGB, Luv, HSI or HSV.

For instance, the authors in [55] start by preprocessing the input images to reduce the processed amount of data. They suggest to use Luv color space; and then, apply mean-

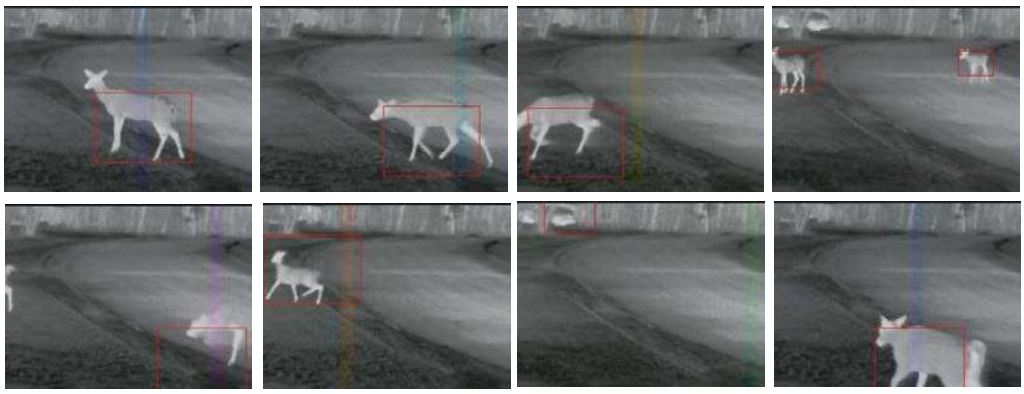


Figure 2.9: Results of deer detection system based on infrared camera. [58].

shift clustering algorithm on the preprocessed images to perform color segmentation. Their approach learns a color model of elephants from training images. Unfortunately, this method is applied only on daytime conditions, and seems not adapted to night conditions. Another colour based detection is based on background subtraction method [38]. The process of this method can be divided into two steps, background extraction and colour based moving object detection. But this method only focuses on stable background and moving target.

Table 2.3.1 gives the summary of active animal detection system based on image processing technology.

2.3.2 Recognition

The recognition stage receives a list of ROIs that possibly contains one or more animals. In this stage, ROIs are classified as animal or non-animal, with minimal false positives and false negatives. Animal recognition is usually performed using template matching or machine learning techniques such as Neural Network (NN), Support Vector Machine (SVM) and AdaBoost.The choice of the suitable recognition algorithm depends essentially on the training sample statistics (e.g., class distribution and size), the features used and the output of the detection algorithm.

Year	Ref	Technique	Advantages	Disadvantages
2006	[5]	Haar-like features based on AdaBoost and image feature based tracking	-Real-time -Smooth and accurate -tracking included	-Some false positive -Only focus on face detection
2009	[38]	Background subtraction method after getting the background image	-Very fast -can detect any kind of animals	-The background must be stable -Cannot work as on-vehicle system
2011	[57]	Haar of oriented gradient	-Various animal head (cat, fox, panda, wolf, etc.)	-Slow -Only front face
2012	[58]	Thermal camera and GNT+HOG	-Fast to get ROIs -High detection rate -plenty of deer postures included	-Only deer detection -cannot work in strong light intensity environment -misidentification (car, human)
2013	[28]	2-stage: LBP+AdaBoost and HOG+SVM trained by separate databases	-Real-time -Variety of animals -Low false positive rate -Different weather consitions	-Only consider two types animal postures

Table 2.3: The overview of active animal detection based on image processing.

Chapter 3

Image Features Extraction

Features are functions of the original measurement variables that are useful for classification and/or pattern recognition. **Features extraction** is the process of defining a set of features, or image characteristics, which will most efficiently or meaningfully represent the information that is important for detection analysis and classification [30]. The purpose of features extraction is to enhance the effectiveness and efficiency of object detection and classification.

Various feature extraction methods focus on object detection. The remaining content of this chapter will discuss three different types of simple features (or descriptions),which are **Haar-like features**, **HOG** (Histogram of Oriented Gradient) and **LBP** (Local binary patterns). All of the three popular features are widely used in object detection or target tracking like human-face (eyes), pedestrian, traffic signs, character recognition, etc.

3.1 Haar-like features

Haar-like features (or Haar features) are traditional digital image features used in object recognition. They are an over complete set of two-dimensional Haar functions, which can be used to encode local appearance of objects[36]. There are two kinds of original

Haar features which are introduced by Papageogiou et al. shown in Figure 3.1 (a) and (b)[36]. In features (a) of Figure 3.1, the two rectangles which are left-right, can have also the position "up-down". Figure 3.1 (c) shows the extended Haar-like feature with three rectangles enriched by Viola and Jones in 2001 [52]. Figures (d-f) illustrate three kinds of Leinhardt's rotated features. From these six types of features, we can find that each Haar-like feature consists of at least two jointed "black" and "white" rectangles. The value of a Haar-like feature is computed as the difference between the sum of the pixels gray level values within the "black" and "white" rectangular regions:

$$f(x) = \sum_{black\,rectangle} (pixel\,value) - \sum_{white\,rectangle} (pixel\,value)$$

If three rectangles appear in one Haar feature which shown in Figure 3.1 (c) or (e), de-

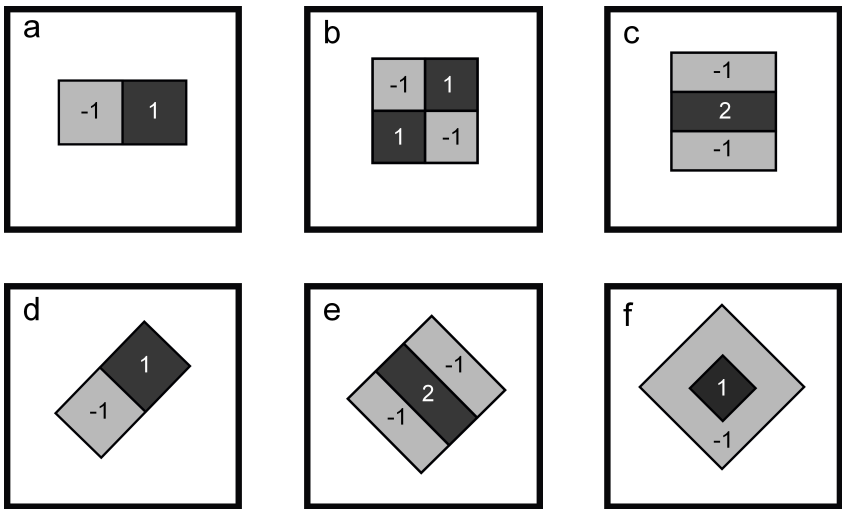


Figure 3.1: Example Haar features shown in detection window

fault integer weights need to be assigned for each rectangles to ensure an equal number of pixels in "black" and "white" regions[37]. The default weights assigned in Figure 3.1 (c) and (e) are -1,2 and -1.

3.1.1 Integral Image

The exhaustive set of the basic Haar features (a-c in Figure 3.1) is extremely large. For instance, there are more than $45,000$ (a-c) types Haar features in a 24×24 sub-window. A convenient and fast way need to be applied in calculate the huge number of features. Integral image (also know as a summed area table) is a new image representation which allows the features used by the detector to be computed very quickly. In computer vision, it was first prominently used within the Viola and Jones robust real-time object detection in 2002 [52]. The most significant advantage for this smart representation is any one of Haar-like features which can be computed at any location and scale in constant but short time.

The following equation gives the algorithm of integral image. As shown in Figure 3.2, the integral image at any point(assume at (x, y)) contains the sum of the pixels above and to the left of x,y, inclusive:

$$P(x, y) = \sum_{x' \leq x,\, y' \leq y} i(x', y')$$

Where $P(x, y)$ and i(x,y) are the integral image value and pixel value in point (x, y)

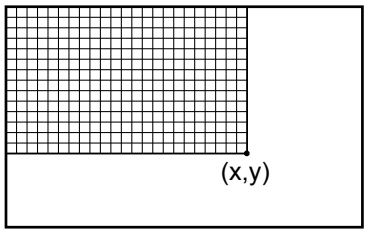


Figure 3.2: Point value in integral image is the sum of all pixels in top-left rectangle

respectively. Using the following two iterations:

$$S(x, y) = S(x, y - 1) + i(x, y)$$

$$P(x, y) = P(x - 1, y) + S(x, y)$$

Where $S(x, y)$ is the x row's cumulative sum, $S(x, -1) = 0$, and $P(-1, y) = 0$, the integral image can be obtained by passing over the original image only one time [52].

3.1.2 Haar feature computation

With the help of integral image, any rectangular's pixels value sum in Haar-like features can be computed in four array references (see Figur 3.3). The integral image value at

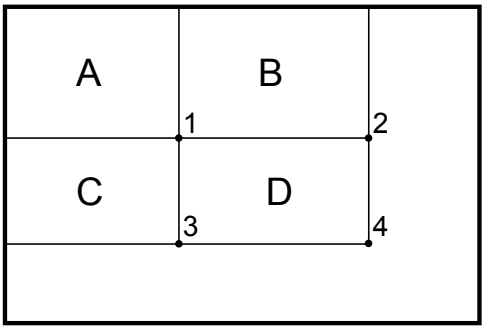


Figure 3.3: Compute rectangular sum using integral image

point 1 is the sum of the pixels value in rectangle A, the value at point 2 is $A + B$, at point 3 is $A + C$, and at point 4 is $A + B + C + D$. The rectangle sum in region D can be computed as follow:

$$P_1 = A, P_2 = A + B, P_3 = A + C, P_4 = A + B + C + D$$

$$D = (A + B + C + D) - (A + B) - (A + C) + A = P_4 + P_1 - P_2 - P_3$$

Therefore, the Haar feature appears in Figure 3.1 (a) can be computed in six array references since it contains two adjacent rectangles, feature (b) and (c) need nine and eight references respectively.

A different method called rotated integral image (adapting of the integral image representation) is applied to deal with 45° oriented rectangular features (in Figure 3.1 (d),(e) and (f)). This method was proposed by Lienhart and Maydt in 2002, they termed the rotated summed area table, denoted $rsat(x, y)$, which makes the rotated rectangles can be computed easily, like vertical and horizontal rectangles, with four references [23]. The *rsat* data structure contains the sum of pixels of the rectangle rotated by 45° with rightmost corner located at (x, y) and extending to the image boundaries. A clearer

expression shows in Figure 3.4 (a).

$$rsat(x, y) = \sum_{x' \leq x,\, x' \leq x - |y - y'|} i(x', y')$$

Figure 3.4 (b) gives the same way(in integral image) to computed region A in rotated integral image:

$$A = L_4 + L_1 - L_2 - L_3$$

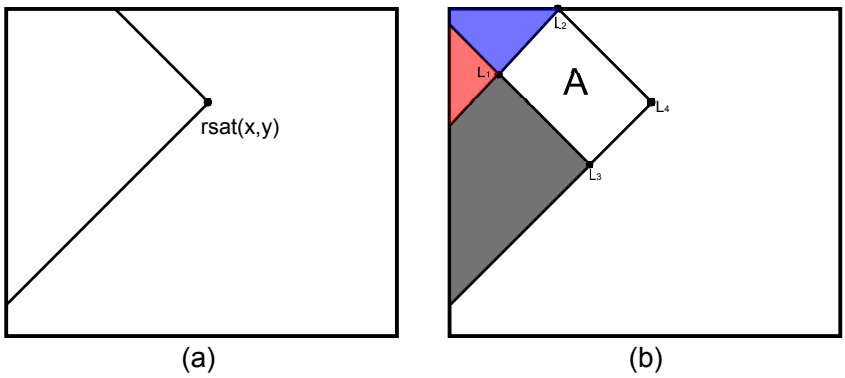


Figure 3.4: Rotated integral image: Lienhart & Maydts' *rsat* representation

3.2 Hisgogram of Oriented Gradients (HOG) features

Histogram of Oriented Gradients (HOG) are another popular image descriptors used in image processing for the purpose of target detection. It is proposed by Dalal and Triggs who focused their algorithm on the problem of pedestrian detection, and had an excellent performance compared with other feature sets [8]. The advantage of HOG descriptor is that it can describe contour and edge feature outstandingly in objects other than human beings, such as bicycles, and cars, as well as common animals such as dogs, deer. Figure 3.5 indicates the overview of HOG features extraction and target detection process based on different classification algorithm, SVM and AdaBoost.

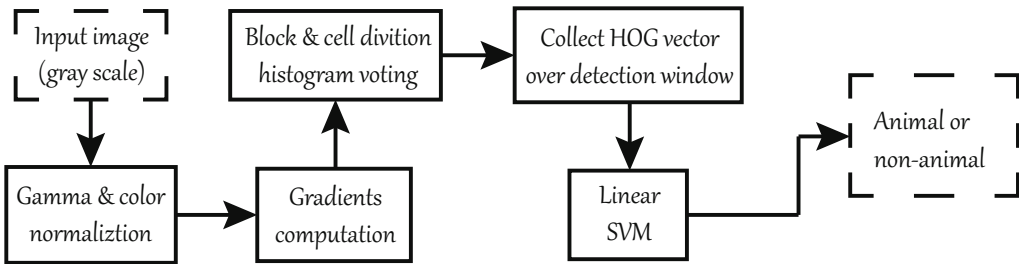


Figure 3.5: Process of HOG features extraction and animal detection adopt SVM algorithm.

3.2.1 Gradient computation

Before the computation of the gradient values, gamma/colour normalization would be applied to improve the performance [8]. After that, the gradient direction and gradient magnitude of each pixel would be calculated. The horizontal and vertical gradient obtained by convolution the simple but best 1-D gradient operator:

- horizontal operator:$[-1, 0, 1]$
- vertical operator:$[-1, 0, 1]^T$

which means, the gradient of pixel (x, y) is:

$$G_x(x, y) = H(x + 1, y) - H(x - 1, y)$$

$$G_y(x, y) = H(x, y + 1) - H(x, y - 1)$$

where, $G_x(x, y)$ and $G_y(x, y)$ are the horizontal gradient and vertical gradient of point (x, y) respectively, $H(x, y)$ is the pixel gray value. Then, the gradient magnitude $G(x, y)$ and gradient direction $\alpha(x, y)$ can be obtained as:

$$G(x, y) = \sqrt{G_x(x, y)^2 + G_y(x, y)^2}$$

$$\alpha(x, y) = \tan^{-1}[\frac{G_y(x, y)}{G_x(x, y)}]$$

3.2.2 Cell histograms and HOG feature vector generation

The next step after gradient computation is to create the cell histograms. The image window was divided into several larger spatial regions called "blocks" which contain small spatial regions called "cells", for each cell accumulating a local 1-d histogram of gradient directions voted by the pixel's gradient direction and magnitude in this cell. The histogram channels are spread over $0^\circ \sim 180^\circ$ if the gradient is "unsigned" or $0^\circ \sim 360^\circ$ if the gradient is "signed". Normally, increasing the orientation bins would improve the performance until the bins number increase to 9. The pixel's gradient magnitude is contributed as the vote weight. Then, we get a 9-dimensional feature vector for each cell.

L1-norm, L1-sqrt, L2-norm, and L2-hys can be applied to normalize the gradient intensity to make the feature vector space robust to local illumination changes:

$$\text{L1-norm: } V = \frac{v}{(\| v \|_1 + e)}$$

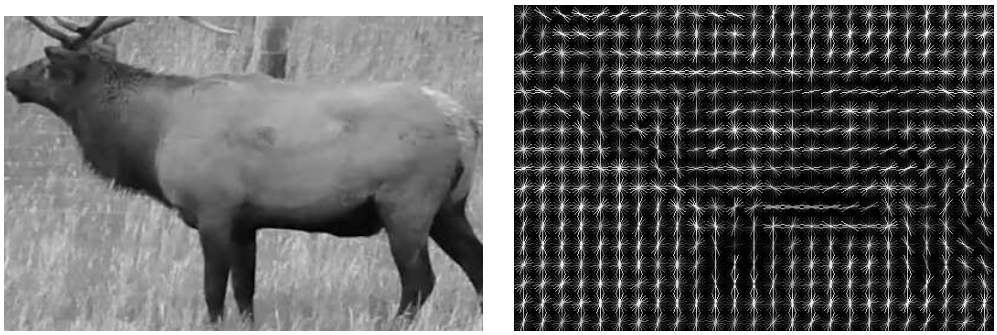
$$\text{L1-sqrt: } V = \sqrt{\frac{v}{(\| v \|_1 + e)}}$$

$$\text{L2-norm: } V = \frac{v}{\sqrt{\| v \|_2^2 + e^2}}$$

$$\text{L2-hys: } V = \frac{v}{\sqrt{\| v \|_2^2 + e^2}}, v \leq 0.2$$

where v is the non-normalized vector containing all histograms and e is a small constant (unimportant). Note that we limited the maximum value of v to 0.2 to improve the performance [24]. Figure 3.6 illustrates an initial moose image and its corresponding HOG image. We can clearly see the outline of the moose from the HOG image.

Assume there are N blocks in a image window and each block contains 4 cells divided into 9 bins, the final HOG feature vectors should be $N \times 4 \times 9 = 36N$ dimensions. After we extract the HOG feature vectors, a classification algorithm would be applied to detect the target. The SVM algorithm was specifically introduced by Dalal. The AdaBoost algorithm will be introduced in the next chapter.



(a) Original image (b) HOG image

Figure 3.6: Initial image and HOG image for a moose. Each " $*$ " presents a cell with 9 bins and the luminance of each direction vector indicates the magnitude of this bin.

3.3 Local Binary Patterns (LBP) features

Local Binary Patterns (LBP) introduced as another powerful local descriptor for micro-structures of images in computer vision [33]. In general, it is an efficient image feature which transforms an image into an array or image of integer labels describing small-scale appearances of the images. These labels or statistic histogram, are then applied for further image analysis. Currently, LBP feature is widely used in texture analysis, target detecting and tracking, face recognition analysis, product quality analysis, surface inspection, etc. In this section, We will introduce several version of LBP feature.

3.3.1 Basic LBP operator

The original local binary pattern feature was proposed by Ojala et al. in 1995 [33]. It only works on a 3×3 pixel block of a gray image. As show in Figure 3.7, for each 3×3 block, basic LBP can be defined as an ordered set of pixel intensities binary comparisons between the centre pixel and its eight neighbour pixels. Where the centre pixel's value is greater than the neighbor's value, label "1". Otherwise, label "0". These eight neighbour pixels labeled by a circle order (i.e. clockwise or counter-clockwise), start from top-left pixel if clockwise, or left pixel if counter-clockwise. The binary code is usually converted

to decimal for convenience. In concrete terms, the basic LBP operator can be obtained

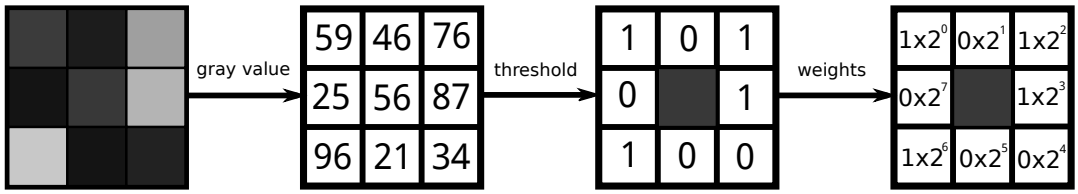


Figure 3.7: Basic LBP operator, LBP binary code:01001101, LBP decimal code:$1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 + 0 \times 2^4 + 0 \times 2^5 + 1 \times 2^6 + 0 \times 2^7 = 77$

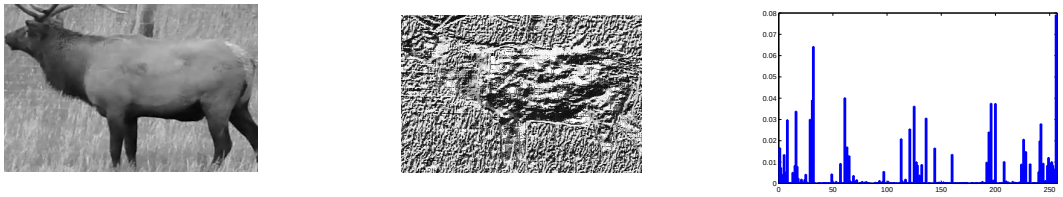
as follows:

$$s(f_p - f_c) = \begin{cases} 1 & \text{if} \quad f_p \geq f_c \\ 0 & \text{otherwise} \end{cases}$$

where f_c is the centre pixel's gray value, $f_p(p = 0, ..., 8)$ are the neighborhoods'. Then, by assigning a binomial weight 2^p for each $s(f_p - f_c)$, the LBP is computed as follows:

$$LBP = \sum_{p=0}^{7} s(f_p - f_c)2^p$$

Since the neighborhood only consists of eight pixels, a total of $2^8 = 256$ different kinds of codes can be obtained from "0" to "255" if code indicated by decimal code. A 256-bin local binary pattern histogram can be obtained by accumulating the frequency of each "number" occurring for all LBP code over an image. Figure 3.8 illustrates an example of LBP image and histogram for an original moose image.



(a) Original image (b) LBP image (c) LBP histogram

Figure 3.8: Moose original image, the corresponding LBP image and histogram

3.3.2 Generic LBP operator

A few years after its original publication, Ojala et al. extended their basic LBP operator to a more generic version[34, 25]. Comparing with the original version of LBP operator just works in a 3×3 pixel block, the generic LBP operator uses neighborhoods of different sizes, to capture dominant features at different scales. In addition, generic LBP was extended from the rectangular domain to circular domain, which allow us choose arbitrary number of neighborhoods in R distance.

The advanced LBP operator also works in gray image, as shown in the following equations, assume f_c IS the gray value of an arbitrary pixel (x, y) which selected as a center pixel in an image $I(x, y)$. And f_p denote the gray levels of a set of sampling points around f_c with the same frequency and radius R, where P is the number of sampling points.

$$\begin{aligned} f_c &= I(x, y), \\ f_p &= I(x_p, y_p), \\ x_p &= x + Rcos(2\pi p/P), \qquad \text{and} \\ y_p &= y - Rsin(2\pi p/P). \end{aligned}$$

where f_c is the centre pixel, and $p = 0, ..., P - 1$.

Considering the coordinates of pixels should be integer, a bilinearly interpolated method was applied when the coordinates of neighborhood pixel
$f_p((x + Rcos(2\pi p/P), y - Rsin(2\pi p/P))$ are non-integer.
Assuming a non-integer-coordinate point $(x_p, y_p) = (i + u, j + v)$, where i, j are positive integers, and u, v are floating numbers between [0.1). The pixel value $f(i + u, j + v)$ can be determined by the surrounding four integer points $(i, j), (i+1, j), (i, j+1), (i+1, j+1)$:

$$\begin{aligned} f(i + u, j + v) =& (1 - u)(1 - v)f(i, j) + (1 - u)vf(i, j + 1) \\ &+ u(1 - v)f(i + 1, j) + uvf(i + 1, j + 1) \end{aligned}$$

Then, the generic local binary pattern operator $LBP_{P,R}$ is defined as

$$LBP_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} s(f_p - f_c)2^p \tag{3.1}$$

Eq.3.1 illustrates that the signs of the differences in a neighborhood are interpreted as a $P - bit$ binary number, which can generate 2^P different values for the LBP code. The binary code is usually converted to decimal for convenience. We can obtain the image's texture distribution after all pixels' LBP_P, R operator value are calculated. Shown as:

$$T \approx t(LBP_{P,R}(x_c, y_c)) \tag{3.2}$$

The Figure 3.9 indicates examples of generic LBP operators with different sampling points and radius. Some widely used generic LBP operators are $LBP_{8,1}, LBP_{8,2}, LBP_{16,2}$ and $LBP_{24,3}$ etc. Figure 3.10 shows three different LBP operators and the corresponding LBP images. The basic LBP (Figure 3.7) is quite similar to $LBP_{8,1}$ except two

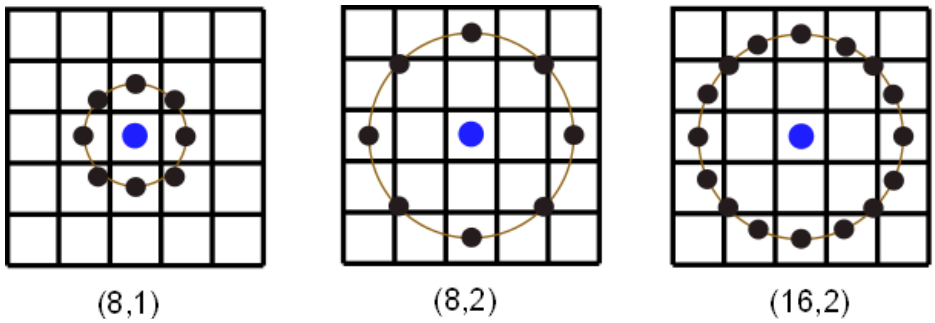


Figure 3.9: Generic LBP operators with (sampling points P, radius R) [25].

differences. Firstly, for making it easier to derive rotation invariant texture descriptors, the neighborhood pixels in genetic LBP operator is indexed circularly. Secondly, the value of diagonal pixels in $LBP_{8,1}$ are computed by the bilinearly interpolated method, comparing the pixels value in basic LBP.

There is another simple genetic LBP expression just consider integer distance R. The LBP operator defined by the given centre pixel (i, j) and its R-neighbour area which

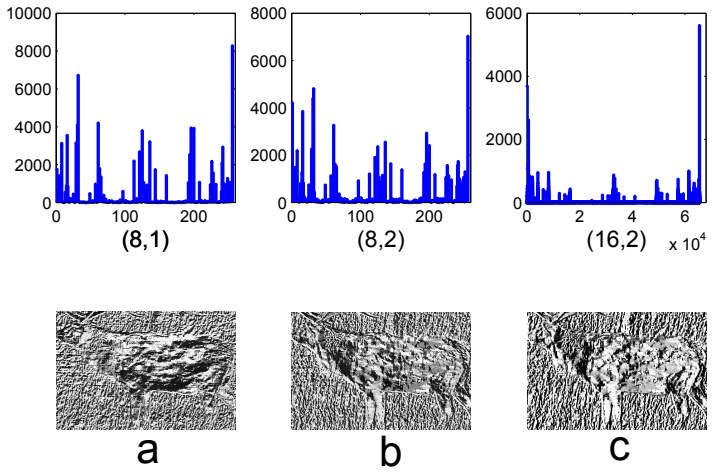


Figure 3.10: Examples of generic LBP histograms and the corresponding LBP images for moose. Horizontal axis indicate the 2^P dimension. a): LBP image (P,R)=(8,1); b): LBP image (P,R)=(8,2); c): LBP image (P,R)=(16,2).

is located R pixels away at vertical or horizontal directions. Actually, the basic LBP operator (Figure 3.7) is this type of LBP feature where $R = 1$. For each scale of R, the LBP code has $P = (8 \times R)$ binary bits and $2^{(8\times R)}$ number of histogram bins [6].

3.3.3 Multi-scale Block LBP operator

As explained in first section, traditional Haar-like feature evaluates the difference between the average intensities of two or more rectangular regions. By using the integral image method, any size of rectangles at any scale or location can be calculated in a very fast and constant time. However,we find the Haar features seem too simple and show some limits.

A new distinctive block features, called Multi-block Local Binary Pattern features, short for MB-LBP was proposed by Zhang Lun [56], they applied MB-LBP and AdaBoost learning method in face detection which has a better performance than Haar-like features. The basic idea of MB-LBP is that the simple difference rule in Haar features is transferred

into 3×3 blocks intensities by LBP operator. That means, the MB-LBP operator is defined by comparing the central block's intensity b_c with those of its 8 neighborhood blocks $b_0, ..., b_7$, see in Figure 3.11. We only apply $LBP_{(8,1)}$ as the operator of MB-LBP since more blocks or blocks with a long distance have vary limited description function comparing with adjacent blocks. Then, the final binary sequence can be obtained as follows:

$$MB - LBP = \sum_{i=0}^{7} s(b_i - b_c) \times 2^i$$

where b_c is the average intensity of the center block, $b_i (i = 0, ..., 7)$ are those of its neighborhood blocks. and:

$$s(x) = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ if } x < 0 \end{cases}$$

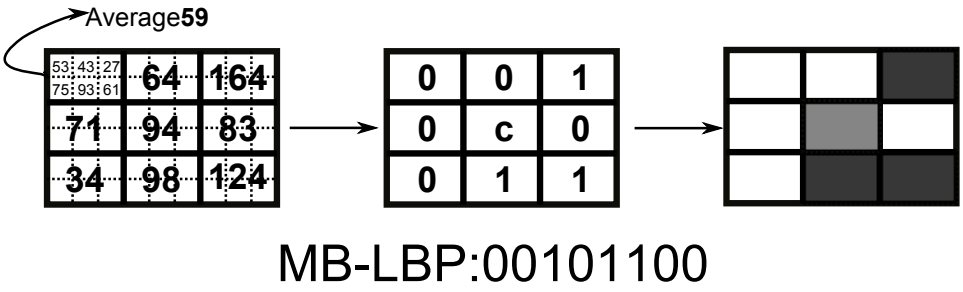


Figure 3.11: Image representation of Multi-block LBP feature. Compared with original LBP calculated in 3×3 pixels, MB-LBP can capture large scale structure.

Comparing with the traditional Haar-like features only describe 2, 3 or 4 rectangles average intensity differences, MB-LBP features contains more rectangles (blocks) intensity relationships. Moreover, Haar-like features are floating-point data, they are inconvenient to store and calculate. MB-LBP features belong to integer texture features which are more convenient and fast for further processing. That is why MB-LBP features are more popular than Haar-like features especially adopt AdaBoost algorithm in objective detection.

Normally, the size of the blocks in MB-LBP are from 1×1 to 5×5. Besides the MB-LBP with block size 3×2 shown in Figure 3.11, some other example with different block size are illustrate in Figure 3.12.

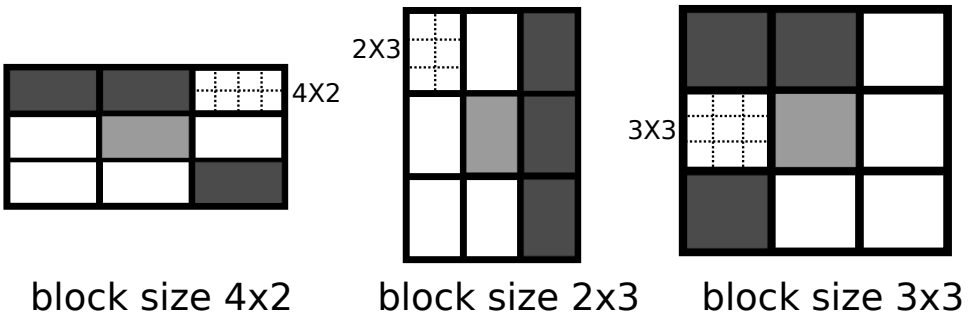


Figure 3.12: Examples of MB-LBP features with different block size.

Chapter 4

Data Classification

Classification algorithm is the core of animal detection system. In brief, the role of classification is to determine whether the input sub-window contain an animal or not. Broadly speaking, given a set of training examples (image, human, data, etc.), each marked as belonging to one of M categories (generally, $M = 2$. animal or non-animal, face or non-face.), a classification algorithm builds a system or model that can assigns new examples which are not belong to the training examples into one category or others. The same process happened in image target detection. Classification algorithm in machine vision and image processing is that of determining whether or not the input image contains some specific object (human-face, pedestrian, animal, traffic sign, etc.). After we have the features to describe the image's details, an algorithm is needed to justify whether the image is our target. The algorithm has two main effects, the first one is to process the image feature in training step to generate the classifier based on the image database, the second one is to analyse the input image and detect the target. In this chapter, two popular classification algorithm are introduced, AdaBoost and Support Vector Machine (SVM).

4.1 Adaptive Boosting (AdaBoost) algorithm

Adaptive Boosting, AdaBoost for short, is a machine learning algorithm proposed by Yoav Freund and Schapire [11]. Schapire and Singer present a generalized version of AdaBoost to improve the performance of this algorithm. In 2001, Viola and Jones creatively applied this algorithm in human face detection [52], which start a new chapter in image processing technology.

As we know, the set of Haar-like features, LBP features (MB-LBP) and HOG features are extremely large. Especially for Haar features, there are 45891 Haar-like features in a sub-window size of 20×20, whereas for MB-LBP and HOG, there are nearly 3600 and 576 respectively. Although the feature sets of MB-LBP and HOG are much smaller than Haar-like features, all of the three image descriptors contain dramatically redundant information. For instance, the background information around the animal's legs and head are useless.

The AdaBoost algorithm is used to pick out effective features and construct them as a powerful classifier. In object detection technology, AdaBoost is used to solve the following three fundamental tasks:

- Evaluating and picking out the significant and effective input features.
- Building "simple" or "weak" classifiers, each of which is built from a single candidate features.
- Establishing the selected weak classifiers into a stronger one by the boosting process.

4.1.1 Weak classifier

We assume each single feature (Haar, HOG or LBP) can be used as a classifier which can separate the positive from negative image examples, even though each one has very limited classification ability. The weak learning algorithm is designed to select a single

image feature which yield the best result. For each feature, an optimal threshold classification function is generated with the purpose of minimizing the number of examples which are misclassified. We define a weak classifier as $h_j(x)$ which consists of a feature f_i, a threshold θ_j and a coefficient factor p_j indicating the direction of inequality sign ("<" or ">" between $f_j(x)$ and θ_j). The Equation 4.1 indicates the optimal threshold classification function [53]:

$$h_j(x) = \begin{cases} 1 \text{ if } p_j f_i(x) < p_j \theta_j \\ 0 \text{ otherwise} \end{cases} \tag{4.1}$$

where x is a sub-window of an input image, f_j indicates the j-th Haar feature or the HOG histogram bins value (we often use $H_{k,j}$ replace f_j to illustrates the j-th histogram bin value in the k-th cell when we extract HOG weak classifier.). The MB-LBP optimal threshold classification function is different from Equation 4.1, we will introduce it in Equation 4.3 after the AdaBoost learning algorithm.

As a matter of fact, just one single feature can distinguish the positive and negative with very low error is a quite difficult task. Based on the experimental result, the best single weak classifier selected in face detection process yield error rates between 10% and 20%, this error rates would be much higher in animal detection. Figure 4.1 gives two of the most powerful weak classifier in Viola and Jones's human face detection. Actually, any feature with the error rates less than 50% can be seen as a useful weak classifier which may applied in further boosting stages.

4.1.2 AdaBoost learning algorithm

The Algorithm 1 [52] shows the learning process based on AdaBoost theory to select T critical weak classifiers from the set of all positive weak classifiers.

Where T classifiers are constructed which is restricted to using a single feature. The final strong classifier is a weighted linear combination of the T hypotheses. i and j are the index of image samples and features separately. At the beginning, the initial weights

1 Given example images $(x_1, y_1), ..., (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.

2 Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.

3 **for** $t = 1; t < T; t++$ **do**

4 Normalize the weights,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

to make w_t as a probability distribution.

5 For each feature, j , train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to

$$w_t, \epsilon_j = \sum_i w_i |h_i(x_i) - y_i|.$$

6 Choose the classifier, h_t, with the lowest error ϵ_t.

7 Update the weights:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

8 **end**

9 The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$

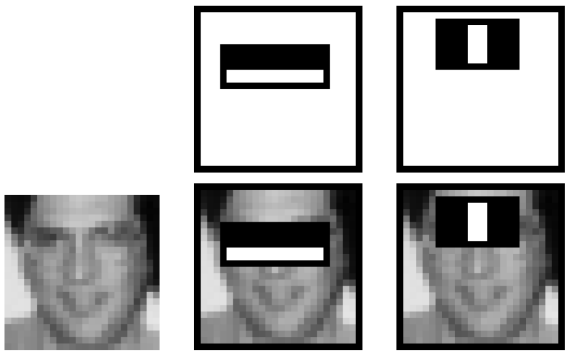


Figure 4.1: Two of the most efficient Haar-like weak-classifiers selected by AdaBoost learning algorithm. The two-rectangle feature measures the difference intensity between the eyes region and upper cheeks region. The three-rectangle illustrates that the intensities in the both eye regions darker than the bridge of nose region.

$w_{1,i}$ for positive and negative are same respectively, $1/2m$ for positive samples and $1/2l$ for negative samples. After that, the weights would be updated in the iterative loop.

For purpose of ensuring fast classification, the learning process must exclude a large majority of the available features, and just focus on a very small set of significant features. Based on the Algorithm 1, each stage of the boosting process only selects a new efficient weak classifier.

After we choose all the T best weak classifiers, a final strong classifier is combined as:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

where the weights α_t are obtained from the loop and inversely proportional to the training errors ϵ_t.

MB-LBP-AdaBoost algorithm

If we apply AdaBoost algorithm on MB-LBP features, it is difficult to use threshold based function since the value of MB-LBP features is non-metric, the output binary code is nothing but a symbol for indicating the binary string. A new weak learners called decision trees or regression trees is applied, the multi-branch tree was adopted to design the weak classifiers based on MB-LBP features [56], the multi-branch tree has 256 branches in all which correspond to a certain discrete value of MB-LBP features, shown as Equation 4.3.

$$h_j(x) = \begin{cases} a_0, & \text{if } x^k = 0 \\ \quad ... & \\ a_j, & \text{if } x^k = j \\ \quad ... & \\ a_{255}, & \text{if } x^k = 255 \end{cases} \tag{4.3}$$

where x^k is the k-th element of feature vector x, and $a_j, j = 0, ..., 255$ are regression parameters to be learned in AdaBoost training progress. We can calculate the best tree-based weak classifier just as we would learn a node in decision tree. The minimization of Equation 4.3 gives the following parameters [56]:

$$a_j = \frac{\sum_i w_i y_i \delta(x_i^k = j)}{\sum_i w_i \delta(x_i^k = j)}$$

Obviously, the parameters $a_j \in [-1, +1]$. $a_j > 0$ indicates the possibility of MB-LBP feature with value j extracted from a positive sample is greater than from a negative one. Thus, we set a threshold T_{MBLBP} for $h_j(x)$, when $h_j(x) \geq T_{MBLBP}$, we say the input window x is the positive animal, otherwise not.

HOG-AdaBoost algorithm

In SVM classifier, all the $36N$ (N is the number of blocks) HOG vectors are extracted and participated in the classification. However, only a small set of histogram values

are applied in AdaBoost based algorithm, this small set histogram values called weak classifiers. This means, each single histogram in one bin of the cell has a capability of classification, even it is very limited. We set a threshold T for each bin value and then compare the value of one image, if the value is larger than T, we consider the image is a positive target, otherwise not. Then we have nine weak classifiers corresponding to each bin, and 36 to each block. The AdaBoost algorithm is aimed at picking up the most powerful weak classifiers from the $36N$ histogram bins and combine them as a strong classifier. The weak classifier in HOG feature vector is defined as Equation 4.4

$$h_{k,j}(x) = \begin{cases} 1 & p_j H_{k,j}(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where x is the input window, $H_{k,j}$ indicates the j-th histogram bin value in the k-th cell. θ_j is the weak classifier's threshold corresponds to the j-th feature. p_i is a coefficient with the value either $+1$ or -1 to control the direction of the inequality.

A number of AdaBoost trained strong classifiers can be linked by "cascade" algorithm to combine a more efficient and accurate classifier. We will introduce this algorithm later.

4.2 Support Vector Machines (SVM) algorithm

Support Vector Machines (SVM), also called Support Vector Networks, is another famous and powerful pattern recognition classification algorithm improved by Vapnik [51]. SVM has a very wide range of applications in data analysis, regression analysis, pattern recognition, etc. We only focus on its application in image classification and image recognition tasks. A classical application in SVM is pedestrian detection proposed by Navneet Dalal and Bill Triggs, they adopted locally normalized Histogram of Oriented Gradient (HOG) descriptors and applied SVM as a baseline classifier throughout the study to build an excellent pedestrian detector [8]. After that, some improved application based on SVM in traffic sign classification [46] and [35], people & vehicle detection[13] and other objective detection shown SVM has an outstanding ability in image recognition and classification.

4.2.1 SVM Algorithm

First, we explain SVM algorithm by a general optimal separating hyperplane problem. Assuming we have a set of training examples database with the size N, we express the example as:

$$(x_i, y_i)_{1 \leq i \leq N}$$

where $x_i \in R^n$, is a kind of descriptor vector belongs to class labeled by $y_i \in \{-1, 1\}$, we can assume the class indicates the binary result of the examples, -1 means false or negative, 1 means true or positive.

The classification aim is to construct the equation of an hyperplane which divides the set of database such that all the points with the same value of $y_i = 1$ are on the one side of the hyperplane, and the other points labeled by $y_i = -1$ are on the other side. In other words, we need to find w and b that satisfy the following inequality:

$$y_i(w.x_i + b) > 0, i = 1, ..., N \tag{4.5}$$

We note that the hyperplanes are not unique, or even not exist. When there exists an hyperplane satisfying Equation 4.5, we say the sample set is linearly separable. In such a case, it is always possible to update w and b such that:

$$\min_{1 \leq i \leq N} y_i(w.x_i + b) \geq 1, i = 1, ..., N$$

Under this condition, the distance between the closest point to the hyperplane is $1/ \parallel w \parallel$. Then, Equation 4.5 can be converted to:

$$y_i(w.x_i + b) \geq 1 \qquad \forall i \tag{4.6}$$

Among the separating hyperplanes, there exist one, for which the distance to the closest point is maximal, is called Optimal Separating Hyperplane (OSH). That is to say, the OSH is separating hyperplane which maximizes the margin $2/ \parallel w \parallel$. See Figure 4.2.1.

Finding the maximum margin is equivalent to minimize

$$\frac{1}{2} \parallel w \parallel^2 \tag{4.7}$$

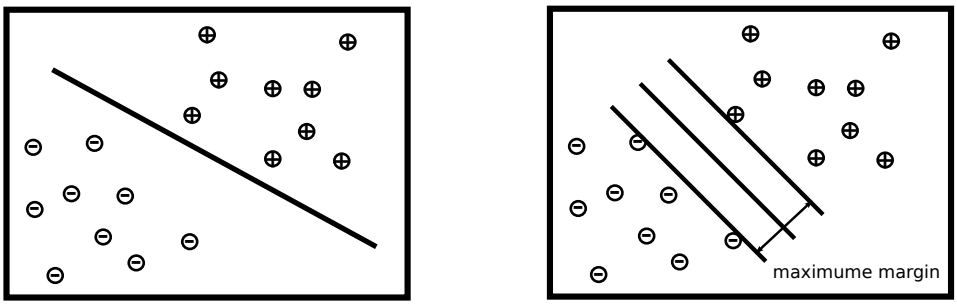


Figure 4.2: An example of optimal separating hyperplane in 2D plane, both hyperplanes separate perfectly the training data. However, the OSH on the right image has the largest margin and is expected to give better classification performance.

The specific process was introduced in *Linear Support Vector Machines* of [4].

Unfortunately, not all data samples are linearly separable. In this condition, it is infeasible to finding the optimal separating hyperplane. A set of slack variables are introduced to handle the non-separable data problem:

$$y_i(w.x_i + b) \geq 1 - \xi_i, i = 1, ..., N \tag{4.8}$$

where $(\xi_1, ...\xi_N)$ with $\xi_i \geq 0$. The purpose of the slack variable ξ_i is to allow misclassified points. The points which would be misclassified have their corresponding $\xi_i > 1$, at the same time, $\sum \xi_i$ is an upper bound on the number of training errors. The generalized OSH problem is to change the objective function to be minimized from Equation 4.7 to:

$$\frac{1}{2}w.w + C\sum_{i=1}^{N}\xi_i \tag{4.9}$$

where Equation 4.8 is a constraint and $\xi_i \geq 0$, C is a parameter to be chosen by the user, a larger C corresponding to assigning a higher penalty to errors. This situation is illustrated in Figure 4.3.

4.2.2 HOG SVM classifier

Object detection based on SVM algorithm in image processing is a non-separable model. As mentioned before, the classic application is Histograms of Oriented Gradients (HOG)

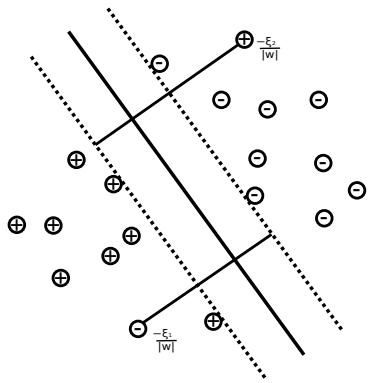


Figure 4.3: An example of applying linear separating hyperplanes in non-separable problem.

plus SVM classifier in pedestrian detection.

In the training step of pedestrian detection, we assume there is a set of pedestrian database contains positive pedestrian samples and negative non-pedestrian. A kind of cell & block-based HOG features are extracted for describing each image sample, this descriptor is a high dimensional vector ($3,780$ dimensions in 64×128 pedestrian window) as x_i appeared in Equation 4.5 and Equation 4.8. In other words, the SVM training algorithm is aimed at finding the classification vector w and the classifier threshold b with the minimum value of Equation 4.9.

Chapter 5

Animal Database Creation

5.1 How to create an animal database?

To the best of our knowledge, there is no public animal database which exists in the literature. Hence, a new database for large animal is constructed. The quality of our database directly affect the classifier's final performance. In order to build a better database, the following critical problems need to be considered:

- Animal species.
- Animal origin.
- Image size and proportion.
- Animal shapes categories.

Initially, this system will just focus more particularly on the most common animals involved in the AVCs: moose and deer, some uncommon animal like bighorn sheep, or large domestic animals such as cows are also considered. However, this project will not deal with smaller wild animals such as raccoons and skunks. Fortunately, only uniform category animal (medium and large Artiodactyla Cervidae animals) need to be considered, since most of the dangerous animals have the similar outline and size.

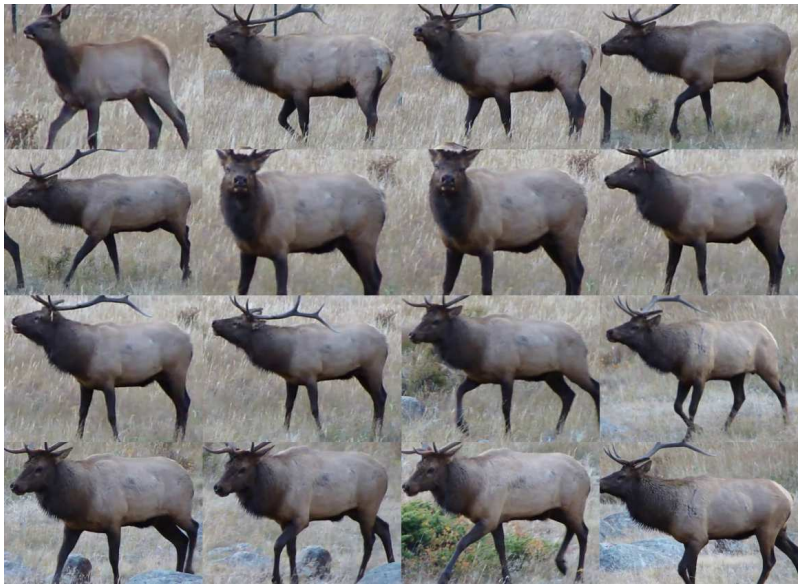


Figure 5.1: A continuous image sampling with 1 : 4 frequency from "an elk near the road" video.

How and where to collect the animals image or video? It is unrealistic to record videos directly from nature or zoos. Thanks to the rich network resource in the Information Age, we built our database from a large set of images and videos collected mainly from Internet. A huge number of family videos, scientific research videos and documentary films made a significant contribution to our database. Approximately 20 hours of video that were selected while a wild moose burst into the residential area, elk and deer appeared in front of a driving truck, biologists observed animal behaviour and so on. At last, the collected videos were downloaded from some video websites (e.g., Youtube, Youku) and converted to image format using the ratio 1 per 4 continuous frames to avoid repeated sampling, as can be seen in Figure 5.1.

On average, an adult North American moose stands $1.4 - 2.1m$ $(4.6 - 6.9ft)$ high at shoulder. The head-to-tail length is $2.4 - 3.2m$ $(7.9 - 10ft)$ [31]. The other Cervidae animals have the similar body ratio. Based on the general characteristics of moose and

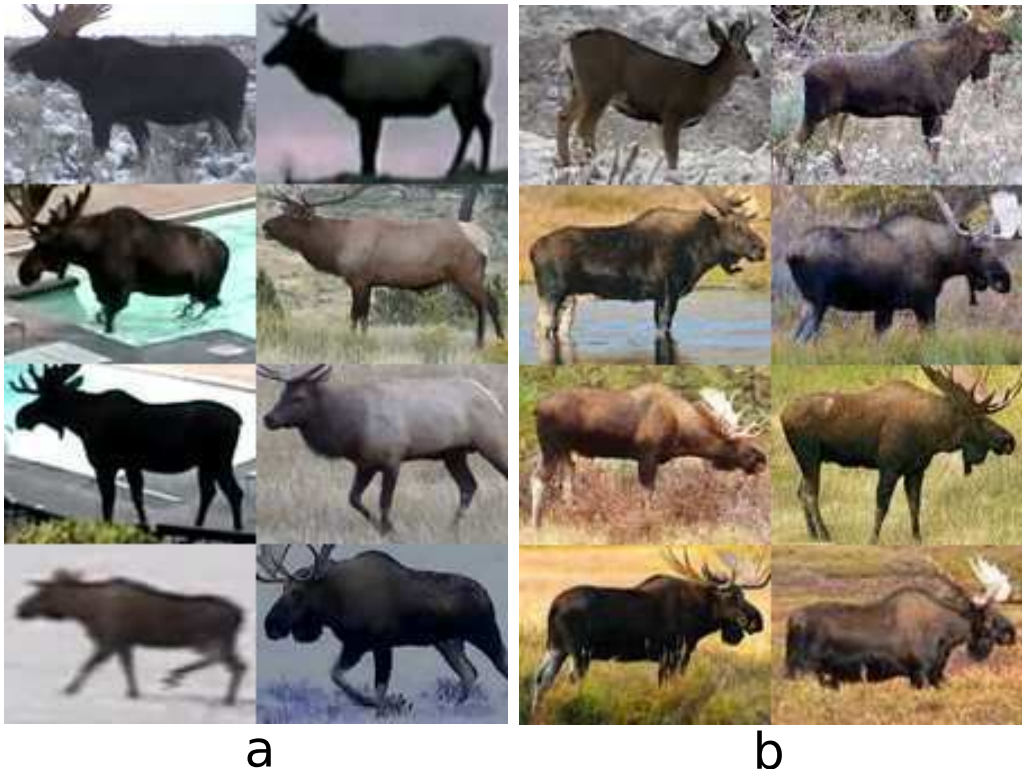


Figure 5.2: a): Head to Left (HTL) shape category; b): Head to Right (HTR) shape category

deer, we select the image size which should match the animals natural size. Hence the image size $width : height = 7 : 5$ is adopted. The images from Internet vary from 42×30 to 560×400 pixels are normalized to 28×20 and 56×40 two different specifications, which applied to AdaBoost classification and SVM classification respectively. We note that each image should contain 10% boundary area around the animals' body, which allows for the usage of edge detector [47].

In fact, the hardest problem of animal body detection is the various body postures (will be fully explain in Chapter 6). It is unrealistic to build an omnipotent classifier which can recognize all kind of animals with random pose. Therefore, as the preliminary stage of the research, we just considered two main side-view shape categories, "Head to Left"(HTL) and "Head to Right"(HTR). Figure 5.2 shows some typical samples of HTL

Figure 5.3: Examples of non-standard HTR or HTL image

Figure 5.4: Future work's shapes.

and HTR categories. Actually, these two categories are the most frequent shapes we would meet on the road. However, some non-standard images like "face to camera" or a certain range of rotation and sideways were also acceptable to expand the detection range. Figure 5.3 gives some non-standard but acceptable images for our database, and animal shapes listed in Figure 5.4 will be considered in our future work such as rear-view, front-view, lie down and so on.

5.2 Positive animal database images

After finishing the video collection, we cut the standard image from the sampled video frames using the constant proportion window ($width : height = 7 : 5$). As mentioned before, the window size was determined by the animal body size plus %10 margin, and would resize to normalized sizes in the end. In order to improve the detection accuracy

Figure 5.5: Example of horizontal flip HTR image to get HTL image.

and adapt the driving detection environment, the animal videos recorded from moving vehicles represent our first of choice. Also note that, multifarious weather conditions e.g. rainy, sunny, snowy and foggy, a wide range of illumination, other variant environment conditions such as different seasons and even slight blurred would dramatically enhance the final system's performance [43].

As above-mentioned, two main animal categories are considered in positive database. Each category concerns the side-view of the animal in the images, HTL category (see Figure 5.2 (a)) and HTR category (see Figure 5.2 (b)). The property of animal body symmetry makes an interesting result. If we flip one category's image by horizontal 180°, we can perfectly get the other image category (shown in Figure 5.5), that gives us an easy way to double our positive database. Meanwhile, we luckily get a balanced database, namely, both of the two category images occupied uniform 50% of the whole database.

Finally, we create a positive database contained 5832 animal images include moose, deer, elk, mustang, wild horse, etc. Figure 5.6 gives some additional positive images for other species and environment conditions.

5.2.1 Positive images used for testing

Both positive and negative database are divided into two subsets to avoid the images used in the training process repeated in the testing set, which can make the final result more

Figure 5.6: Other positive images include moose, deer, wild horse, elk, Cervus canadensis and White-tailed deer

persuasive. Based on the fact that Viola and Jones have applied 5000 positive images in their human-face training database [53] and Tilo Burghardt trained 680 in lion-face detection [5], we choose 3462 images as our Positive Training Database. Therefore, the remaining 2370 was insufficient for performance testing. To obtain more testing pictures, the image contrast adjustment was implemented for the Positive Testing Database. In particular, we increase and decrease the image contrast by 50% separately to triple the testing database (7110) and simulate the sunny and cloudy environment as well. Figure 5.7 gives the initial image and corresponding adjusted image. What calls for special attention is that this way only applied to testing database rather than training database, which can guarantee the authenticity and practicality of the final classifier.

5.3 Negative animal database images

A good negative (background) database can improve the final detector's performance to some extent, especially for the detector in the specific detecting environment (driving on the road). Here are some principles of negative dataset creation:

Figure 5.7: Left:initial image, Middle:decrease contrast by 50 to simulate the cloudy weather condition, Right:increase contrast by 50 to simulate the sunny condition.

- The most crucial point is that the target object (i.e., animals) can not be contained in the database.
- Any other animals like dog, cat, etc. are excluded from the negative dataset, since they may have similar shapes with big animals.
- The image size and proportion are unfixed, since the negative training inputs are randomly chosen from the background dataset.
- Images containing road-objects such as vehicles, pedestrian, traffic sign, road surface, forest, grass, house, tree, etc constitute the main database.
- Weather, illumination and other environment conditions should be considered as well.

We use more than 8000 background images as our negative database, see Figure 5.8.

5.4 Animal detection database summary

Our database contains:

- 3462 positive images used for training with several different size: 21×15, 28×20, 35×25, and 56×40.

Figure 5.8: Negative database

- 7110 positive images having 56×40 pixels are used for accuracy evaluation purposes (Trained images were not included).
- 7000 negative images are used as training background.
- 31712 negative images having 56×40 pixels are used for false positive testing.

Chapter 6

Double-layer Animal Detection System

6.1 Animal detection: challenges and issues

Comparing with human-face, pedestrian or traffic sign detection, a number of challenges and difficulties have to be addressed in animal detection. For instance, human-face has relatively stable texture feature which can be appropriately described by Haar feature. HOG descriptor is applied to pedestrian detection since the human body's outlines are nearly invariable even when they are walking into different directions. On the other hand, traffic sign has distinct colour characteristic, which can be recognized by certain colour segmentation. However, with animal detection (moose,deer, etc.), too much differences in colour, outline, shapes and other variations are perceived. Moreover, different kind of animals can appear under any imaging conditions such as different illumination lighting, and blurred and complex backgrounds. The following challenges are addressed in this chapter:

6.1.1 Animal body vs. animal face

Face detection is an important and promising research direction since Viola and Jones proposed a cascade method based on AdaBoost algorithm [53]. Tilo Burghardt and Janko presents an excellent real-time method for detecting and tracking the lion faces using Haar-like features trained by AdaBoost [5]. A new face verification algorithm based on Gabor wavelets and AdaBoost was proposed by Mian Zhou and Hong Wei which has a good result on a single species animal as well [59].

Nevertheless, on the one hand, moose and deer's faces are stereo instead of human-face (lion-face and cat-face) which are flat. The face is invisible when a moose is crossing the road; on the other hand, it is not easy to detect the face from the same colour body even if the face is observable. Moreover, most of Cervidae animal bodies are unicolor and have similar outlines. Furthermore, human-face can be characterized by a skin texture, while animal-face textures are variable and more complicated. To sum up, choose body as our detecting target is a more feasible choice.

6.1.2 Various body postures

The body postures of road side animals exhibits very high variability within the same class, and between different classes. Compared to human-face and body, which are almost standard and unique, the animal body have higher appearance variations. It is almost impossible to detect all animal shape categories from one detector based on current image processing technology. That is one of critical reasons why we use a double-layer detection structure, which will extract the region of interests (ROIs) with all shape categories from first stage, and the second stage is adopted to eliminate the false positive ROIs with a number of one-category classifiers. At the present stage, only two side-views (HTR and HTL) are considered, since they are the most commonly postures when the animals are crossing on the road.

6.1.3 Light conditions (illumination)

As we know, the image colour is very sensitive to the variation of the illumination intensity and lighting directions. The illumination of the outdoor environment is changeable and uncontrollable since it changes along with the vehicle's lighting, different time of a day, different seasons, and different weather conditions. In addition, the light from the sun or other light source will reflected by the surface of the road or other object, the light will obviously affect the video quality which may miss our target.

6.1.4 Image blurring

The camera installed in a driving vehicle to catch a moving animal will cause the image blur when the vehicle driving fast, running on a bumpy road, making a turn or the camera is out of focus, this situation will reduce the detection rate and cannot be avoided radically.

6.1.5 Mobility of animals and vehicles

Our detection system is designed to identify moving animals either by road-side devices or in-vehicle systems. In the former case, we only consider the animal mobility, which makes it less complex than the second case, where both animals and vehicles are moving. This situation complicates the animal detection and tracking.

6.1.6 Accuracy of detection

Accuracy includes minimizing false negative rate and false positive rate. The false positives happen when the system triggers a false warning without any animal, which may disturb drivers in their driving task. On the other hand, false negative indicates when the system does not identify animals crossing the roads. False negatives happen in consequent frames could be very dangerous for drivers and animals.

6.2 Classifier training

Machine learning provide an effective method for processing nondeterministic models which contain incomplete information by describing and predicting a probability density over the variables in questions. Specific to our image target detection, machine learning focuses on prediction (detection), based on known properties learned from the training data. The aim of training process (i.e., learning) is to select suitable features and inform the system which of these features have common characteristic from the positive samples [18]. Figure 6.1 shows the training process and multi-scale detection process of detection system.

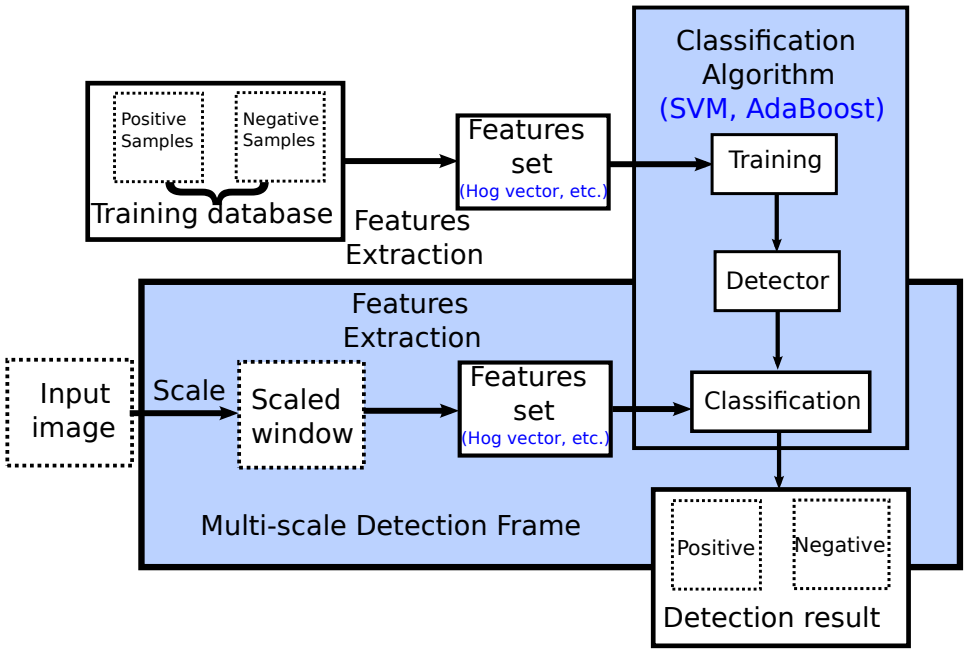


Figure 6.1: Object detection flow chart.

Given a type of feature (or some types of features) and a training database of positive and negative image samples, any kind of machine learning approaches could be used to train a classification detector. Before Viola and Jones proposed the AdaBoost learning algorithm, and support vector machine learning function, Rowley, Naluja and Kanada applied a limited set of simple image features on neural network (NN) [42]. During the

same period, an advanced mixture of Gaussian model applied as a training procedure by Sung and Poggio [49]. Also, Roth and Dan used a new and unusual image representation and Winnow training approach [41]. For the animal detection system, whether AdaBoost based classifier or SVM classifier need to be trained to judge which kind of sub-windows are true target and which are not. The machine learning algorithms will iterate and choose the useful common descriptors as their basis of recognition.

6.2.1 A cascade of AdaBoost classifiers

First of all, we adopt AdaBoost algorithm to train three different classifiers using Haar, LBP and HOG descriptors separately. Assuming each feature can be considered as a weak classifier, the total number of each type of features in a certain image is extremely large. AdaBoost training function aim at selecting a small number of powerful weak-classifiers from the enormous amount of feature sets, and gives each selected weak classifier a specific weight to form a final strong classifiers, see Function4.2.

A cascade of classifiers (hierarchical classier) is constructed in order to increase the detection performance while radically reduce the recognition time. The word "cascade" means that the resultant classifier consists of several simpler classifiers (stages) that are applied subsequently to an input window until at a certain stage the detecting sub-window is rejected (negative result) or all the stages are passed (positive result). AdaBoost training approach is applied on each stage, and one of the three type features is considered as the input to each stages' classifier.

Note that the order of each stage is not randomly decided. The key principle is that the simple but efficient AdaBoost classifiers are arranged at the early stages to reject many of negative sub-windows while accept almost all positive sub-windows. The subsequent complicated and stronger classifiers aim to achieve low false positive rates. Figure 6.2 illustrates the principle of cascade classifier, and Algorithm 1 shows the corresponding pseudo-code [52].

Input: Detection sub-window
Output: Detection result: Positive or Negative
for $i = 1; i \leq N; i++$ **do**
    $S_i = 0$
    Initialize the calculation result S_i for stage i/N classifier.
    for $j = 1; j \leq T; j++$ **do**
        $S_i = S_i + \alpha_j h_j(x)$
        where $h_j(x)$ is the j/T weak-classifier, namely the j_{th} selected feature in this stage.
    end
    if $S_i \geq Threshold_i$ **then**
        $RESULT \leftarrow TRUE$
        The looping would continue
    end
    else
        $RESULT \leftarrow FALSE$
        skip detection and the result is negative.
    end
end
return RESULT

Algorithm 1: Cascade classifier

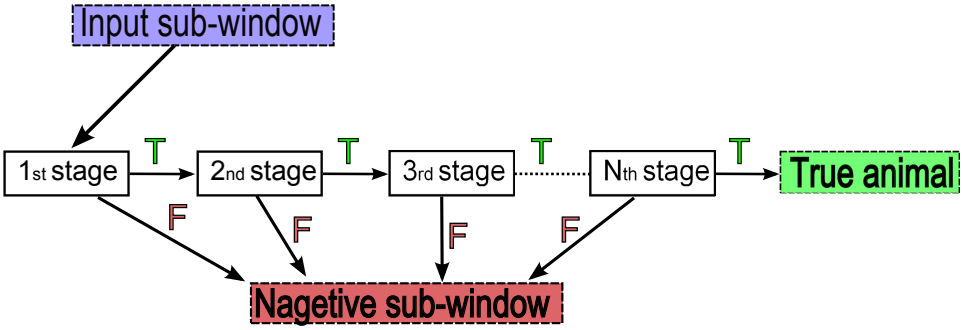


Figure 6.2: A cascade classifier is applied to all input sub-windows. The false output of any stage leads to the immediate rejection of the input. This means that a great number of negative sub-windows are rejected rapidly by the initial stages, subsequent stages eliminate additional negatives and nothing but true positive target will trigger and pass the evaluation of every stage.

Classifier training

Assuming we need to train a cascade classifier with stage number N, than the false positive rate (FPR) and detection rate of the classifier are:

$$F = \prod_{i=1}^{N} f_i \qquad \text{and} \qquad \mathrm{T} = \prod_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{t_i}$$

where F and T are the FPR and the accuracy rate of the final classifier respectively. f_i and t_i are the FPR and the detection rate of the ith stage classifier.

If we want that our classifier achieves the detection rate of 90% (for the whole 20 stages), the minimum hit rate for each stage would be:

$$\log_{0.9} 20 \approx 0.995$$

At the same time, if we define the FPR of each stage classifier as $\leq 50\%$, the maximum false positive rate would be less than:

$$0.50^{20} \approx 0.95 \times 10^{-6}$$

Which would be a very low false positive rate.

Note that these false positive rate F and accuracy rate T only obtain from the training database, the further evaluation test is necessary.

6.2.2 Number of positive samples in training process

The number of positive samples used in training database is a critical factor which dramatically affects the final detector's performance. In the traffic speed sign detection work by Keller et al [19], a speed limit classifier was trained on $2,880$ positive images. It achieves a very high detection rate around 92.4%. A HOG based pedestrian detector designed by Dalal selected $2,478$ human images in his training process [8]. The famouse Viola-Jones' human-face detection applied $5,000$ human-face samples in his cascade classifier [52].

Before we start building animal classifiers, we need to decide how many positive samples will be used in training process. Therefore, a simple experiment for exploring the relationship between detection rate and the size of positive database is carried out. We build 8 LBP-AdaBoost classifiers under same parameters but different "number of positive samples" from 100 to $3,200$. After that, we evaluate the performance of each classifier based on test images and draw Figure 6.3 to show the relationship. This figure indicates that the detection rate increases rapidly when the number of positive sample increases from 100 to 2800, after that, the detection rate remains at about 83%. Hence, we apply $3,200$ positive images in training process.

6.2.3 The used classifiers

Our system is trained and tested using the following hardware platform:

- Processor: Intel® Core™ i5 − 2450M cpu @2.50GHz × 2
- Installed memory (RAM): 4.00 GB
- Graphics Chipset: AMD Radeon HD 7650M

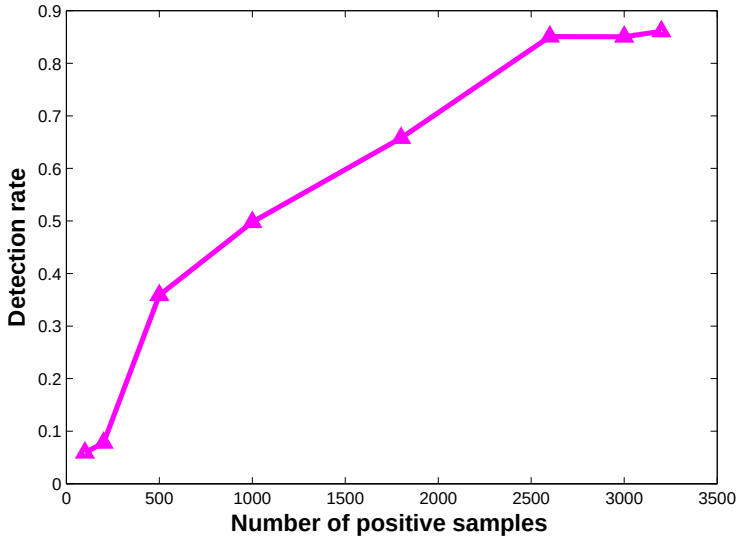


Figure 6.3: Relationship between detection rate and the number of positive samples applied in training database.

and software environment:

- Microsoft Windows 7 64-bit Professional with Service Pack 1
- Microsoft Visual Studio 10 professional
- OPENCV 2.4.3 [2]

In this chapter, Har, LBP and HOG features are trained separately on the same image database (positive and negative images), using the well know AdaBoost algorithm. These classifiers are referred in our paper to "Haar+AdaBoost", "LBP+AdaBoost" and "HOG+AdaBoost" classifier. Table 6.1 lists the explanations and values of training parameters about AdaBoost, cascade, and database.

Specifically, the "numPos" and "numNeg" are the number of positive/negative samples used in training process for every stage, these values should be less than the total number of training database images. 3200/3462 positive and 6200/7000 negative samples are input in each stage. 99.5% and 50% are set as the minimal detection rate and

-numPos	<number of positive samples>	3200
-numNeg	<number of negative samples>	6200
-numStages	<number of stages>	20
-featureType	<type of features>	Haar,HOG,LBP
-w	<width of training sample>	28
-h	<height of training sample>	20
-minHitRate	<minimal desired hit rate for each stage >	0.995
-maxFalseAlarmRate	<Maximal desired false alarm rate for each stage>	0.5

Table 6.1: AdaBoost classifier training parameters

maximal false positive rate respectively to pursue the high accuracy rate and low FPR. In addition, balancing the training efficiency is also needed to be taken into account, that's why we choose 50% (not so low) as the stage's FPR.

One interrupted case may occur during the training process. After the first 15 or more stages, training process may keep running for a very long time but cannot achieve the current stage's goal. This situation indicates the classifier is overtrained. Classifier will not find extra powerful weak-classifiers to satisfy the present stage target and the previous stages are good enough. The precision of the cascade classifier is shown by "acceptanceRatio" in the training process (See Figure 6.4), if the value is less than $1.00e-06$ in the present stage, the classifier is probably overtrained. That's why the final classifiers may contains no more than 20 stages.

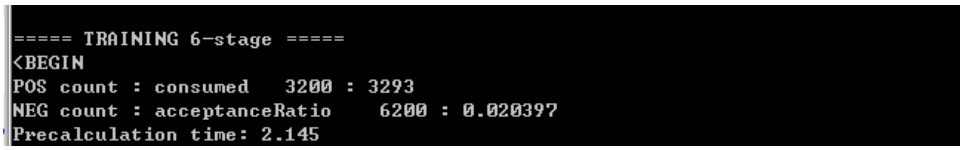


Figure 6.4: The current training stage: 6. "acceptanceRatio" indicates the classifier's training precision

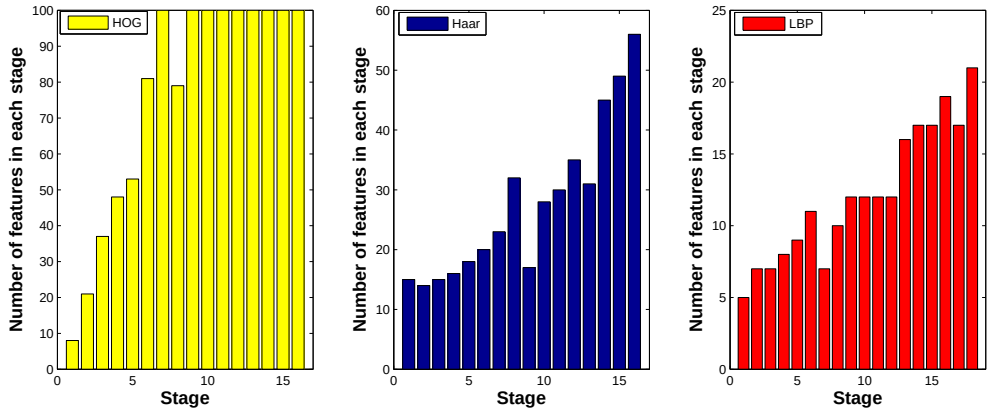


Figure 6.5: Number of features chosen by each stage for the three classifiers.

Figure 6.5 shows the number of features selected for each cascade stage. The classifier HOG+AdaBoost applies the largest number of weak-classifiers in each stage, LBP cascade classifier only uses less than 10 features in earlier stages and around 17 in latter stages, and Haar detector chooses increasingly from 15 features to 56 features. Before evaluation test, LBP features show more powerful and more efficient than other two descriptors.

6.3 Comparison between three classifiers

To evaluate the performance of each classifier, three main criteria are considered: detection accuracy, detection time and the energy consumption. The details about energy consumption will be introduced in the next chapter.

6.3.1 Detection time

Initially, we chose a continuous video instead of single images as our input to simulate the real-time situation. An angry moose is running around and showing its several postures including HTR, HTL, face-view, rear-view and so on in the video. We tested the three cascade classifiers on this same video to record the detection time. We then obtained

the two figures illustrated in the following: Figure 6.6 illustrates the specific detection time (ms) for each frame and Figure 6.7 directly indicates the differences between the average detection speed of the three classifiers calculated from the previous figure.

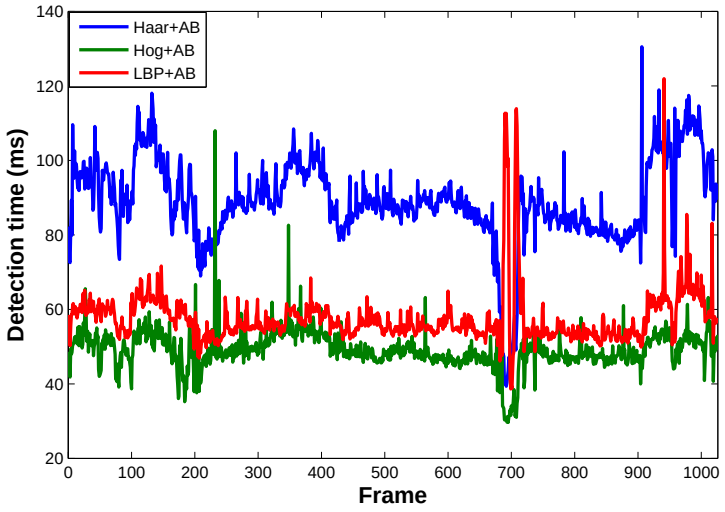


Figure 6.6: Detection time per frame: the input video size: 320×240; total frame number: 1026; initial (minimal) detection window size: 56×40; scale rate: 1.05.

Figure 6.7 indicates the Haar+AdaBoost detector consumes the most time ($89.0ms$) compared with the other two detectors; and that HOG feature performs the best ($49.1ms$). However, it performs faster than LBP+AbaBoost by just $8ms$ per frame when only considering the speed criterion. Besides the "minimum initial window size" and "scale rate", the speed of the cascaded classifier is directly related to the image feature types, the number of features (weak-classifiers) triggered per sub-window, and the quantity of final "positive" (true or false positive) results.

Table 6.2 lists some comparisons between the three descriptors. We can find some information to explain the speed performance: Firstly, because of feature complexity, calculating a single Haar or LBP feature is more complicated than calculating HOG features when the HOG block size is very small; this is the main reason why HOG+AdaBoost perform best in speed test. Actually, HOG+AdaBoost has a huge number of false posi-

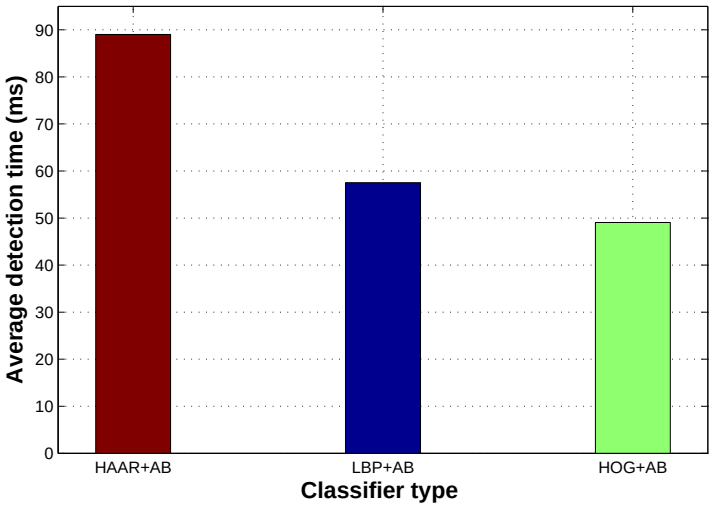


Figure 6.7: Average detection time of three classifiers: $89.0ms$, $57.5ms$ and $49.1ms$

tive results (See Figure 6.8) which decreases the speed, but it is still the fastest. When it comes to Haar and LBP, LBP is a kind of binary feature and the number of total features in the LBP cascade classifier is much less than in the Haar detector (See Figure6.5); this makes the LBP+AdaBoost speed much faster than that of Haar.

6.3.2 Accuracy rate

Actually, the video detection results (Figure 6.8) give us an apparent performance difference between the three detectors. LBP classifier wins the comparison easily and obviously, whether in true positive results or in false negative misjudgements. In addition, Haar cascade detector obtains a better result than HOG classifier especially in a false positive rate.

For further data analysis, we need to finish the accuracy test based on our prepared testing database that includes $7,110$ positive images and $31,712$ background samples. Firstly, we run one of the three detectors several times on positive samples and negative

[1]It is only the intuitive sense based on animal video detection, the statistical data will be shown later.

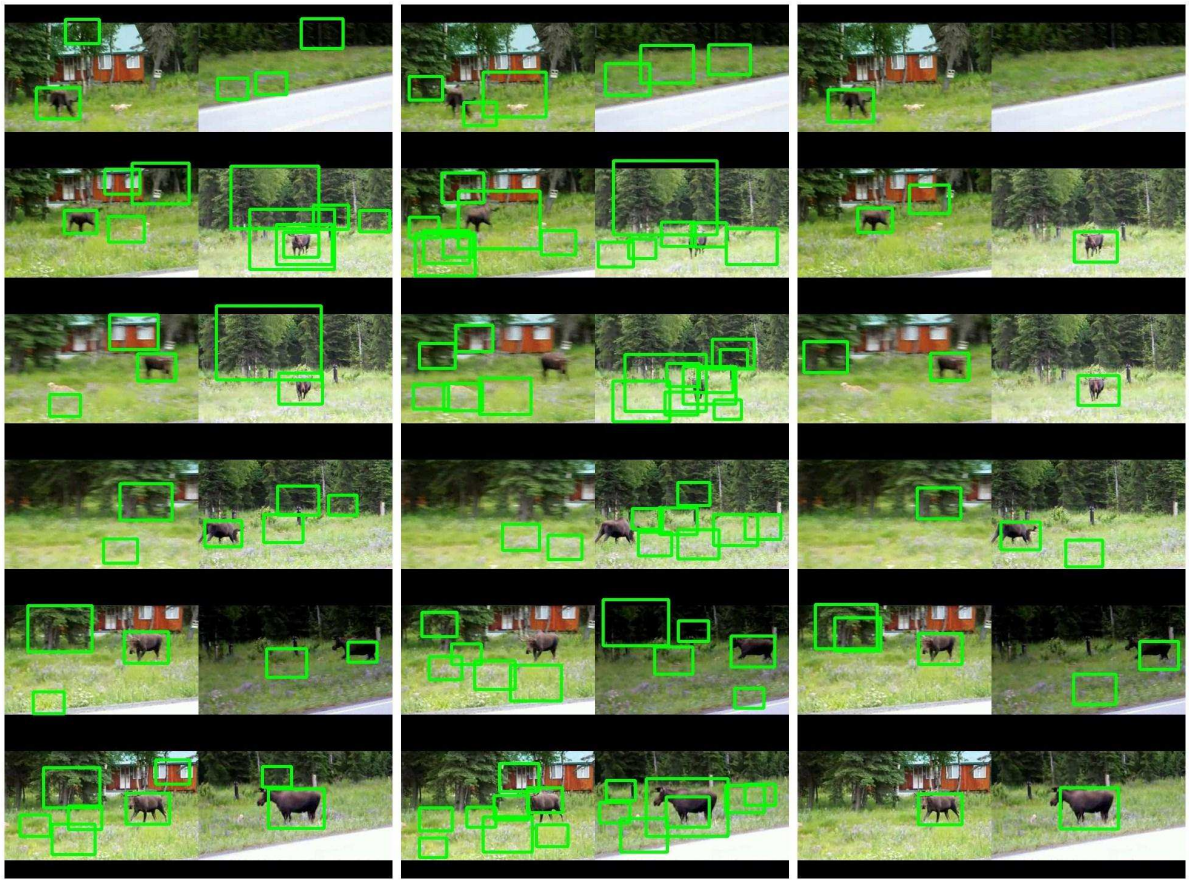


Figure 6.8: Video detection results: Left:Haar+AdaBoost; Mid:HOG+AdaBoost; Right:LBP+AdaBoost.

	Haar	**LBP**	**HOG**
Proposed by	Viola[52]	Ojala[32]	Dalal[8]
Training time(AB)	$> 36h$	$\approx 12h$	$\approx 3h$
Application example	Human face	Product quality	Pedestrian
Image type	Gray	Gray	Gray or colour
Data Type	Integer	Binary	Float or vector
Pixel feature		$\surd$	
Block feature	$\surd$	$\surd$	$\surd$
Features quantity in 28×20	$> 200,000$	≈ 3600	560
True positive rate[1](Figure 6.8)	Very high	Very high	High
False positive rate[1](Figure 6.8)	Many	A few	A great number

Table 6.2: Some information comparisons for Haar, LBP, and HOG.

samples separately with different scale rates. Then we record whether the positive sample is identified and how many false positive results we obtained. After several experiments are finished with one classifier, we plot Detection Error Tradeoff (DET) curves on a log-linear coordinate; i.e. the "Miss Rate versus False Positive Per Image (FPPI)" curve [9] for the current classifier. Finally, we repeat the steps on the other two classifiers. Figure 6.9 illustrates the evaluation results for the three kinds of classifiers. We applied "miss rate" instead of traditional ROC curve "correct detection rate($= 1 - missrate$)"; and this makes a better detector curve appear in the lower position.

The LBP+AdaBoost and Haar+AdaBoost detectors dramatically outperform the HOG+AdaBoost system, and the former one performs slightly better than the latter one when the FPPI rate is greater than 10^{-3}. In addition, The LBP detector has a relatively lower miss rate of around 30% compared to the Haar's of 45% when the FPPI rate is less than 10^{-4}. We noticed that the Haar based classifier curve can achieve an extremely low miss rate; at that point, the system can recognize $7,030/7,110$ positive samples but has more than $3,000$ false positive results in $31,712$ negative inputs. That does not make

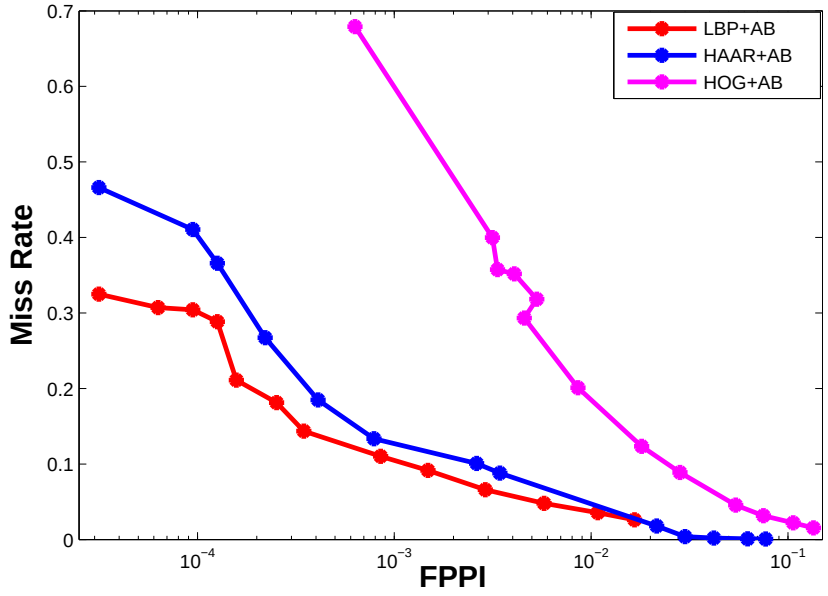


Figure 6.9: Evaluation results for the three feature classifiers reported as Miss-Rate vs False Positive Per Image (FPPI) curves; lower curves indicate better performance.

much sense in our practical application since too many false positive results would lead a great deal of false alarms.

If we associate the evaluation results of Figure 6.9 with each stage feature number (See Figure 6.5), we will find that LBP are very efficient features: only 219 LBP features applied in 18 stages achieves the highest detection rate.

6.4 HOG-SVM classifier

Navneet Dalal and Bill Triggs proposed a new type of feature sets called grids of Histograms of Oriented Gradient (HOG) descriptors which significantly outperform other existing features, and, their final grid HOG descriptors classified by linear Support Vector Machine (SVM) provide excellent performance in case of pedestrian detection. They proposed a new approach based on evaluating well-normalized local histograms of image

gradient orientations in a certain grid. In practice, this was implemented by dividing the initial image into several large overlapping spatial regions (called "blocks"), and each large spatial region also contained some small non-overlapping spatial regions (called "cells"). Then, this new approach accumulated a one-dimensional histogram of gradient over every pixel in the "cell". Finally, the authors voted the linear gradient into several "bins" to generate the final high dimensional feature vector and applied a conventional SVM based algorithm to finish the classification [8].

6.4.1 HOG parameters

We want to try this method in our animal detection study and analyze its performance compared to AdaBoost based classifiers. Before this, we have listed all the parameters involved in HOG descriptors:

- Detection window: sub-window for detection.
- Blocks

  Size: number of cells (or pixels) contained in a block.

  Stride: number of cells (or pixels) overlapped by blocks.
- Cells

  Size: number of pixels contained in a cell.
- Histogram

  Bins: the number of orientation bins in $0° - 180°$.

  Sign: gradient signed or unsigned (unsigned usually).

  Weighting vote method.

Assume the image size in Figure 6.10 is 56×40, the block size would be 16×16 pixels (or would measure 2×2 by cells) and would contain 4 cells with a size of 8×8; and, the

whole image would be divided into 24 blocks and $24 \times 4 = 96$ cells; if we set the "bins" as 9, the image HOG vector dimension would be $96 \times 9 = 864$. Generally speaking, the dimension of an image HOG vector can be calculated as follows:

$$((W.width - B.size)/B.Stride.size + 1)\times$$
$$((W.height - B.size)/(B.stride.size) + 1)\times$$
$$(C.number \times bins)$$

Where W means window, B is "Block" and C is "cell".

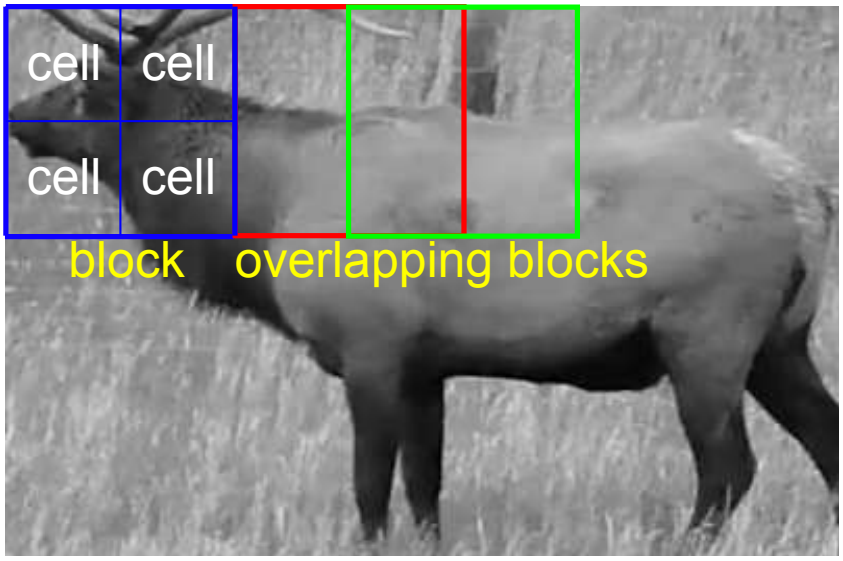


Figure 6.10: An example of block and cell distribution of an animal image, the blue rectangle (Left-top) indicates that a block consists of 4 cells and the red and green rectangles are two overlapping blocks.

While HOG vector classified by SVM detector (hereafter referred to as HOG+SVM) also needs training before being able to detect, we use the same positive and negative databases but different image size (56×40) to train our HOG+SVM classifier.

To the best of our knowledge, the exploitation of HOG features trained by SVM algorithm to detect animals was not yet performed in the past; and, Suard mentioned that different HOG parameters directly impact the final classifier's performance [48]. We

need to test and select the optimal parameters of HOG+SVM that yield the best performance results. We trained 5 groups of HOG+SVM classifiers with different parameter combinations as shown in Table 6.3.

Group	Dimension	Block	Stride	Cell	Bins
Hog 1	864	16×16	8×8	8×8	9
Hog 2	1152	16×16	8×8	8×8	12
Hog 3	4212	8×8	4×4	4×4	9
Hog 4	486	24×24	16×16	8×8	9
Hog 5	576	16×16	8×8	8×8	6

Table 6.3: Different parameters of the HOG descriptors, (Note: all block size, block stride size and cell size are expressed by $pixel \times pixel$).

6.4.2 Performance evaluation

As mentioned before, we evaluated the performance of our classifiers following two main criteria: detection speed and detection rate. We applied the same evaluation methods on the five HOG+SVM classifiers after they were trained and the same video test for speed evaluation and image testing for calculating the accuracy rate.

Figure 6.11 illustrates the detection speed of the five HOG+SVM classifiers compared with the LBP+AdaBoost detector.

From the bar charts, we find that HOG1, HOG2, HOG4, and HOG5 perform faster (between $158ms$ to $165ms$) than HOG3 detector at $343ms$. The best two classifiers in speed tests are HOG1's $158.63ms$ and HOG5's $157.97ms$. When it comes to SVM and AdaBoost speed comparisons, AdaBoost algorithm based classifiers are dramatically faster than HOG+SVM classifiers. AdaBoost consumes no more than 1/3 times what HOG+SVM classifiers consume.

Unlike AdaBoost based classifiers, a certain kind of HOG+SVM classifier takes almost constant time in one detection sub-windows. This is because all HOG operators and

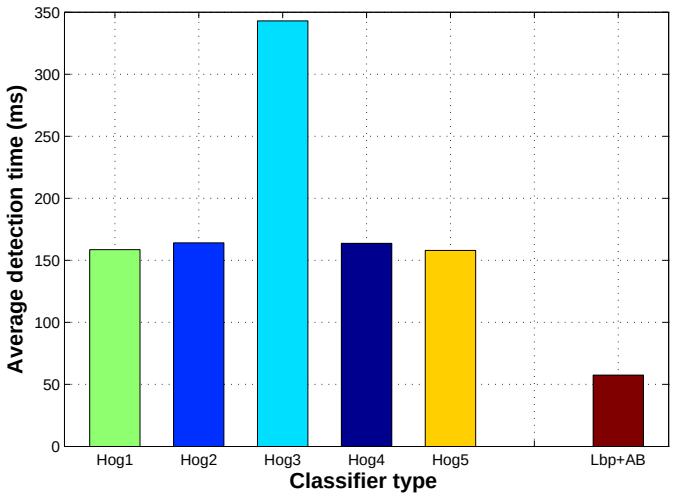


Figure 6.11: Five types of HOG+SVM detectors' average detection time: $158.63ms$, $164.16ms$, $343.00ms$, $163.68ms$ and $157.97ms$. The LBP+AdaBoost detection time is shown as the right bar.(Video size: 320×240; total frame number: 1026; initial (minimal) detection window size: 56×40; scale rate: 1.05.)

the HOG histogram need to be calculated in one sub-windows; however, the AdaBoost classifiers inconsistently expend time which depends on the number of cascade stages triggered by the sub-window. It can be observed from Table 6.3 and Figure 6.11 that HOG+SVM detection time directly depends on the dimension number.

After we finish the speed test, we also need to do the accuracy evaluation test using our testing database ($7,110$ positive samples and $31,712$ negative samples). In order to save time and to obtain smooth curves, we applied Miss-rate versus False Positive Per Window (FPPW) instead of Miss Rate versus False Positive Per Image (FPPI) to show the classifier performance.

From the Detection Error Tradeoff (DET) curves indicated in Figure 6.12, one can deduce the following conclusion: HOG 1, HOG 2, HOG3 and HOG 4 have a similar performance which is much better than HOG 5. In addition, HOG 1 and HOG 3 have outstanding accuracy rates when the FPPW rate range from 10^{-4} to 10^{-2} (interesting

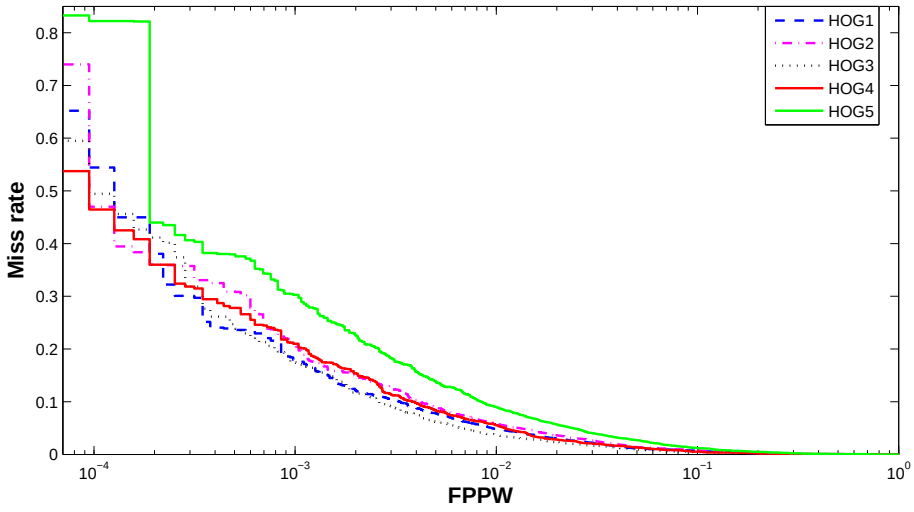


Figure 6.12: HOG+SVM classifiers with different parameter evaluations: Miss-rate vs False Positive Per Window (FPPW) curves. Lower curves also indicate better performance.

range). However, unfortunately, if we consider the detection speed shown in Figure 6.11, Hog 3's $343ms$ is comparatively slow when compared to HOG 1's $158.63ms$. On the whole, HOG 1 classifier performs better than the other four in terms of both detection speed and accuracy rate.

6.5 Double-layer classifier

6.5.1 Problems

All classifiers, regardless of AdaBoost algorithm or SVM algorithm mentioned above, have an excellently high detection rate in case of real-time detection; at the same time, the relatively high false positive rate has been troubling us as well. Hypothetically, if we drive a vehicle equipped with the above mentioned detection system, there would be continuous false alarms to remind us that "a huge group of moose appear on the road". We need to take steps to eliminate the false positive results as much as possible, even if

this means sacrificing a few true positive rates.

We shall assume the existence of two different conditions. The first one occurs if we adopt the single detector above; we can then detect 80% of the actual animals appearing in 10 consecutive frames caught by the camera; but 20% of frames will be misjudged if the consecutive frames do not contain any animals. The other condition is one in which only 75% of the animal rectangles are identified and all false positive results are rejected by an advanced system. We say the latter is more acceptable for our driving application, since it can be determined that the animal must exist in front of the road if we get more than 5 true result frames in a continuously received sequences of 10 frames. Then, we designed a double-layer detection system with a simple tracking method to improve the performance by significantly reducing the false positive rate.

6.5.2 Architecture design

Firstly, we need to emphasize that we designed the architecture of our system with two main criteria in mind: detection accuracy and recognition speed. The reasons behind adopting these criteria are explained as follows. We set the detection accuracy rate as a critical goal in order to decrease the false positive rate (false alarm); the reliability of the system will be enhanced as well. In addition, in order to get a real-time detection system which would work in a driving vehicle, the speed of recognition is adopted as a second design criterion, which enables quick target recognition. To achieve the aforementioned criteria, a double-stage system is suggested: see Figure 6.13.

In the first stage, we apply a fast detection algorithm which supplies the second stage with a set of regions of interest (ROIs) containing real animals and probably other similar objects (false positive targets). To fulfil the system requirements, the detection step of the first stage should be simple and fast because it is applied to the entire input frame. We choose AdaBoost as the first stage classification algorithm since it is faster and less complex than SVM. For that purpose, LBP adopts AdaBoost algorithm must be the best classifiers who meet the requirement of the first stage. Hence, it is selected

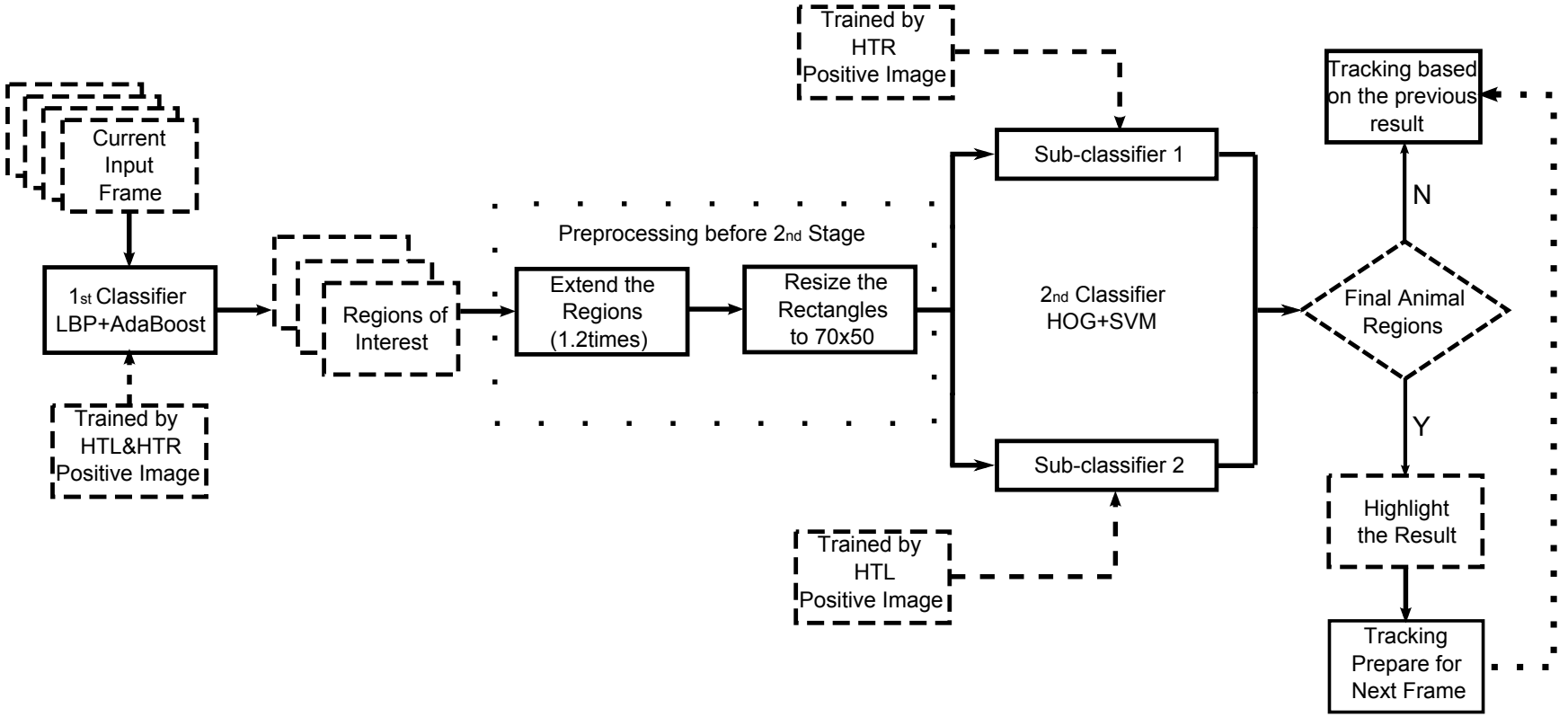


Figure 6.13: Architecture design of the two-stage animal detection system: a single LBP+AdaBoost classifier works as the first stage; two HOG+SVM sub-classifiers work as second stage.

as the classifier of the first stage. This first classifier is applied to the whole image to get ROIs that may contain animal targets. Furthermore, the first stage classifier is a single detector trained by all category positive databases (both HTL & HTR categories in our present study).

In the second stage, HOG+SVM is applied because of its excellent performance in pedestrian detection and its strong ability to eliminate the false positive results obtained from the first stage. To recognize the animals we want, HOG+SVM is applied to each ROI generated by the first stage. We choose the HOG 1 mentioned in the last section as our second stage classifier parameter. Specifically, Block Size: 2×2 cells (16×16 pixels); Stride Size: 8×8; Cell Size: 8×8 and Bins is 9. To improve the detection performance and to eliminate the false positive results as much as possible, the second stage classifier is composed of two sub-classifiers, where each one is used to recognize a well-defined side-view category of animal Head-to-Left (HTL) or Head-to-Right (HTR), see Figure 5.2. This means that the two sub-classifiers are trained by the HTL and HTR positive databases separately. We determine that a ROI obtained from the first stage is a true positive result if the region passed any one of the two sub-classifiers, and, the false positive ROI would be refused by both sub-classifiers.

Figure 6.13 illustrates that there is preprocessing work between the two layer detection, The reason will be explained in the following:

HOG+SVM detector usually can recognize a larger rectangle than the AdaBoost based detector according to our experience. Therefore, an unfortunate result happened where the red ROIs shown in Figure 6.14 obtained from the LBP+AdaBoost stage, were not large enough to be detected by the HOG+SVM classifier (green rectangle in Figure 6.14). To solve this problem, an expanding region (blue one) was inserted into the second stage instead of the initial ROIs. We have tried to extend the frame scale by 1.05 1.4 times and have found 1.2 times was big enough to cover the minimum detectable rectangle size for HOG+SVM.

The second extra work between the two stages consists of adjusting the extended

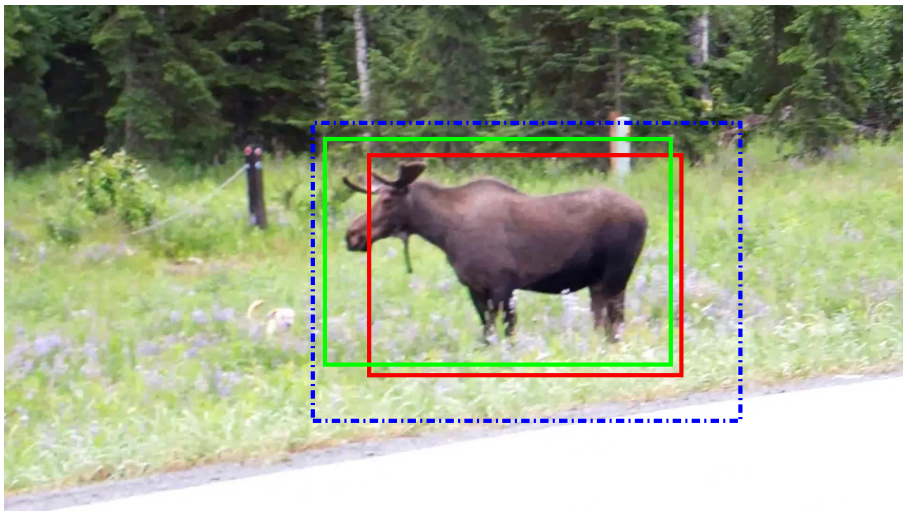


Figure 6.14: An example shows that a ROI detected by the first 1st classifier (red rectangle) does not cover the minimum detectable rectangle size (green one) of the 2nd stage classifiers, unless the ROI is extended to the blue region.

region to a certain size. The ROIs can also be very large which may cause extra-time to be processed by the second stage. For example, if the ROI is a very large moose with a size of 280×200, and the minimum detecting window of the second stage is set to 56×40 for a larger detection range, a large amount of detection time would be wasted on the incomplete animal body; consequently, a substantial time can be saved if we resize the 280×200 region to a smaller one. We then can choose 70×50 as our normalized size.

We find that almost all animals we need to detected are unicolor. If one frame does not detect any target but a previous frame has, a single colour filter tracking algorithm is applied to track the detected animal in that frame which will be introduced later.

6.5.3 Evaluation of double-layer detection system

We test our double-stage system based on speed and accuracy rate as well, and compare the results to those of single classifiers like LBP+AdaBoost and HOG+SVM. We refer to the double-layer detection system by its more abbreviated name of "(LBP+AB)/(HOG+SVM)".

Moreover, we have built another two layer system but changed the order (LBP+AdaBoost set as the second classifier after HOG+SVM, and named as (HOG+SVM)/(LBP+AB)) to facilitate a comparison.

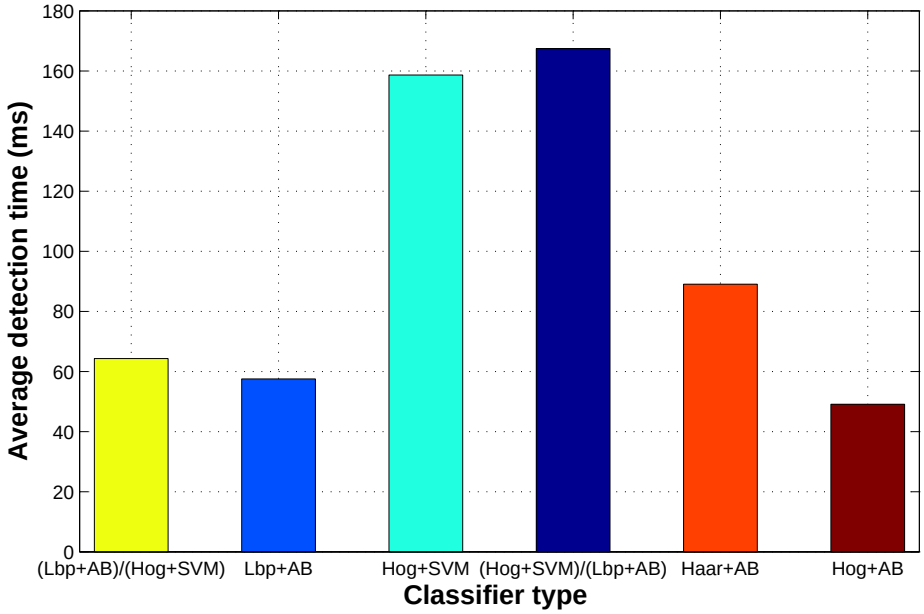


Figure 6.15: The average detection speed: LBP+AdaBoost cascade HOG+SVM is 64.32ms, HOG+SVM as first stage cascade LBP+AdaBoost is 167.45ms. (Video size: 320×240; total frame number: 1026; initial (minimal) detection window size: 56×40; scale rate: 1.05.)

On the one hand, the final average detection speed shown in Figure 6.15 is expressed in the bar chart. The left bar indicates our double-layer system detection time; this is only 7.2ms slower than the single LBP+AdaBoost classifier. As explained before, although HOG+SVM consumes a great deal of time compared with other classifiers, due to the preprocessing for ROIs, only an extra 7.2ms is spent on the second stage classifier. However, exchanging the order of the two stages leads to a completely different result: 167.45ms per frame would be the cost for (HOG+SVM)/(LBP+AdaBoost) detector, which is slower than the single HOG+SVM detector.

On the other hand, we evaluate the accuracy performance of our system by testing the database used before; and we calculate the final Detection Error Tradeoff (DET) curve expressed in "Miss Rate versus False Positive Per Image (FPPI)" curves. To reflect better the double-stage system performance, we plot the Miss rate vs FPPI curves of other classifiers together with the final curve in Figure 6.16. But, note that lower curves indicate better performance.

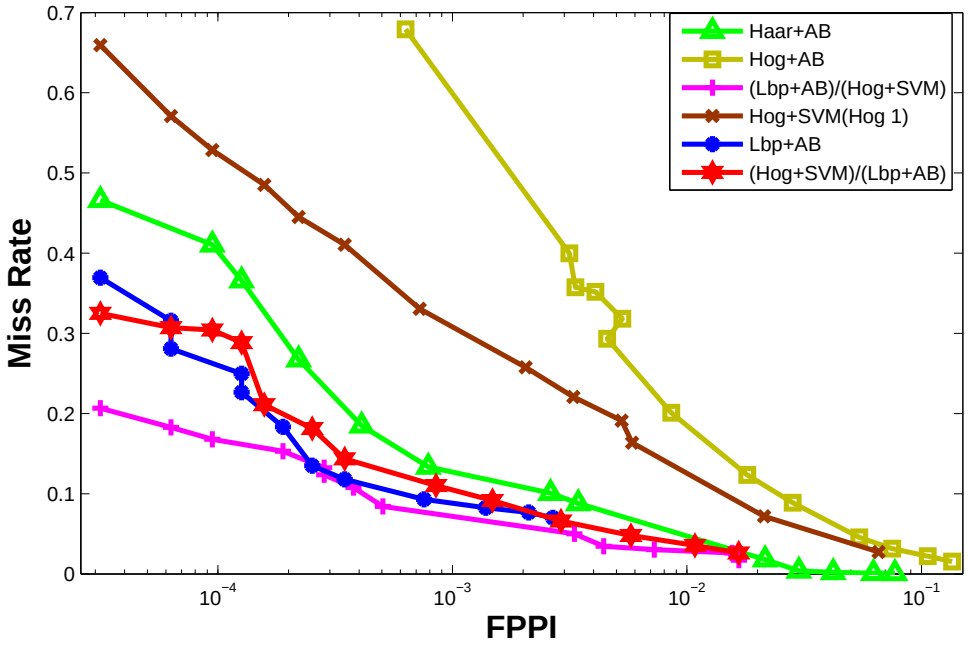


Figure 6.16: Miss rate vs FPPI curves used to demonstrate the double-layer animal detection system (the lowest curve has a better performance).

From the curve shown in Figure 6.16, one can draw the following conclusions. The double-stage classifier (LBP+AdaBoost)/(HOG+SVM) performs better than single classifiers and the reordered (HOG+SVM)/(LBP+AdaBoost) classifier. Especially, in the case where the false positive per image rate is around 1×10^{-4}, (LBP+AdaBoost)/(HOG+SVM) has a distinct advantage compared with other detectors. When the FPPI increases to more than 5×10^{-4}, the three classifiers, (LBP+AB)/(HOG+SVM), (HOG+SVM)/(LBP+AB) and single LBP+AdaBoost, have similar levels of accuracy. Moreover, Haar+AdaBoost

classifier achieves a middle level performance but still does much better than HOG descriptor classifiers: HOG+AdaBoost and HOG+SVM.

In the final section of this chapter, some image and video detection results will be posted to show the powerful and high performance of our double-layer system.

6.6 Tracking by Hue, Saturation and Value (HSV) colour space

As mentioned above, almost all the animal bodies we need to detect are unicolor. We applied a kind of unicolor target tracking algorithm based on HSV image to track the animal body we detected in previous frames. However, we note that the detection result should be trusted and the tracking method is just used in the assisting role. The following flow chart shown in Figure 6.17 indicates how the tracking process works in our system.

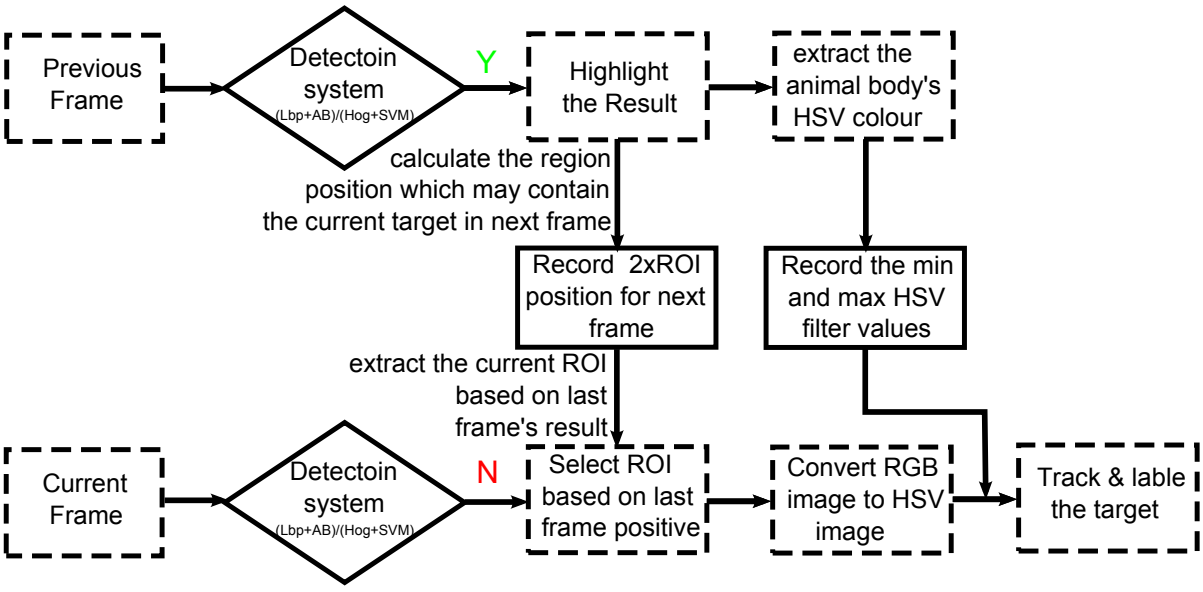


Figure 6.17: Tracking process: The tracking result will be executed only when the current frame does not detect the animal but a previous frame has

When an animal rectangle is detected by our double-layer system, as shown in the left-top image of Figure 6.18, the body sub-window will be converted to HSV image (left-bottom image in Figure 6.18) and the system will record the min and max HSV (3 channels) filter values. If the classifier does not recognize any rectangle in the next frame, a $4 \times resultrectanglesize$ of ROI determined by the last frame will be pushed to the tracking region, then the animal body will be recognized by the HSV filter thresholds.

Figure 6.18: RGB image & HSV image. Only the body part (black rectangle in the left-bottom image) was selected to calculate the HSV filter values. The white rectangle in the middle image is the ROI determined by the last frame detection result.

Under the tracking method assistance, we can not only track the undetected animal target, but we can also recognize nonstandard body shapes; this makes our system more powerful for animal detection. Figure 6.19 demonstrates some examples of tracking results in video detection, The recognized targets were labeled as "$\oplus$" at the centre of the animal body.

6.7 Nighttime detection

The report [50] shows that AVCs take place at all times over a 24-hour period. However, the most dangerous period for AVCs seems to be at nighttime, which is from $6:00\ PM$

Figure 6.19: Examples of video tracking result

to $8:00\ AM$. This leads us to pay more attention to nighttime detection. The night period leads to relatively weak illumination and a limited field of view, and its detection environment is very different from that of daytime. Nevertheless, our system can also work under the night condition.

Initially, a thermographic camera (or infrared camera) is applied to capture the animals in the nighttime environment. An image can be formed using the infrared radiation released from warm-blooded animals [21]. Instead of having a common camera form an image using the visible light with $450 - 750\ nm$ wavelengths, the thermographic camera operates in wavelengths as long as $14,000\ nm$. Figure 6.20 indicates the process of generating a thermal image.

Figure 6.21 illustrates some examples of animal thermographic images collected from infrared cameras. In the context of animal detection technology, there are two main differences between thermographic and common images. On the one hand, some similar

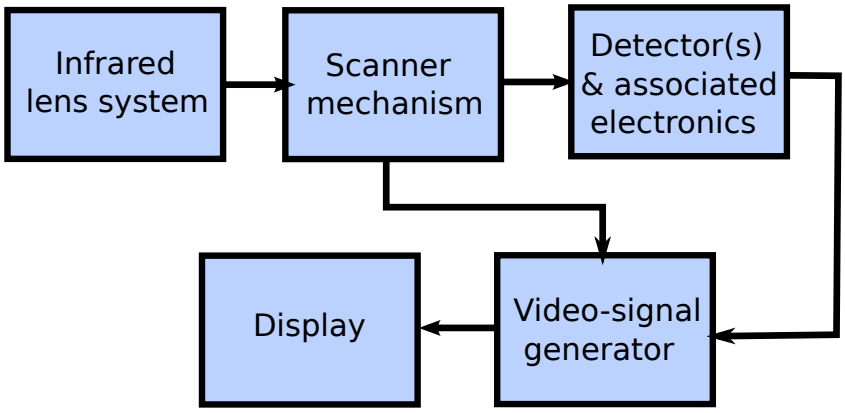


Figure 6.20: The main subunits that make up a thermal imaging camera [54].

objects, such as branches, signs, underbrush and some colour backgrounds, are filtered out by the infrared camera. This can lower the false positive rate automatically. On the other hand, animals texture features are papered over or undermined, but the profile features are strengthened. Therefore, the double-layer system needs some adjustments in order to adapt infrared cameras.

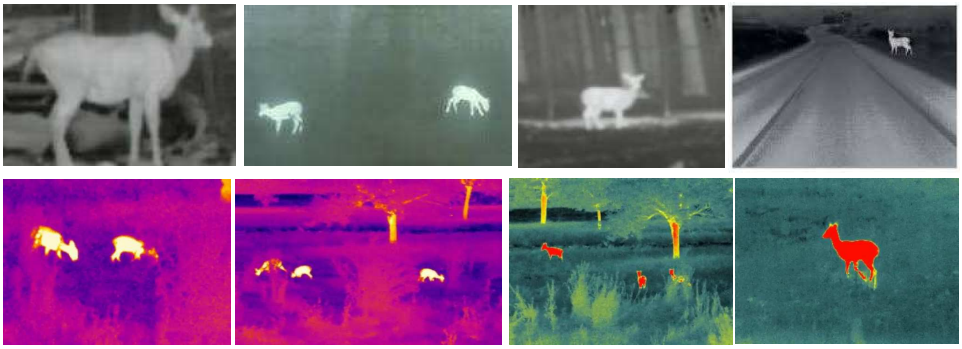


Figure 6.21: Examples of animal thermal images: the above four are generated from black and white infrared camera and the bottom four are from a colour one.

Firstly, the first stage classifier is substituted by HOG adopt AdaBoost classifier to obtain the ROIs from the input images. That is because HOG features have a stronger ability to describe object contour compared with LBP features. As mentioned earlier, the number of false positive results would not be too great since some of the false positive

rectangles are filtered by the infrared camera. We then retrain the two HOG+SVM sub-classifiers in second stage based on thermographic animal images; and, each sub-classifier also focuses only on one type of animal posture (HTL or HTR). Table 6.4 lists the training parameters of the two sub-classifiers.

No. of positive images for each posture	327
No. of negative images	6000
Window size	56×40
Block size	16×16
Stride size	8×8
Cell size	8×8
No. of Bins	9
Dimension	864

Table 6.4: Parameters of HOG which adopt SVM classifiers that perform in the second stage of detection in nighttime detection.

Since there are only 327 thermographic animal images collected in the positive database, the nighttime detection result is not as good as that of daytime detection. However, we predict that the final result would be better than daytime detection if we applied a greater number of positive images. This is because the image of the animal body has been strengthened by the infrared camera. Some nighttime detection results from images and videos are shown in Figure 6.23, Figure 6.26 and Figure 6.27

6.8 Experimental result

Firstly, we show some still image results to compare the results of the two stages. Figure 6.22 and Figure 6.23 illustrate the daytime and nighttime detection results in different weather conditions. The left images provide the output of the first classifier (LBP+AdaBoost) which contains the true animal image body and other false positive

ROI; the right images indicate the final detection results of the overall classifier. We can find that most of the false positive rectangles were eliminated by the second classifiers.

We then show some frame result cut from two typical continuous animal videos in which a moose is running beside the road side in a sunny scenario; and, in the second video, a moose is shown crossing the road in snowy weather. The results include the overall detector output which is marked as a green rectangle and the tracking result which is labeled as "$\bigoplus$". The first video, see Figure 6.24, contains plenty of moose shape categories which include "HTR", "HTL", front-view, rear-view, turn-back-view and so on. Some of the shapes cannot be detected based on the current system, which will be left to our future work. And the other video just contained one standard category of shapes: a black moose walking across a snow-covered road. However the camera filmed from near view to far view, see Figure 6.25.

At last, two nighttime animal detection video results are demonstrated in Figure 6.26 and Figure 6.27. Both of the videos are generated from infrared cameras.

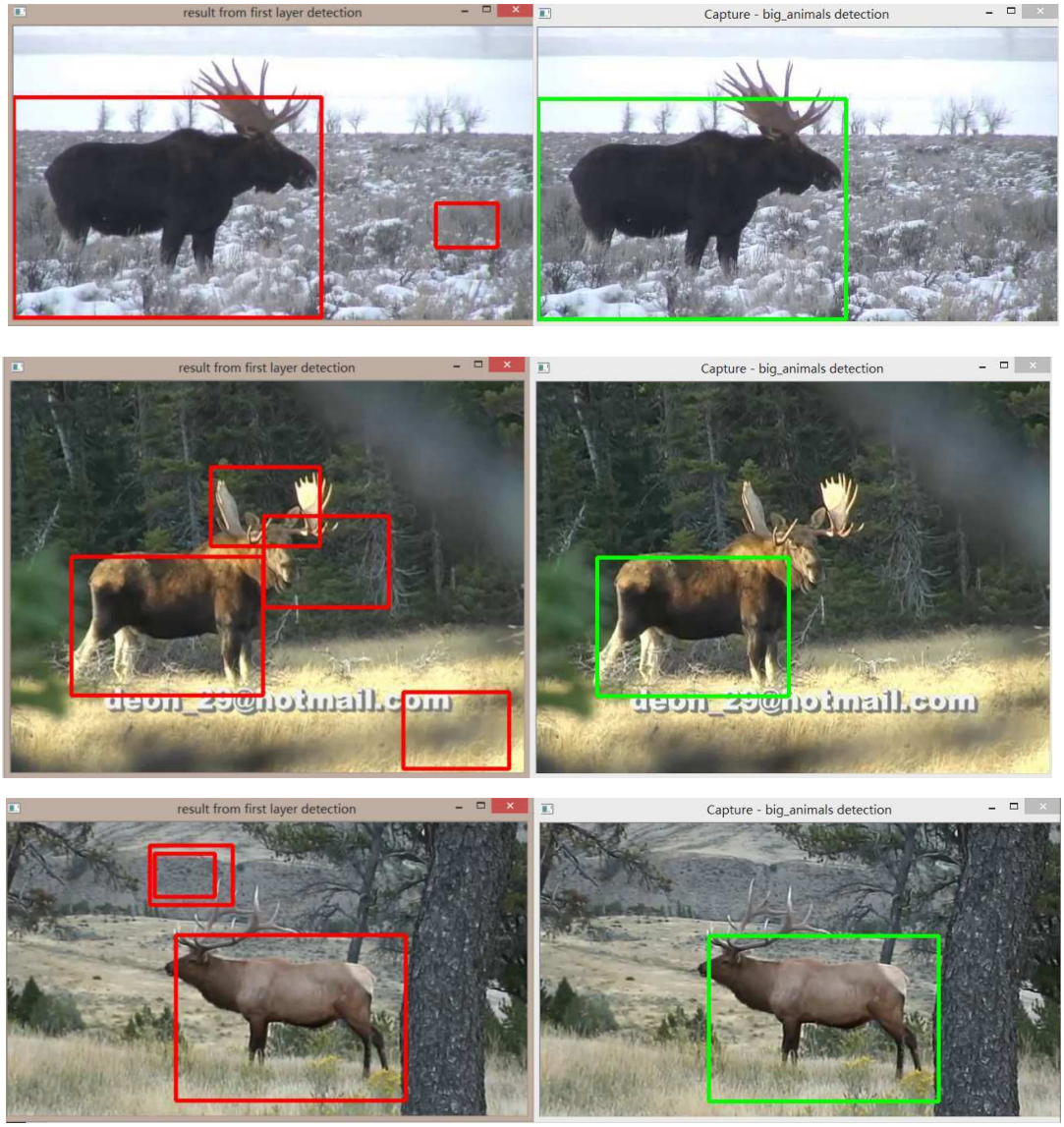


Figure 6.22: Daytime experimental results performed on still images in snowy, sunny and cloudy conditions. Left window: the output of the first classifier; right window: the detection result of the overall classifier.

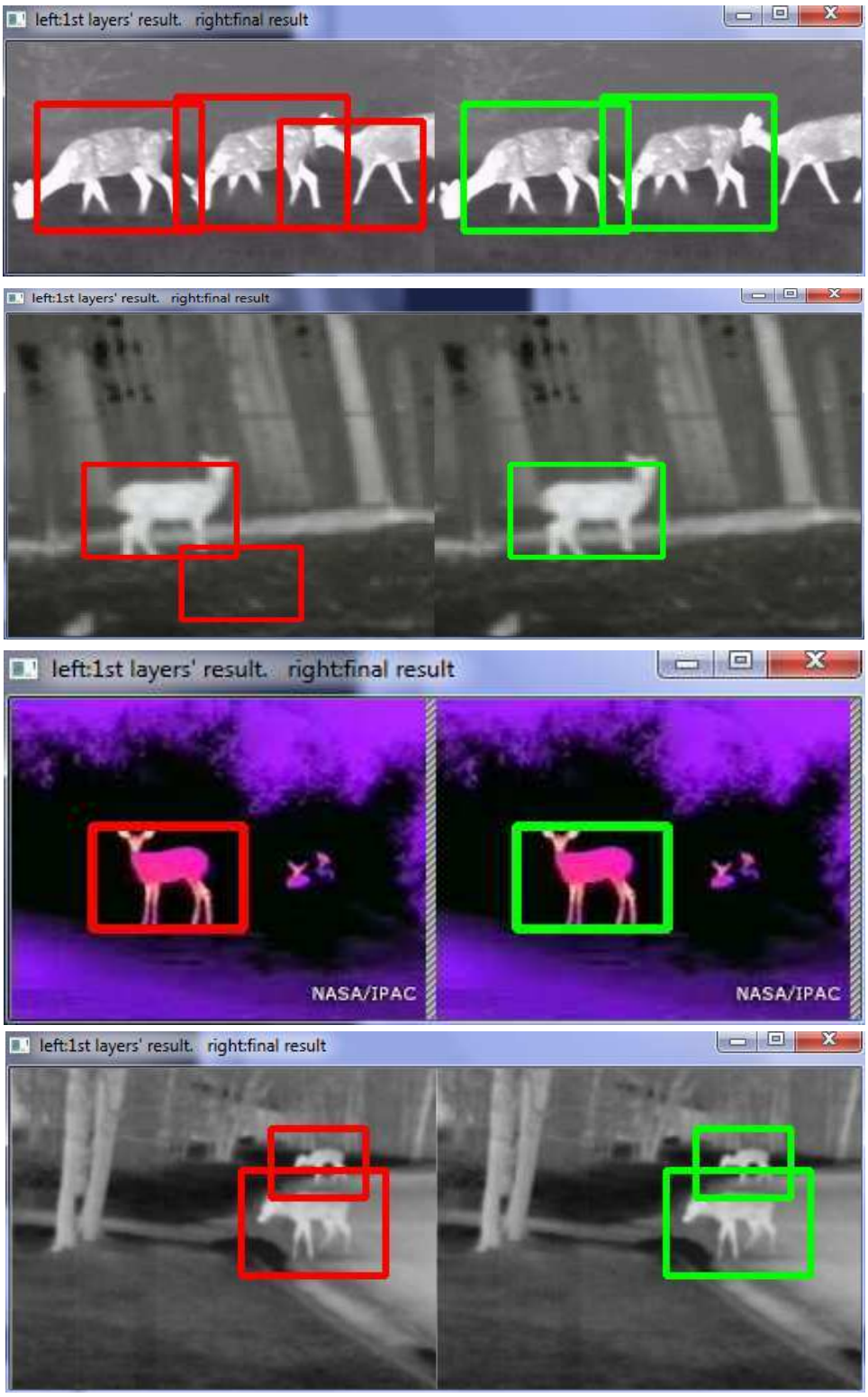


Figure 6.23: Nighttime experimental results performed on still images. Left window: the output of the first classifier; Right window: the detection result of the overall classifier.

Figure 6.24: Experimental results performed on videos in sunny conditions.

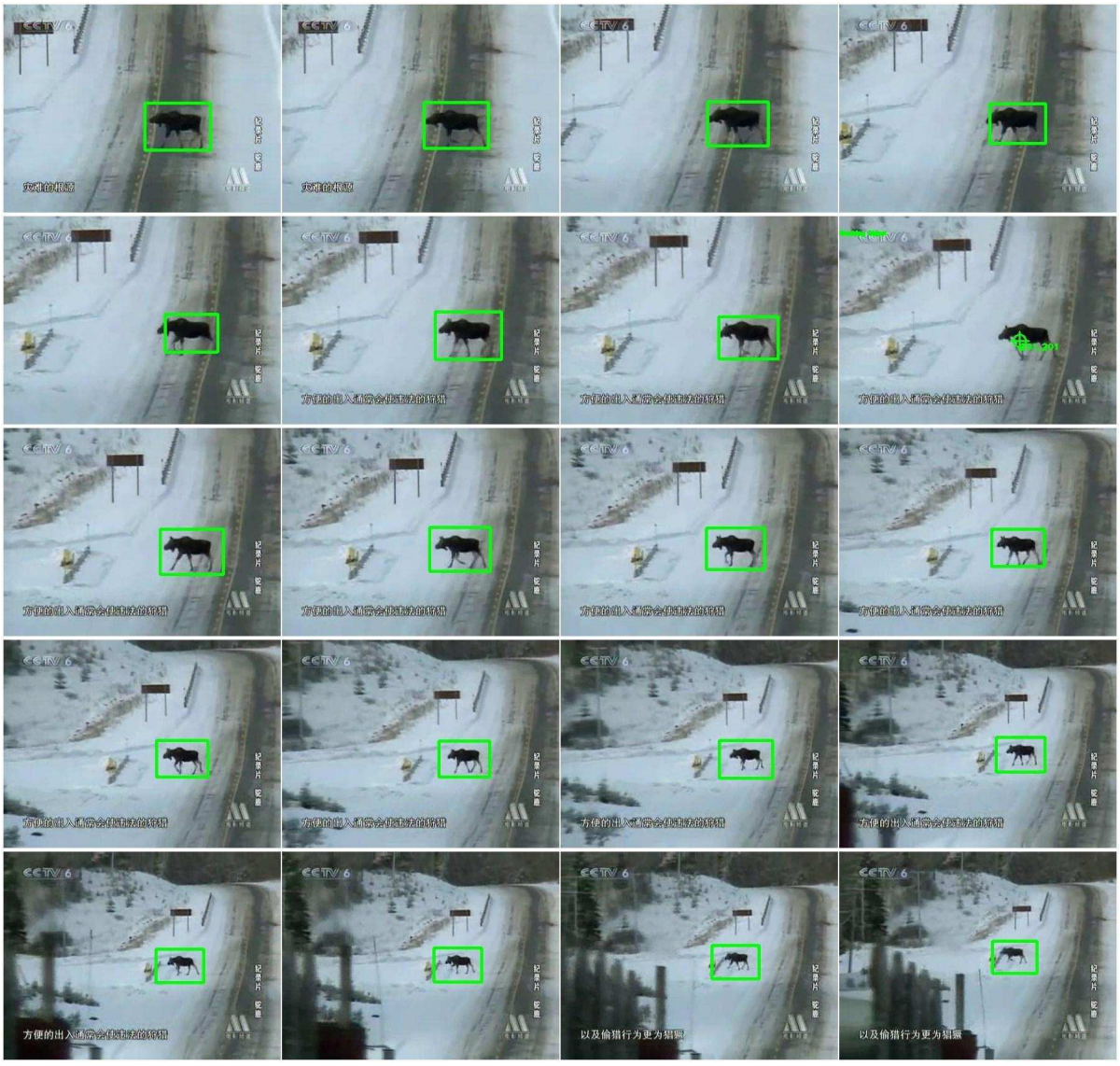


Figure 6.25: Experimental results performed on videos in snowy conditions.

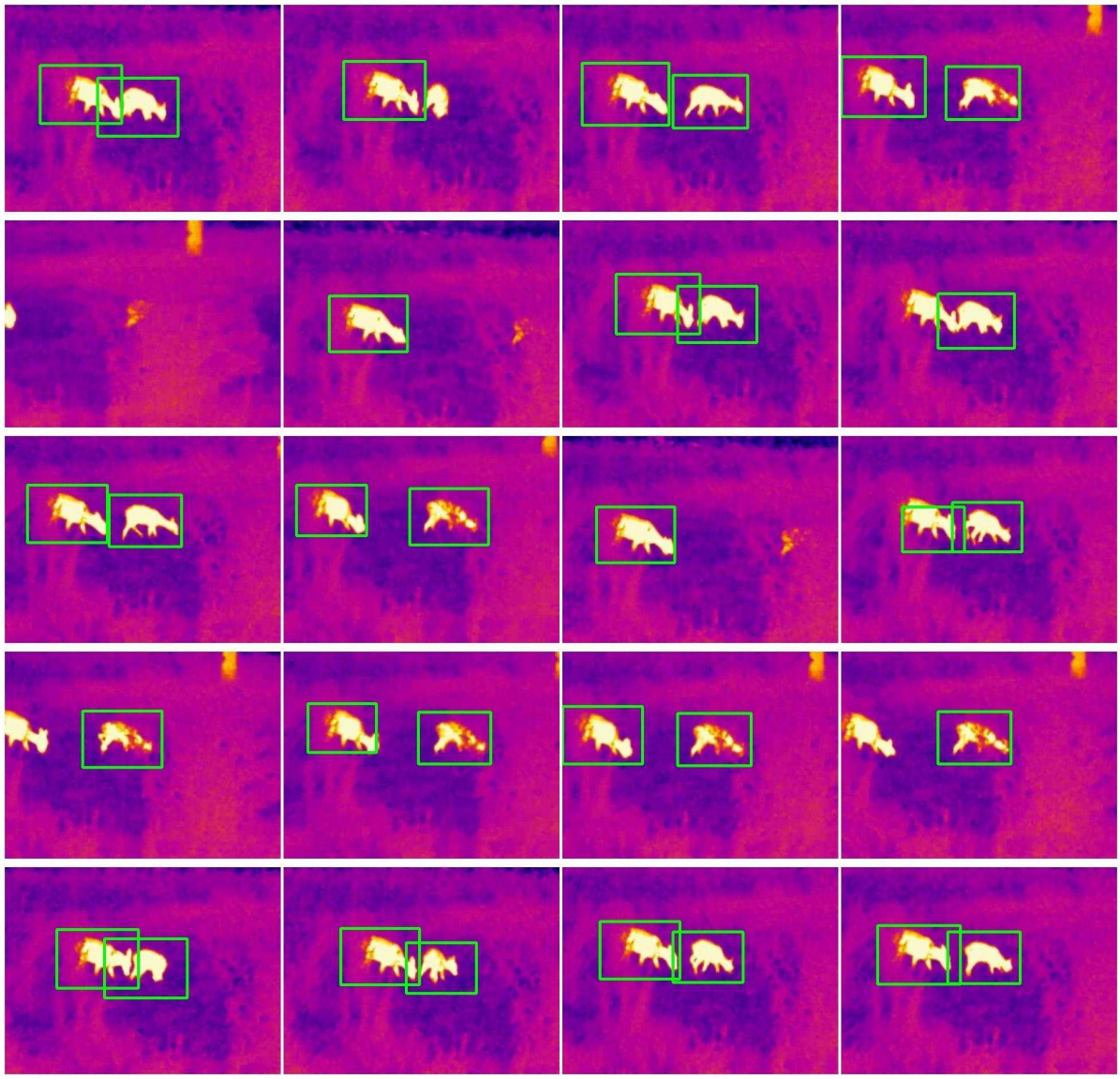


Figure 6.26: Experimental results performed on infrared videos of nighttime.

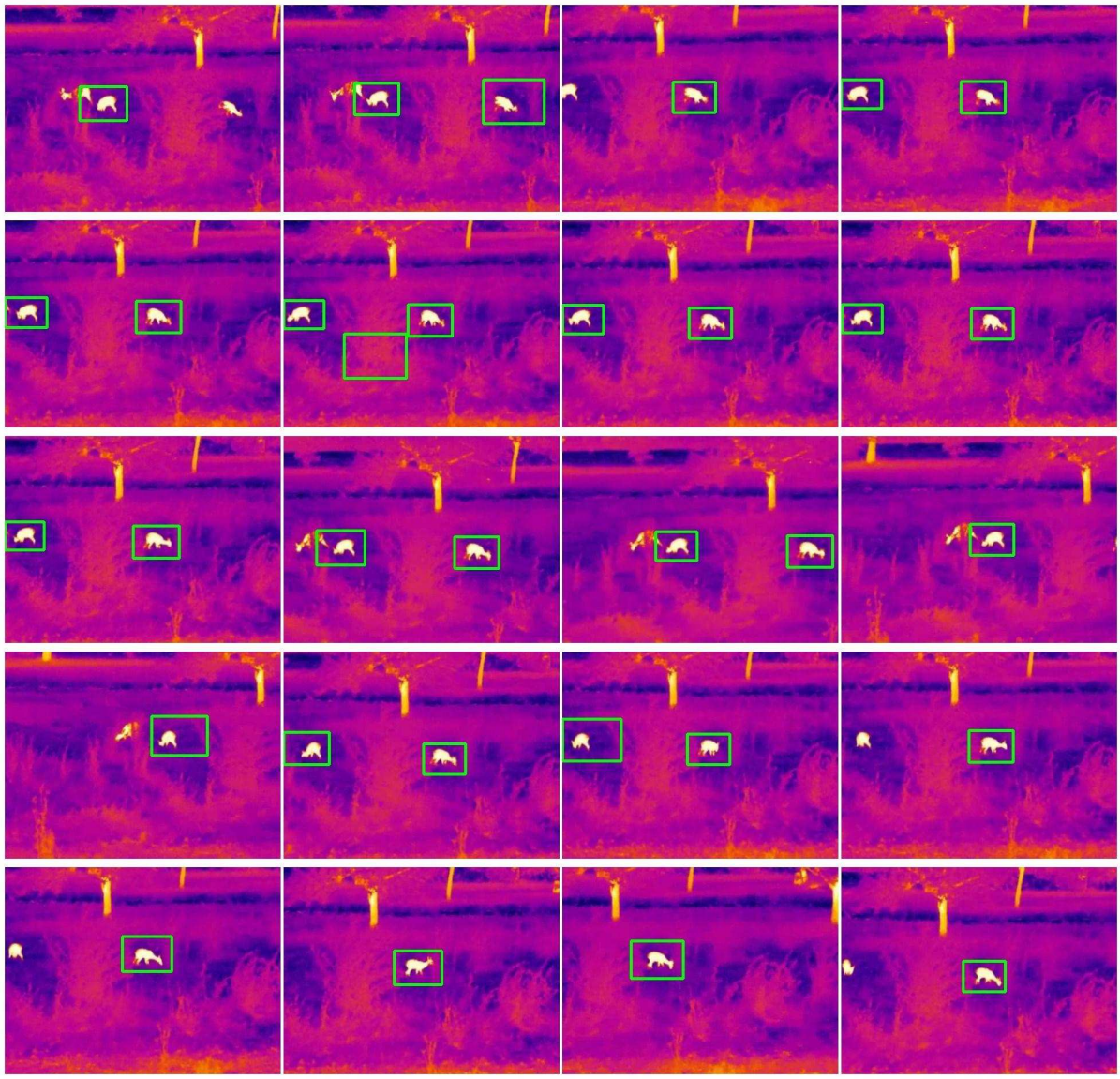


Figure 6.27: Experimental results performed on infrared videos of nighttime.

Chapter 7

Energy consumption

Over the last few years there has seen a tremendous growth in mobile device applications. Smart phones and other hand-held, vehicle-equipped mobile devices are becoming increasingly powerful, and are now used for much more than basic communications. Based on Moore's Law, the doubling of transistors every 18 months has now resulted in the doubling of system level components. For instance, the current smart phone processing power is no less potent than high-end laptops. However, in contrast to the rapid development of hardware and computing power, battery durability is the weak point in mobile devices. Therefore, energy efficiency becomes the critical problem for mobile application development. Since our animal detection system will be built not only in driving vehicle, but also on roadside devices. Energy efficiency should be another critical criterion besides real-time and the detection accuracy factors.

In this chapter, we will continue to compare the image descriptors and classify algorithms, taking the energy consumption as a key factor.

7.1 System

Our double-layer animal detection system was based on PC platform as shown in the following:

Hardware components:

- Processor: Intel® Core™ i5 − 2450M cpu @2.50GHz × 2
- Memory (RAM): 4.00 GB
- Graphics Chipset: AMD Radeon HD 7650M

And software environment:

- Microsoft Windows 7 64-bit Professional with Service Pack 1
- Microsoft Visual Studio 10 professional
- OPENCV 2.4.3

The PC platform based system helps us to obtain experimental data and to make result comparisons. But, it is inconvenient to test the energy consumption since Windows OS is complicated and a great number of unrelated processes will affect the result. Moreover, the final vehicle system will be built in an embedded system instead of in a PC. Therefore, we chose to transplant our detection system into a mobile device to perform the energy test.

7.1.1 Transplanting from PC to ANDROID device

Over the recent years, the evolution of mobile devices, especially the ANDROID OS devices, have drastically changed people's lives. Android has gone beyond IOS and has retained its leading position in the smart phone industry since the end of June 2013 [17]. An ANDROID mobile device is often expected to offer computer-like functionality. In addition, some applications such as smart home system [40], smart wheelchairs [29], and the Internet of Things [7] have a wide-range of bright prospects. Therefore, ANDROID based Advanced Driver Assistance System (ADAS) will also become a hot spot in the near future.

ANDROID is a popular operating system based on the Linux kernel and was designed primarily for embedded or mobile devices such as laptops, tablets and smart phones. This OS was researched and developed by Android Inc., which was purchased by Google in 2005. Since the first version of Android OS (Linux kernel version 2.6) was unveiled in 2007, more than one million applications developed by worldwide developers have been applied in hand-held devices and on other embedded devices.

Therefore, we have decided to transplant the animal detection system from PC Windows platform to Android devices. Here are the advantages that explain why we have chose android devices:

- **Open source.** Android is a kind of open-source OS, and Google releases the source code under the Apache License. The open-source code and permissive licensing allows the system and applications to be freely distributed and modified by any enthusiastic developers and device manufacturers. In addition, we do not need to pay extra fees to apply Android OS to our future smart vehicle system.

- **Compatibility.** As we know, the purpose of Android is to establish an open and accessible platform for developers to build innovative applications. The object detection systems would relate to other applications such as communication and networks in intelligent vehicular networks. Android compatibility has defined the technical details of the common platform; it has furthermore provided plenty of tools used by Original Equipment Manufacturers to make sure that the Android based applications can run on a variety of devices and systems. All of these give us a foundation for further application development. In addition, the animal detection system was based on "OPENCV" environment which is another open source library in the image processing area. OPENCV is compatible with Android development which has made our transplant more convenient.

- **Android devices**. First of all, a mobile phone has all the necessary components we need for energy consumption tests of animal detection, such as camera, display

functions and independent battery. It is very convenient and efficient compare with other embedded systems. Secondly, the image processing technology requires a relatively high data computing ability: the new generation of android devices provide powerful hardware support (CPU and RAM). Finally, an android smart phone is easy to get and the cost is lower.

Before we begin transplanting and energy testing, it is necessary to have a general understanding of how Android systems and applications work. Figure 7.1 illustrates the system level view of Android application operations.

Initially, Android OS was designed based on the Linux kernel, the bottom level of Android is composed of device drivers development which is the same as other typical Linux device drivers. Second, Hardware abstraction layer (HAL) works as a standard interface that allows the Android system to call the device driver layer when being agnostic about the lower-level implementations of drivers and hardware. After that, nearly all functionality has be applied to some sort of system service such as video/audio service, search service, and etc. System services are divided into two groups; the system services contains Activity Manager, Window Manager, Package Manager and media services including Audio, camera, content, and so on. Binder Inter-Process Communication system then connects the application framework with the Android system services code. It is important to note that, most application developers like us work at the Application framework level [1].

7.1.2 Environment building

The previous two-layer animal detection system design is based on C++ and OPENCV environment. However, Android APP development combined with OPENCV is based on JAVA and the relative Software Development Kit (SDK). It is much more complicated than the C++ environment, and we need to build the development environment before transplanting the system. We list the processes as follows:

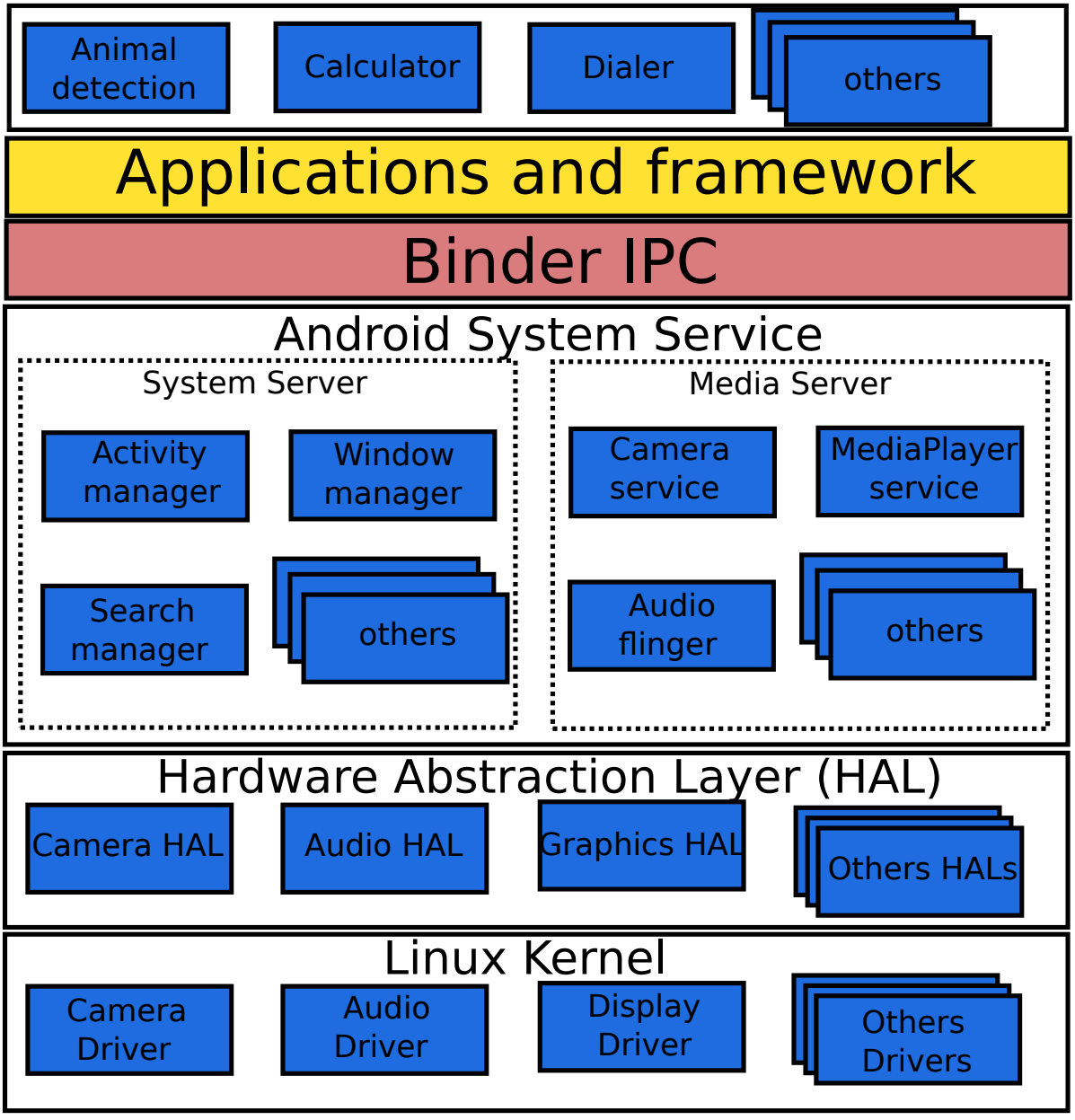


Figure 7.1: Android system architecture [1]. Five layers: Linux Kernel, HAL, Android System Service, Binder IPC, and applications and framework.

- The new development environment also works in Microsoft Windows 7 64-bit Professional OS.

- Build the JAVA environment which including JAVA Development Kits (JDK) installation, environment variable configuration and ECLIPSE installation.

- Install Android Development kits Android SDK and set up its environment variables. Add Android Development Tools (ADT is a plugin designed to develop Android applications in Eclipse integrated development environment) in Eclipse: Install New Software with "Name:ADT" and
  "Location: http://dl-ssl.google.com.android/eclipse". At this point, the Android development environment is finished. The following is for OPENCV environment.

- Install Android NDK for Windows 64-bit and set up its environment variable. The NDK is a useful toolset which helps developers to implement the native-code languages (C/C++) in the Android environment.

- Install Cypwin in Windows OS. It is a large collection of tools which simulate Linux distribution-like functionality in Windows. This is necessary beacuse all OPENCV's C/C++ complication needs Linux's gcc environment. Cygwin can compile C++ code and generate it as an "os" dynamic library which can be called by Java. OPENCV for Android already has a JAVA version's API. However, it is not comprehensive enough and some of the modules do not work well. That is why we need to use C++'s API.

- Install ADT plugin to achieve C/C++ code; and, JAVA code can be compiled simultaneously. After that, Install Sequoyah plugin to make the Android project support for the Native development.

- Add OPENCV for Android SDK in the system and import the existing OPENCV projects into Eclipse's workspace.

- At this point, OPENCV for the Android environment has been completed; we can now build our project in the JAVA environment. It should be noted that when we finish the project building, we need to apply Cygwin to compile the C++ code. This is because the project will be related to JNI (Java Native Interface) programming.

After we set up the development environment, we begin to build our energy test for the animal detection application. The APP is named as "mOOse" and we add human-face detection and traffic sign detection (Canada's speed limit signs are only included) in the APP. We design four main activities and corresponding layouts in our APP: Welcome page, target selection, method selection and detection interface, see Figure 7.2. All classifiers used in the APP are trained in PC and C++ platform.

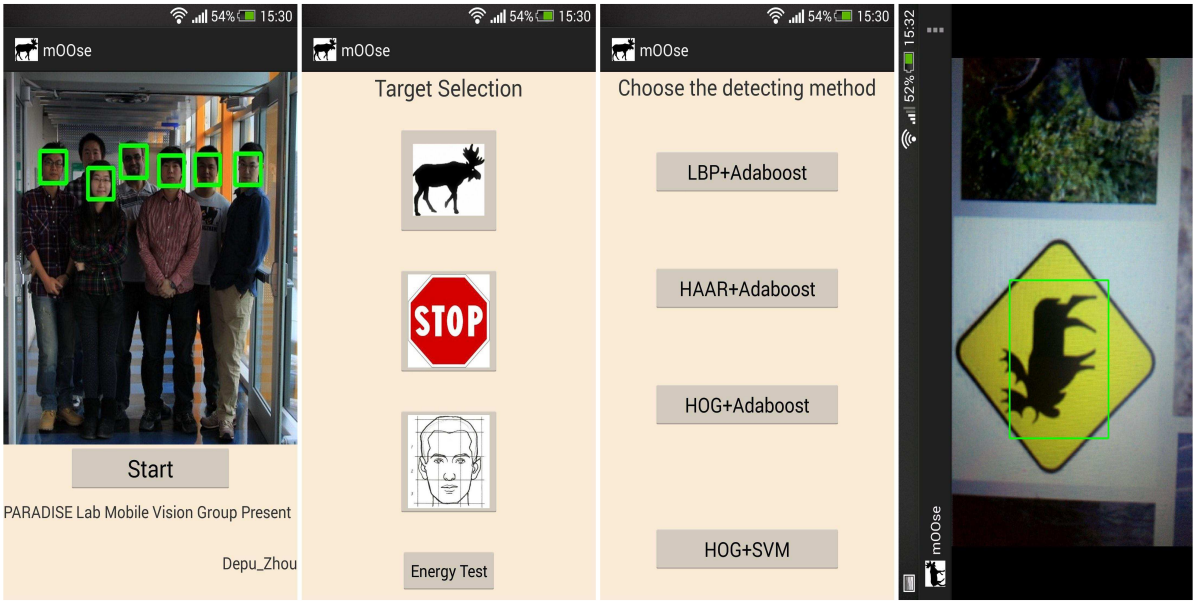


Figure 7.2: The four activities in "mOOse" APP: start pages, target selection (animal, human-face and traffic signs), method selection (HOG, Haar, LBP adopt AdaBoost and HOG applied SVM), and detection process.

7.2 Energy consumption test

7.2.1 Method and relative software

We choose two different Android devices to perform the energy test. HTC ONE and Google Nexus 5. Here is the information regarding the two devices including related hardware and software:

- HTC ONE:

  CPU Speed: Qualcomm® Snapdragon™600, quad-core, 1.7GHz

  Display: 4.7 inch, Full HD 1920×1080, 468PPI

  RAM: 2GB DDR2

  OS: Android 4.3 with HTC Sense ™

  Camera(rear): HTC UltraPixel Camera 4MP

  Battery: Li-polymer 2300 mAH

- Google Nexus 5

  CPU Speed: Qualcomm® Snapdragon™800, 2.26GHz

  Display: 4.95 inch, Full HD 1920×1080, 445PPI

  RAM: 2GB

  OS: Android 4.4(KitKat®)

  Camera(rear): 8MP

  Battery: Li-polymer 2300 mAH

An appropriate method is needed to perform the energy tests. As shown in Figure 7.3, we fix the smart phone in front of a monitor and keep them stable. A moose video played on the monitor repeatedly to provide the input video for the smart phone. We use the rear camera of the smart phone to shoot the video and to detect through one

of the four detection methods. After the beginning, we record the energy consumption data every 10 minutes and take samples 6 times in a total of 60 minutes for each method (the second device, Nexus 5, is sampled every 5 minutes over a period of 30 minutes). To make sure the video content is the same for each algorithm, we use a moose video for only 1 minute and play it in continuous loop. Figure 7.4 provides two screenshot images when the application was running in front of the monitor.

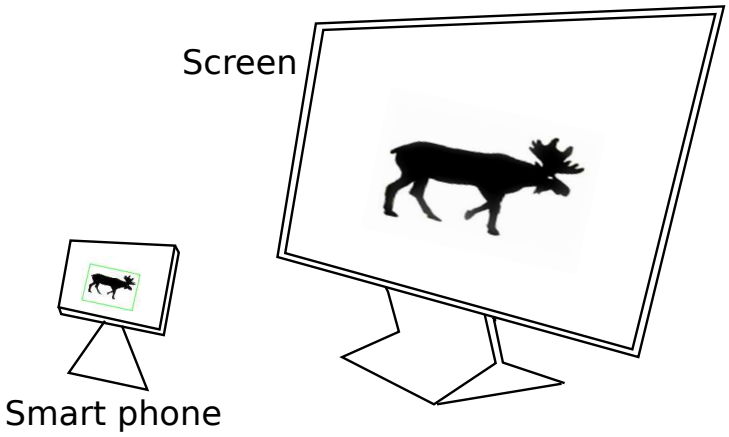


Figure 7.3: Energy test method.

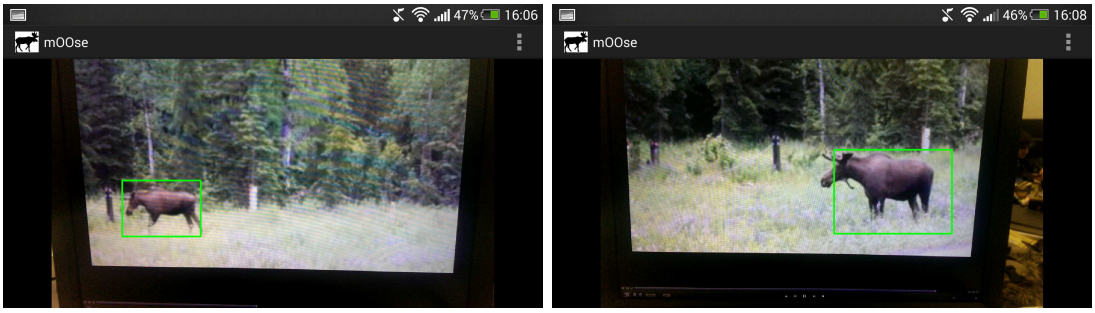


Figure 7.4: Detection results in "mOOse" APP. The mobile phone was fixed in front of the monitor and its rear camera captured the image.

After the method designed, we determine that a software (APP) for the purpose of recording the power consumption is indispensable; this is because the system "battery level bar" is not accurate enough. "Power Tutor 1.4" is an appropriate real-time sys-

tem/application power monitoring software which can record and display the power usage information for a single APP. It also can perform an analysis of the power consumed by major system components such as CPU, network interface, display, etc. Figure 7.5 shows the interface for Power Tutor and some of the results of power consumption. The two left graphics illustrate the power usage information for different APPs, the information includes the proportion of power usage, running time, and energy consumption (mW). The third pie chart indicates the average power usage of device components. The last graphic is the real-time consumption of components such as CPU, LCD, WIFI module and the total consumption.

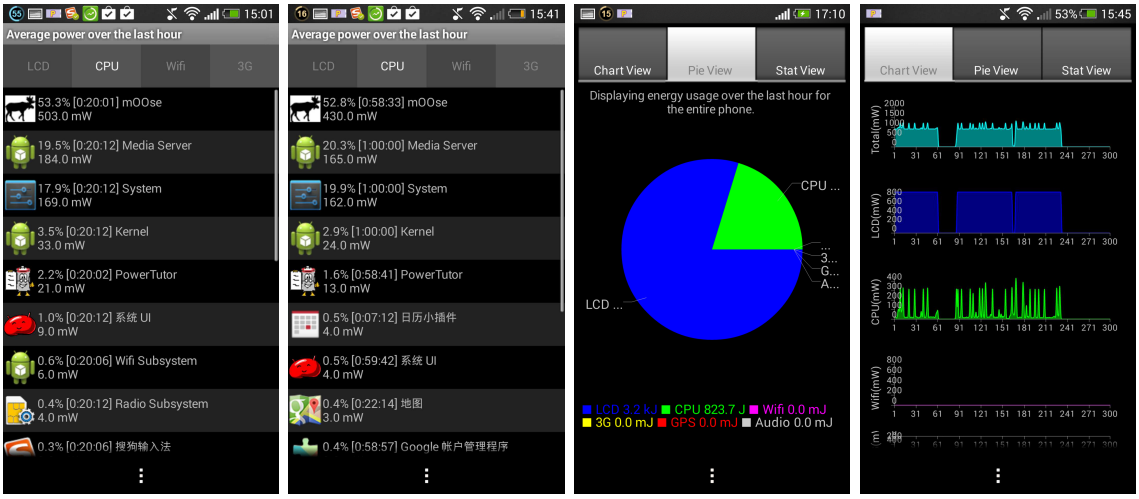


Figure 7.5: Power Tutor interface and some examples of power consumption results

Additionally, in order to exclude the maximum external interference, we terminate all unrelated processes and functions before testing. For example, we disable Wi-Fi, Bluetooth, Mobile Data, and Audio, and close all the software and system applications except “mOOse”, “Power Tutor”, and camera module. In this case, we assume that most of the power is consumed in related works.

7.2.2 Result and analysis

We perform four group of tests separately based on four detection methods in both devices and we draw the accumulated energy consumption (*Joule*) in 60 (HTC ONE) and 30 (Nexus 5) minute curves, as shown in Figure 7.6, and Figure 7.7 screenshot images indicate the average power (mW) over the experiments.

We draw the following conclusion: the energy consumption results are consistent with the speed test result. In other words, a faster detector consumes less energy, and vice versa.

When comparing the two classification algorithms, AdaBoost is significantly more energy-efficient than SVM algorithm. The average power of HOG+SVM classifier in HTC ONE is 430 mW in 60 minutes, that is 80 mW higher than the Haar+AdaBoost classifier's 350 mW. The reason is HOG+SVM detector needs to compute all HOG operators for each pixel of each block and each cell. After that, the whole detection window will generate a high dimensional histogram vector; and, a high-dimensional vector multiplication will be executed to perform the classification. All of these processes will cost an amount of energy, time and memory. To make matters worse, all the sub-windows needing to perform the detection will execute the whole detection process no matter what windows are positive or negative. However, when we apply AdaBoost as the classification algorithm, no matter which kind of feature (weak classifiers) we use, only the critical features, which were selected by the training process, are extracted and computed in the detection process. Moreover, the cascade structure classifiers will reject the sub-windows in the very beginning steps when the sub-windows are negative. That is why SVM will cost more energy and time.

When it comes to the different image descriptors adopted by AdaBoost algorithm, the Figure 7.7 and Figure 7.6 indicates that HOG+AdaBoost detector has the most energy-saving features, with only a 222.0 mW; and, LBP detector consumes 276.0 mW which is less than Haar detector's 350.0 mW.

Initially, LBP and Haar features have a lot common points. However, LBP features

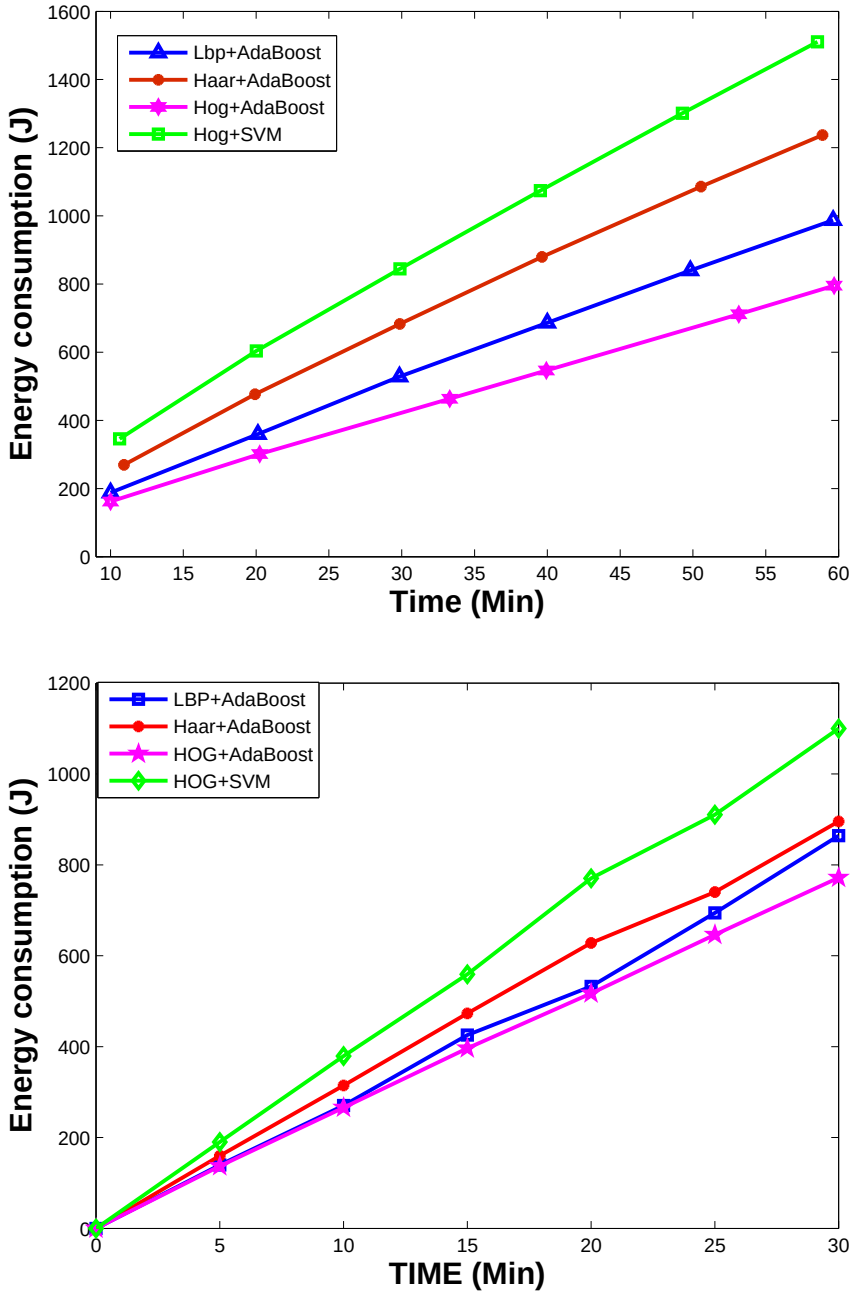


Figure 7.6: Energy consumption results. The first graph is the result of HTC ONE, including 6 samples with an interval of 10 minutes. The second graph is from Nexus 5, it includes 6 samples with an interval of 5 minutes.

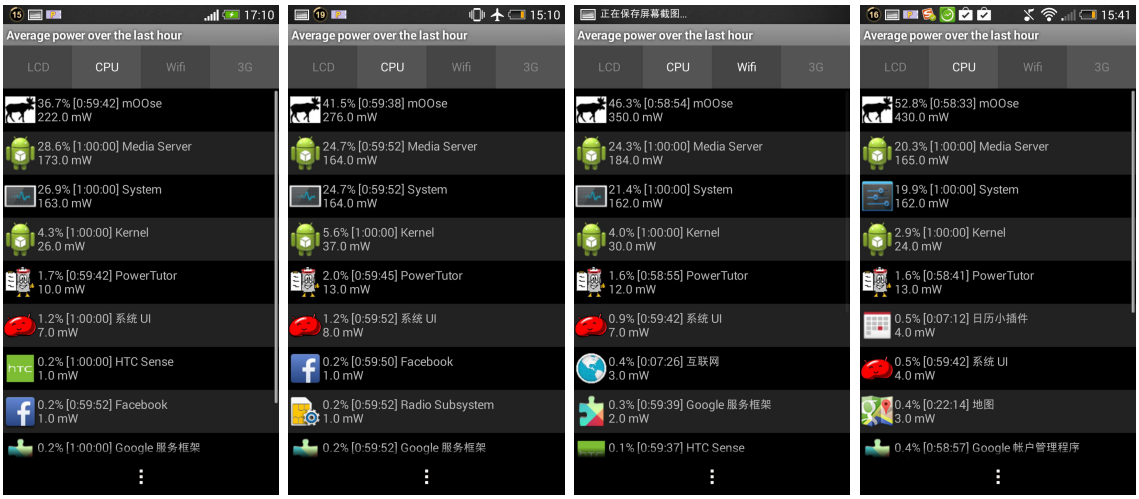


Figure 7.7: Average Power consumption over 60 minutes (mW): from left to right are shown HOG+AdaBoost, LBP+AdaBoost, Haar+AdaBoost and HOG+SVM. We can see that other system applications such as "Media Server", "System", "Kernel", and "Power Tutor", consume quite limited and similar power usage over the last 60 minutes.

are more efficient than Haar, since LBP descriptors contain more contrasting rectangles intensity information than Haar features. Therefore, the number of critical LBP features is dramatically less than the chosen Haar features in each cascade stage. For instance, 11 and 14 Haar features were applied in stage 0 and stage 1 of the Haar+AdaBoost system; these figures are only 5 and 6 respectively in LBP+AdaBoost. In addition, LBP is a kind of binary feature compared to the large integer number of Haar features. Therefore, Haar features need larger computations, which need more time, memory and energy.

However, both LBP and Haar feature calculations are more complex than those of HOG features when the detection window is very small (28×20 in AdaBoost classifiers). Haar and MB-LBP both work on adjacent rectangles or blocks, even integer image can accelerate the computation of an rectangle pixel sum values. However, HOG feature just aims at the gradient of pixels in small cells and a bin value of each cell is extracted as a weak classifier. So it is thus more energy efficient. By the way, our HOG+AdaBoost classifier just works with pixels of small cells and blocks, which leads HOG+AdaBoost

to perform not well enough when compared with Haar and LBP classifiers adopted by AdaBoost.

7.3 Energy-saving measures

In practical applications, some measures will be effective to reduce the energy consumption of image processing based Advanced Driver Assistance System (ADAS).

For example, it is redundant to process every frame in the real-time detection. Every one frame in a consecutive flow of two or three frames is sufficient for animal detection. Or, we can determine the detection frequency based on vehicle speed, a faster speed leads to a higher detection frequency. Moreover, the scale rate of our detectors is 1.05. If we increase the scale rate to 1.1 or 1.2, the detection speed will increase a great deal and the energy consumption will decrease as well. Of course these methods will also reduce the detection accuracy.

In addition, displaying the detection results also consumes large amounts of energy. It is unnecessary to keep displaying the image result continuously; a wise choice is to only display a result that captures an animal image or to just output the detection positive or negative result.

Bibliography

[1] Android low-level system architecture, Last time accessed on: 2014/03/14. URL: http://s.android.com/devices/index.html.

[2] Opencv (2.4.3-2.4.9) documentation, Last time accessed on: 2014/04/11. URL: http://docs.opencv.org/.

[3] Wildlife damage management, Last time accessed on: 2014/04/24. URL: http://www.aphis.usda.gov/wildlife_damage/nwrc/spotlight/deer_vehicles_Aug09.shtml.

[4] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[5] Tilo Burghardt and Janko Calic. Real-time face detection and tracking of animals. In *Neural Network Applications in Electrical Engineering, NEUREL. 8th Seminar on*, pages 27–32. IEEE, 2006.

[6] Yunyun Cao, Sugiri Pranata, and Hirofumi Nishimura. Local binary pattern features for pedestrian detection at night/dark environment. In *Image Processing (ICIP), 18th IEEE International Conference on*, pages 2053–2056. IEEE, 2011.

[7] Hsin-Yi Chang. Power-saving internet of things architecture for smart living applications. Master thesis, National Taiwan University of Science and Technology, 2013.

[8] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, CVPR. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[9] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):743–761, 2012.

[10] Centers for Disease Control, Prevention, et al. Nonfatal motor-vehicle animal crash-related injurie, united states, 2001–2002. *Journal of safety research*, pages 675–678, 2004.

[11] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156, 1996.

[12] Mary Gray. Advances in wildlife crossing technologies. *Public Roads, U.S. Department of Transportation*, 73(2), 2009.

[13] Feng Han, Ying Shan, Ryan Cekander, Harpreet S Sawhney, and Rakesh Kumar. A two-stage approach to people and vehicle detection with hog-based svm. In *Performance Metrics for Intelligent Systems 2006 Workshop*, pages 133–140, 2006.

[14] Marcel Pieter Huijser, Tiffany Holland, Matt Blank, Mark Greenwood, Patrick Tracy McGowen, Barrett Hubbard, and Shaowei Wang. The comparison of animal detection systems in a test-bed a quantitative comparison of system reliability and experiences with operation and maintenance: Final report. Technical report, Western Transportation Institute, 2009.

[15] Marcel Pieter Huijser and Patrick Tracy McGowen. Overview of animal detection and animal warning systems in north america and europe. *Western Transportation Institute, Montana State University*, 2003.

[16] Marcel Pieter Huijser, Patrick Tracy McGowen, Julie Fuller, Amanda Hardy, and A Kociolek. Wildlife-vehicle collision reduction study: Report to congress. Technical report, 2007.

[17] Vinamra Jain and Ashok Sharma. The consumers preferred operating system: Android or ios. *Department of arketing, Amity Business School, Amity University*, 2013.

[18] Tony Jebara. *Machine learning: discriminative and generative (Kluwer International Series in Engineering and Computer Science)*. Kluwer Academic Publishers Norwell, MA, USA. ISBN:1402076479, 2003.

[19] Christoph Gustav Keller, Christoph Sprunk, Claus Bahlmann, Jan Giebel, and Gregory Baratoff. Real-time recognition of us speed signs. In *Intelligent Vehicles Symposium*, pages 518–523. IEEE, 2008.

[20] Pooya Khorrami, Jiangping Wang, and Thomas Huang. Multiple animal species detection using robust principal component analysis and large displacement optical flow. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR), Workshop on Visual Observation and Analysis of Animal and Insect Behavior*, Tsukuba 11-15 November 2012.

[21] Paul Klocek. *Handbook of infrared optical materials*. MARCEL DEKKER, INC., Library of Congress Cateloging-in-Publication Data, 1991.

[22] Keith K Knapp, Xin Yi, Tanveer Oakasa, Wesley Thimm, Eric Hudson, and Chad Rathmann. Deer-vehicle crash countermeasure toolbox: a decision and choice resource. Technical report, Midwest Regional University Transportation Center, Deer-Vehicle Crash Information Clearinghouse, University of Wisconsin-Madison, 2004.

[23] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *Image Processing. Proceedings. International Conference on*, volume 1, pages I–900. IEEE, 2002.

[24] David Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[25] Topi Mäenpää. *The Local Binary Pattern Approach to Texture Analysis: Extenxions and Applications, pages 33–36.* Oulun yliopisto, 2003.

[26] Juan E Malo, Francisco Suárez, and Alberto Díez. Can we mitigate animal–vehicle accidents using predictive models? *Journal of Applied Ecology*, 41(4):701–710, 2004.

[27] Abdelhamid Mammeri and Azzedine Boukerche. Computer vision for animal-vehicle collisions mitigation: Challenges and issues. *submitted to IEEE Intelligent Transportation Systems (ITS) magazine*, 2014.

[28] Abdelhamid Mammeri, Depu Zhou, Azzedine Boukerche, and Mohammed Almulla. An efficient animal detection system for smart cars using cascaded classifiers. IEEE International Conference on Communications (ICC), Sydney, 2014.

[29] Aleksandar Milenkovic, Mladen Milosevic, and Emil Jovanov. Smartphones for smart wheelchairs. In *2013 IEEE International Conference on Body Sensor Networks (BSN)*, pages 1–6. IEEE, 2013.

[30] Mark Nixon and Alberto Aguado. *Feature Extraction and Image Processing for Computer Vision.* Academic Press, 2012.

[31] Ronald Nowak and John Paradiso. *Walker's Mammals of the World, pages 1081-1091*, volume 1. Cambridge Univ Press, 1999.

[32] Timo Ojala, Matti Pietikainen, and David Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions.

In *Pattern Recognition, Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, pages 582–585. IEEE, 1994.

[33] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996.

[34] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987, 2002.

[35] Gary Overett and Lars Petersson. Large scale sign detection using hog feature variants. In *Intelligent Vehicles Symposium (IV)*, pages 326–331. IEEE, 2011.

[36] Constantine P Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *Computer Vision, Sixth International Conference on*. IEEE, 1998.

[37] Sri-Kaushik Pavani, David Delgado, and Alejandro F Frangi. Haar-like features with optimally weighted rectangles for rapid object detection. *Pattern Recognition*, 43(1):160–172, 2010.

[38] Chen Peijiang. Moving object detection based on background extraction. In *Computer Network and Multimedia Technology, CNMT International Symposium on*, pages 1–4. IEEE, 2009.

[39] Deva Ramanan, David A Forsyth, and Kobus Barnard. Building models of animals from video. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(8):1319–1334, 2006.

[40] RA Ramlee, MA Othman, MH Leong, MM Ismail, and SSS Ranjit. Smart home system using android application. In *Information and Communication Technology (ICoICT), 2013 International Conference of*, pages 277–280. IEEE, 2013.

[41] Dan Roth, Ming-Hsuan Yang, and Narendra Ahuja. A snow-based face detector. *Urbana*, 51:61801, 2000.

[42] Henry Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1):23–38, 1998.

[43] Andrzej Ruta, Yongmin Li, and Xiaohui Liu. Real-time traffic sign recognition from video by class-specific discriminative features. *Pattern Recognition*, 43(1):416–430, 2010.

[44] Andreas Seiler. Predicting locations of moose–vehicle collisions in sweden. *Journal of Applied Ecology*, 42(2):371–382, 2005.

[45] Mohammad Ashkan Sharafsaleh, Marcel Huijser, Christopher Nowakowski, Mark C Greenwood, Larry Hayden, Jonathan Felder, and May Wang. Evaluation of an animal warning system effectiveness phase two-final report. 2012.

[46] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1453–1460. IEEE, 2011.

[47] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

[48] Frédéric Suard, Alain Rakotomamonjy, Abdelaziz Bensrhair, and Alberto Broggi. Pedestrian detection using infrared images and histograms of oriented gradients. In *Intelligent Vehicles Symposium, IEEE*, pages 206–212. IEEE, 2006.

[49] Kah-Kay Sung and Tomaso Poggio. Example-based learning for view-based human face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1):39–51, 1998.

[50] Louis-Paul TARDIF. *Collisions involving moter vehicles and large animals in Canada*. Transport Canada, 2003.

[51] Vladimir Vapnik. *The nature of statistical learning theory 2 edition - Statistics for engineering and information science*. springer ISBN 0-387-98780-0, 2000.

[52] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, CVPR. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[53] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 4, 2001.

[54] Thomas Williams. *Thermal Imaging Cameras: characteristics and performance*. International Standard Book Number-13: 978-1-4200-7185-6, CRC Press Talor and Francis Group, 2009.

[55] Matthias Zeppelzauer. Automated detection of elephants in wildlife video. *EURASIP Journal on Image and Video Processing*, 2013(1):1–23, 2013.

[56] Lun Zhang, Rufeng Chu, Shiming Xiang, Shengcai Liao, and Stan Z Li. Face detection based on multi-block lbp representation. In *Advances in Biometrics*, pages 11–18. Springer, 2007.

[57] Weiwei Zhang, Jian Sun, and Xiaoou Tang. From tiger to panda: Animal head detection. *Image Processing, IEEE Transactions on*, 20(6):1696–1708, 2011.

[58] Debao Zhou. Infrared thermal camera-based real-time identification and tracking of large animals to prevent animal-vehicle collisions (avcs) on roadways. *Intelligent Transportation Systems Institute, Center for Transportation Studies, University of Minnesota*, 2012.

[59] Mian Zhou and Hong Wei. Face verification using gaborwavelets and adaboost. In *Pattern Recognition,ICPR 18th International Conference on*, volume 1, pages 404–407. IEEE, 2006.