MINISTRY OF EDUCATION AND SCIENCE OF REPUBLIC OF KAZAKHSTAN
SULEYMAN DEMIREL UNIVERSITY
ENGINEERING FACULTY

Department of Computer Engineering

0004.885.5

Dosmukhamed Zhanibekov

Traffic Sign Detection through Image Processing and Pattern Recognition

6M070400– «Computing systems and software» speciality

Kaskelen, 2013

MINISTRY OF EDUCATION AND SCIENCE OF REPUBLIC OF KAZAKHSTAN
SULEYMAN DEMIREL UNIVERSITY
ENGINEERING FACULTY

SULEYMAN DEMIREL UNIVERSITY

Department of Computer Engineering

«Accepted for defense»
responsible director of department
d.t.s professor Amirgaliyev E. N.
«10» 06 2013 year

Director of department
for postgraduate education
Ph.s.c. Aydogdu S.
«11» 06 2013 year

Master dissertation

Traffic Sign Detection through Image Processing and Pattern Recognition

6M070400– «Computing systems and software» speciality

Master student _____ Zhanibekov Dosmukhamed
(signature) (name-surname)

Supervisor _____ d.t.s professor, Amirgaliyev E. N.
(signature) (position, name-surname)

Kaskelen, 2013

ABSTRACT

Urbanisation, growth of cities and their population bring serious changes into our lives. That includes increasing numbers of cars on the road and traffic complexity. Since the very early stages of car and traffic development the very first concern of the designers and engineers was safety on the road. Most commonly it depends on the drivers and pedestrians directly. Attentiveness or inattentiveness of the traffic participants is one of the reasons why accidents can happen.

Traffic signs provide important information for drivers about road condition and hazards. Their discriminating shape and colors make them easily recognizable by humans. Same factors can help development of a visionbased TSR system. Beside the application of TSR in autonomous vehicles, it can also serve as an assistant driver (e.g. when combined with speedometer output) to notify the driver about approaching a traffic sign (e.g. even before driver sees it) or his risky behavior (like driving above the speed limit).

Safe driving was one of the three identified main work areas and meant to employ autonomous vehicle control for safer driving with less mental load on the driver.

My proposed method is composed of three main stages:

1. detection, which is performed using a novel application of maximally stable extremal regions
2. recognition, which is performed with LBP features
3. mobility, which is performed using mobile phone

РЕЗЮМЕ

Урбанизация, рост городов и населения принести серьезные изменения в нашу жизнь. Это включает в себя все большее число автомобилей на дорогах и сложности движения на них. С самых ранних стадиях движения автомобиля и развития самой первой заботой дизайнеров и инженеров была безопасность на дороге. Чаще всего это зависит от водителей и пешеходов напрямую. Внимательность или невнимательности участников дорожного движения является одной из причин, почему несчастные случаи могут произойти.

Дорожные знаки дают важную информацию для водителей о состоянии дорог и опасностей. Их форма и дискриминационные цвета делают их легко узнаваемыми людьми. Такие же факторы могут помочь формированию концепции системы TSR. Кроме применения TSR в автономных транспортных средств, оно также может служить в качестве помощника, чтобы уведомить водителя о приближении дорожный знака (например, еще до того, водитель увидит его).

Мной предлагаемый способ состоит из трех основных этапа:

1. обнаружение дорожного знака

2. распознавание, которое осуществляется с LBP

3. мобильности, которая выполняется с помощью мобильного телефона

ТҮЙІН

Урбанизация, қаланың және оның халқының өсуі біздің өмірімізге өзгерістерді әкелді. Ол жолдағы автомобильдің көптігін және қозғалысының күрделігің әкелді. Инжинерлер мен дизайнерлергін ертеден келе жатқан уайымы, ол жолдағы кауіпсіздік. Көбіне бұл жүргізушіден мен жүргіншілерге тәуелді болады.

Жол белгілері, жолдың күйі туралы маңызды ақпаратты жүргізушілерге береді. Сол себептен мен, жүргізушілер мен жаяу жүргіншілерге арналған бағдарламаны жасауды ұйғардым. Бұл бағдарламаның үш негізгі бөліктен тұрады:

1. жол бегісін табу
2. кай жол бегісі екенін LBP арқылы тану
3. және осының бәрін мобильді қылу

CONTENT

DEFINITIONS, ACRONYMS, ABBREVIATIONS

TSR	Traffic sign recognition
LBP	Local binary patterns.
IDE	An integrated development environment or interactive development environment is a software application that provides comprehensive facilities to computer programmers for software development.
eclipse	In computer programming, Eclipse is a multi-language Integrated development environment comprising a base workspace and an extensible plug-in system for customizing the environment. It is written mostly in Java.
openCV	OpenCV is a library of programming functions mainly aimed at real-time computer vision.
android	Android is a Linux-based operating system designed primarily for touchscreen mobile devices such as smartphones and tablet computers.
SDK	A software development kit is typically a set of software development tools that allows for the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar development platform.
ADB	The Android Debug Bridge is a toolkit included in the Android SDK package. It consists of both client and server-side programs that communicate with one another. The ADB is typically accessed through the command-line interface.

INTRODUCTION

TSR can be considered part of the bigger problem of autonomous vehicles. An autonomous vehicle system relies on vision-based recognition of surrounding area in order to make driving decisions. This vision-based recognition system functions as the feedback provider for control of steering wheel, accelerator, brake, ..., and needs to:

- Recognize road and lane to allow control system follow the course of own vehicle.
- Detect obstacles on the road till control system avoid them.
- Detect the passing vehicles (e.g. by side or back cameras) to notify the control system about probable hazards.
- Detect and interpret the traffic signs to provide feedback for safe driving.

2. BACKGROUD

Fast-paced developments in the car industry require car makers to introduce newer, better, faster and cheaper features for their products. Among those features and functionalities highest priority is deservedly given to safety of the car including both passive and active safety options. Most of the time, safety of a driver and his passengers directly depends on information the driver gets about the traffic. One of the essential roles of traffic management is given to traffic lights that allow us to stay in control of whatever happens on the road. To make drivers more careful of the traffic lights it would be good to signal if there is a traffic light ahead. Since, existing parking sensors are not able to sense signs and traffic lights, and GPS navigation does not provide information about traffic lights on the road, these features cannot be used for traffic light detection.

Video monitoring however might provide enough information to processing to see if there are traffic lights ahead. Even though there is no such system or device available on the market currently, some of the auto giants have already looked into this idea. For example, Opel has been playing with recognition of traffic signs on the road and planning to install it on the Opel Insignia in the future, which is on serial production. Mercedes is also working on such system but did not release details of their achievements yet.

2.1.TODAY TRAFFIC SITUATION

In Kazakhstan today, nobody can doubt the fact that video surveillance systems have brought about a positive change in the general security situation. International experience, and unfortunately also experience in Kazakhstan, testifies to the fact that without properly designed systems and professionally trained personnel, it is impossible to ensure law and order, which lies at the heart of all urban security.

The development of Kazakhstan's municipal video surveillance systems is mainly in this area, as well as monitoring of road users – obtaining footage of traffic violations is also improving the situation in this area, helping to protect society from corruption and favouritism. On the whole, Kazakhstan's municipal video surveillance

systems are developed within the country's 'Safe City' programme and have their own characteristics. In order to understand them better, their development path needs to be traced right back to the very beginning, and all their influencing factors need to be assessed.

Figure 1 - video surveillance systems

By 2011-2012, spending on video surveillance in Kazakhstan's regional centres had increased tenfold from 2008. A new OCC and 600 cameras were provided in Atyrau and the number of surveillance cameras on the streets of Pavlodar, Atkau and Semipalatinsk increased. Even in relatively small Kostanay (300,000 inhabitants) over $600,000 was spent, with dozens of video cameras connected to an OCC and Department of Internal Affairs centres as part of the 'Safe Streets' programme.

There were several reasons for this unexpected growth, among them issues of increasing vehicle numbers and attempts to change the situation on the roads, as well as a general desire to enhance the level of road safety. But the most important factor of all was a new one – the threat of terrorism. In 2011 there were several high-profile successful and failed acts of terrorism, and global experience shows that without modern video surveillance, countering this threat is not possible.

We can agree that is difficult to picture what happened in Kazakhstan in 2011 taking place in Europe – a radical Islamist stole a car, drove to a gun shop in order to rob it, killed two police officers in the subsequent chase, and then went on to the Office of National Security. Having arrived, he opened fire on the building with a grenade launcher and only after this blew himself up (and not just himself). Clearly,

if the police had been able to see this situation evolving, they would have been able to quickly find a way of stopping or liquidating the terrorist. However, at the intersection of Taraz, in South Kazakhstan, they could not see him, and he met no organised resistance. For this reason, it is most likely that the reality and high probability of a new terrorism threat led to cameras being quickly installed on streets and in schools, hospitals, playschools and other public buildings.

Projects were financed from city budgets, although there were attempts to do things a different way – in Petropavlovsk in 2009, "taking into account the local budget deficit", the Department of Internal Affairs asked the authorities of the North Kazakhstan Region and the city of Petropavlovsk to "assist in the acquisition of external surveillance cameras and the connection of them to OCCs for newly constructed buildings, large markets, supermarkets, and other facilities at the expense of construction companies and owners of these facilities." Needless to say, after the first terrorist attacks in the south, the government money was quickly found.

The main conclusion is the existence of structure of municipal video surveillance systems. Its function in Kazakhstan has expanded to include not just accompaniment of official figures and recording of traffic violations, but also monitoring in schools, hospitals, and public places.

At the same time, urban video surveillance systems in other agencies – the fire and health services, the Ministry of Emergency Situations, etc – has not developed. There have also been no attempts to integrate situation rooms, which would ensure the effective use of security cameras in a range of crisis situations.

2.2. RECORD STATUS OF ACCIDENTS

According to information from the data bank records of accidents and people in them have suffered for four months in 2013 for the Republic reported 4,854 accidents, 3,414 against accidents recorded in the same period of 2012.

Thus, in the past period registered an increase in the number of road accidents in the 1440 facts. When viewed in the context of each region, the highest number of

traffic accidents committed in the territory of Almaty - 1262 (633), South Kazakhstan - 754 (495), Almaty - 423 (394), Zhambyl - 401 (222) East Kazakhstan - 346 (230), Karaganda - 228 (168) and Aktobe - 211 (132) regions.

Most of the accidents occurred in the current period as a result of accidents with pedestrians - 2323, due to the collision of vehicles - 1710, due to rollover - 334, hit a snag - 225, hitting the cyclists - 33, hit a stationary vehicles - 59 , impact on animals - 13, hit a horse-drawn carts - 3 and other types of accidents - 154.

The reasons for non-compliance has been committed by an accident of traffic, namely speeding - 1127, pedestrians crossing the roadway at an undisclosed location - 384, exit at the oncoming traffic - 488, when driving through crosswalks - 493, disobeying signals of regulation and requirements of road signs - 151, improper passing - 30 left for other violations - 2181. The largest number of accidents committed in settlements 3829, in which 277 people were killed. On the roads of the country's registered - 1025 crash, which killed 399 people. In general, the situation in the Republic as compared to the same period last year is as follows. Died as a result of an accident 676 persons (590) and wounded - 6173 persons (4239)

At the same time, a large number of persons killed in road accidents registered in Almaty - 113 (116), South Kazakhstan - 109 (98), Zhambyl - 76 (49), Almaty - 45 (51), East-Kazakhstan - 41 (40), Karaganda - 41 (45), Akmola - 41 (19) and Mangistau - 34 (23).[2]

2.3.MOBILE MARKET GROTH

As mobile phone users around the world upgrade to smartphones, they are overwhelmingly choosing Android. Google's operating system was installed on more than one-third of all mobile phones that shipped last quarter, and on more than two-thirds of all smartphones.

Research firm Canalys estimates that 438 million mobile phones shipped during the fourth quarter of 2012, roughly the same number as in the year-ago quarter. But a much larger portion of those were smartphones – 216.5 million or just

under 50% of the total. Canalsys says Q4 smartphone shipments were up 37% from the year-ago quarter.

The numbers from Canalys are very close to those released last month by IDC, which estimated total Q4 smartphone shipments at 219.4 million. Both firms ranked Samsung as the market leader with roughly 63 million smartphones shipped. (Samsung does not publicly report shipment numbers for its mobile devices.)

Apple is of course the No. 2 vendor in the worldwide smartphone market; the company said last month that it shipped 47.8 million iPhones last quarter. In the United States, Apple remains the leading smartphone vendor with more than one-third of the market by some estimates.

Canalys and IDC both rank China's Huawei as the world's No. 3 smartphone vendor, with about 11 million units' shipped during the fourth quarter. The two research firms differ on who ranks No. 4 and No. 5. Canalys thinks that ZTE shipped 10.1 million phones and Lenovo shipped 9.5 million; IDC thinks Sony shipped 9.8 million and ZTE 9.5 million.

All the top five smartphone vendors ranked by both firms make primarily Android phones. Mobile operators are hoping that a third operating system will emerge as a serious competitor to Android and Apple's iOS, but right now BlackBerry and Microsoft's Windows combine for just 10% market share.

"BlackBerry, Microsoft and Nokia, as well as other Android vendors, have strategies and devices in place to attack, but the task is daunting to say the least," said Pete Cunningham, principal analyst at Canalys. "Vendors left in the wake of the top vendors must at the very least improve their portfolios, time-to-market and marketing, as well as communicate their differentiators. Microsoft, BlackBerry and other new OS entrants, such as Mozilla, must make the OS switch as simple as possible and drive and localize their respective app and content ecosystems."

2.4.LEAD MY WAY

This is name of my android application which I gave. It runs on android with 2.2 version and upper. Uses openCV library. For classsifieing uses LBP feature extraction. It was develeppen on eclipse juno 4.2 IDE.

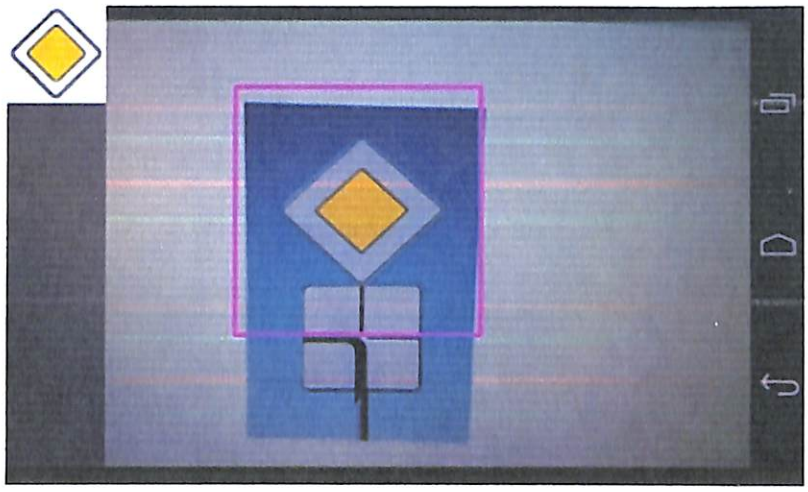

Figure 2 – Lead my way screenshot

Figure 3 – QR code to application in play market

3. LITERATURE REVIEW

This chapter discusses research work and materials that were referenced during the design and implementation of theproject. It includes materials on imageprocessing techniques used, researchon featureextraction and additional researchon traffic light systems.

3.1.IMAGEPROCESSING

ImageProcessing is a technique toenhanceraw images received from cameras/sensors placed on satellites, spaceprobes and aircrafts or pictures taken in normal day-today life for various applications.

Various techniques have been developed inImageProcessing during the last four to five decades. Most of the techniques are developed for enhancing images obtainedfrom unmanned spacecrafts, spaceprobes and military reconnaissance flights. Imageprocessing systems are becoming popular due toeasyavailabilityof powerful personnel computers, large size memory devices, graphics softwares etc Image processing usually refers to digital imageprocessing, but optical and analog imageprocessing alsoarepossible. This article is about general techniques that apply toall of them. Theacquisition of images (producing the input image in the first place) is referred toas imaging.[10]

Imageprocessing is referred toprocessing of a 2D picture byacomputer. Basic definitions:

An image is defined in the "real world" is considered to bea function of two real variables, for example, a (x, y) withaas theamplitude (e.g. brightness) of the imageat the real coordinateposition (x, y).

Modern digital technologyhas made it possible to manipulate multi-dimensional signals with systems that range from simple digital circuits toadvanced parallel computers. The goal of this manipulation can be divided into threecategories:

- ImageProcessing (image in -> imageout)
- ImageAnalysis (image in -> measurements out)

- Image Understanding (image in ->high-level description out)

An image may beconsidered tocontain sub-images sometimes referred toas regions-of-interest, ROIs, or simply regions. This concept reflects the fact that images frequentlycontain collections of objects eachof whichcan be the basis for a region. In a sophisticated imageprocessing system it should bepossible toapply specific imageprocessing operations to selected regions. Thus onepart of an image (region) might beprocessed to suppress motion blur whileanother part might beprocessed to improvecolor rendition. Sequenceof imageProcessing is a technique toenhance raw images received from cameras/sensors placed on satellites, spaceprobes and aircrafts or pictures taken in normal day-to-day life for various applications.

Various techniques have been developed in ImageProcessing during the last fourto five decades. Most of the techniques are developed for enhancing images obtained from unmanned spacecrafts, spaceprobes and military reconnaissance flights. ImageProcessing systems are becoming popular due toeasyavailabilityof powerful personnel computers, large size memory devices, graphics softwares etc.

ImageProcessing is used in various applications suchas:

- Remote Sensing
- Medical Imaging
- Non-destructiveEvaluation
- Forensic Studies
- Textiles
- Material Science.
- Military
- Film industry
- Document processing
- Graphicarts
- Printing Industry

The common steps in image processing are image scanning, storing, enhancing and interpretation. The schematic diagram of image scanner-digitizer diagram is shown in figure 1.

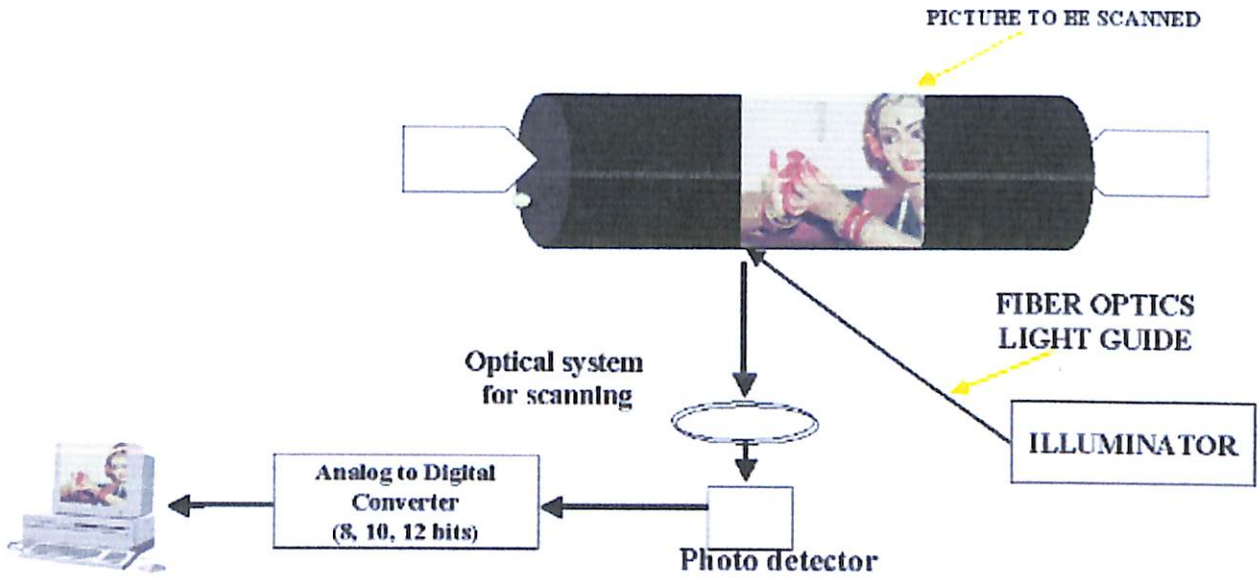

Figure 4 - The schematic diagram of image scanner-digitizer diagram

Methods of Image Processing

There are two methods available in Image Processing.

Analog Image Processing

Analog Image Processing refers to the alteration of image through electrical means. The most common example is the television image.

The television signal is a voltage level which varies in amplitude to represent brightness through the image. By electrically varying the signal, the displayed image appearance is altered. The brightness and contrast controls on a TV set serve to adjust the amplitude and reference of the video signal, resulting in the brightening, darkening and alteration of the brightness range of the displayed image.

Digital Image Processing

In this case, digital computers are used to process the image. The image will be converted to digital form using a scanner – digitizer and then process it. It is defined as the subjecting numerical representations of objects to a series of operations in

in order toobtain a desired result. It starts withone imageand produces a modified version of the same. It is thereforeaprocess that takes an image intoanother.

The term *digital imageprocessing* generally refers toprocessing of a two-dimensional picture bya digital computer. In a broader context, it implies digital processing of any two-dimensional data. A digital image is an arrayof real numbers represented bya finite number of bits.

Theprincipleadvantageof Digital ImageProcessing methods is its versatility, repeatabilityand thepreservation of original dataprecision.

The various ImageProcessing techniques are:

- Image representation
- Imagepreprocessing
- Imageenhancement
- Image restoration
- Imageanalysis
- Image reconstruction
- Image datacompression

3.2.PATTERN RECOGNITION AND ITS CONCEPTS

In machine learning, pattern recognition is theassignment of a label toa given input value. An exampleof pattern recognition is classification, whichattempts toassign each input value tooneof a given set of classes (for example, determine whether a given email is "spam" or "non-spam"). However, pattern recognition is a more general problem that encompasses other types of output as well. Other examples are regression, whichassigns a real-valued output toeach input; sequence labeling, whichassigns aclass toeach member of a sequenceof values (for example, part of speech tagging, whichassigns apart of speech toeach word in an input sentence); and parsing, whichassigns aparse tree toan input sentence, describing the syntactic structureof the sentence.

Pattern recognition algorithms generallyaim toprovidea reasonableanswer for all possible inputs and toperform "most likely" matching of the inputs, taking intoaccount their statistical variation. This is opposed topattern matching algorithms, which look for exact matches in the input withpre-existing patterns. Acommon exampleof apattern-matching algorithm is regular expression matching, which looks for patterns of a given sort in textual dataand is included in the searchcapabilities of many text editors and word processors. In contrast topattern recognition, pattern matching is generally not considered a typeof machine learning, althoughpattern-matching algorithms (especially with fairly general, carefully tailored patterns) can sometimes succeed in providing similar-qualityoutput to the sort provided bypattern-recognition algorithms.

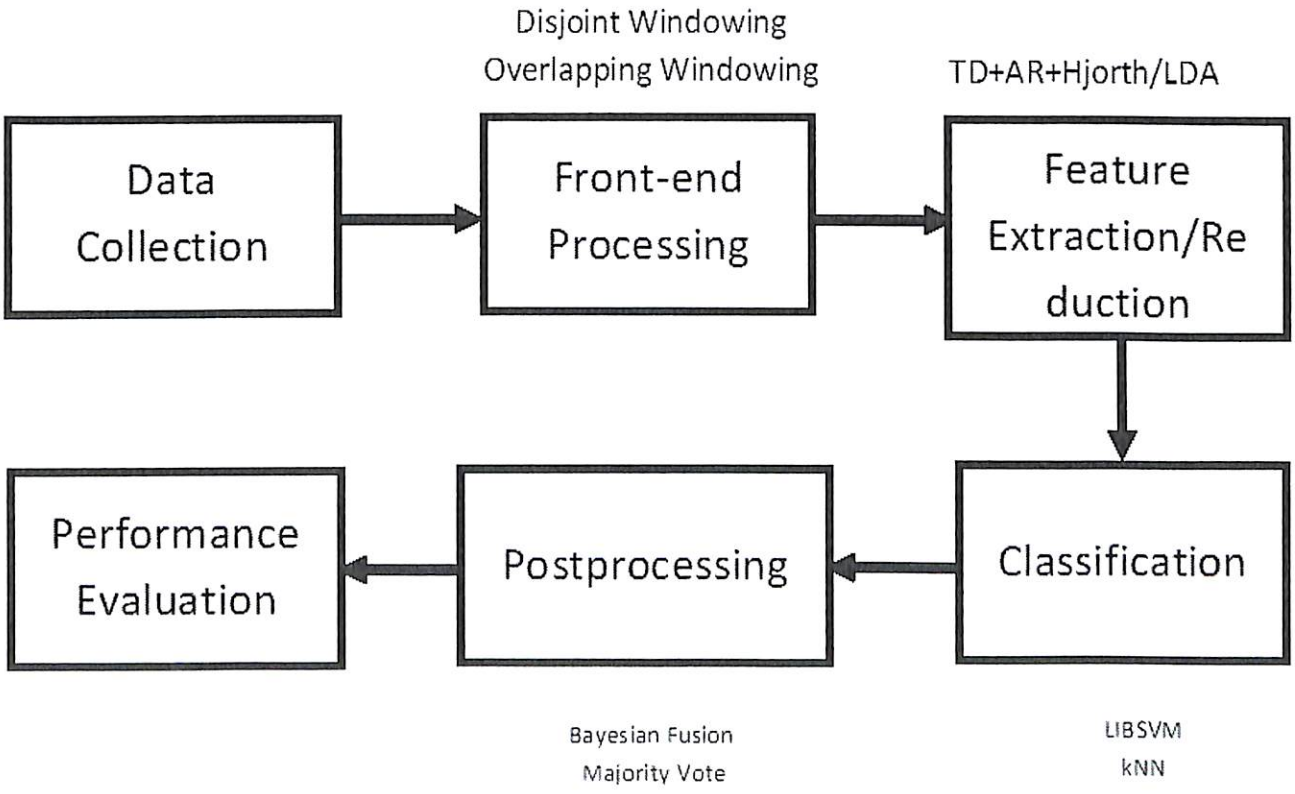

Figure 5 – Pattern recognition concepts

In manypractical problems, there is a need to make some decision about thecontent of an imageor about theclassication of an object that it contains. For example, the user of a notebook computer may beable to give input using handprinted characters. In this case, there would be m = 128 ASCIIcharacters and

eachhandprinted object would beclassied intooneof the m classes. Theclassication of an object {whether it is an 'A' or an '8', etc{would be based on the features of its optical imageor perhaps of apressure footprint, which is alsoan image-like representation. Theclassication process might actually fail, either because thecharacter is badly made, or because theperson invented a new character. Usually, a reject class is included in a system design in order tocover suchcases. Image dataput into the reject class might beexamined again later at somehigher level, might result in the formation of a new class, or might just be saved in raw form for viewing.

One definition for recognition is to know again. A recognition system must contain some memoryof theobjects that it is to recognize. This memory representation might be built in- perhaps as is the frog's model of afiy; or might be taught byprovision of a large number of samples- as a schoolteacher teaches thealphabet; or it might beprogrammed in terms of specific image features- perhaps as a mother would teachachild to recognizefire trucks versus buses. Recognition and learning of patterns are subjects of considerable depthand interest tocognitivepyschology- pattern recognition- and computer vision. This chapter takes apractical approachand describes methods that havehad success in applications- leaving somepointers to the large theoretical literature in the references at theend of thechapter.

3.3.COMMON MODEL FOR CLASSICATION

Summarize theelements of thecommon model of classification; this breakdown is practical rather than theoretical and done so that pattern recognition systems can be designed and built using separately developed hardwareand software modules.

Classes

There is a set of m known classes of objects. Theseare known either by some description or byhaving a set of examples for eachof theclasses. For example- for character classification- wehaveeither a description of theappearanceof eachcharacter

or wehavea set of samples of each. In the general case- there will bea special reject class for objects that cannot beplaced in oneof the known classes.

An ideal class is a set of objects having some important common properties: in practice, aclass to whichan object belongs is denoted by someclass label. Classification is aprocess that assigns a label toan object according to some representation of theobject's properties. Aclassifier is a deviceor algorithm which inputs an object representa-tion and outputs aclass label.

A reject class is a genericclass for objects that cannot beplaced in anyof the designated known classes.

Sensor/transducer

There must be some device to sense theactual physical object and output a (usually digital* representation of it for processing by machine. Most often- the sensor is selected from existing sensors (theshelf built for a larger class of problems. For example- toclassify vegetables in the supermarket- wecould first try using a general color camera that would providean image representation from whichcolor- shape- and texture features could beobtained. To recognizecharacters made byan impression using a stylus- we would useapressure sensitivearray.

Since this is a book about machine vision- sensors that produce 3D arrays of sensed dataareof most interest. However- pattern recognition itself is more general and just as applicable to recognizing spoken phone numbers- for example- as phone numbers written on paper.

Featureextractor

The featureextractor extracts information relevant toclassification from the data input by the sensor. Usually- featureextraction is done in software. Softwarecan beadapted to the sensor hardwareon the input sideand can evolve through researchand development tooutput results highly relevant toclassification. Many image features were defined in theprevious chapter.

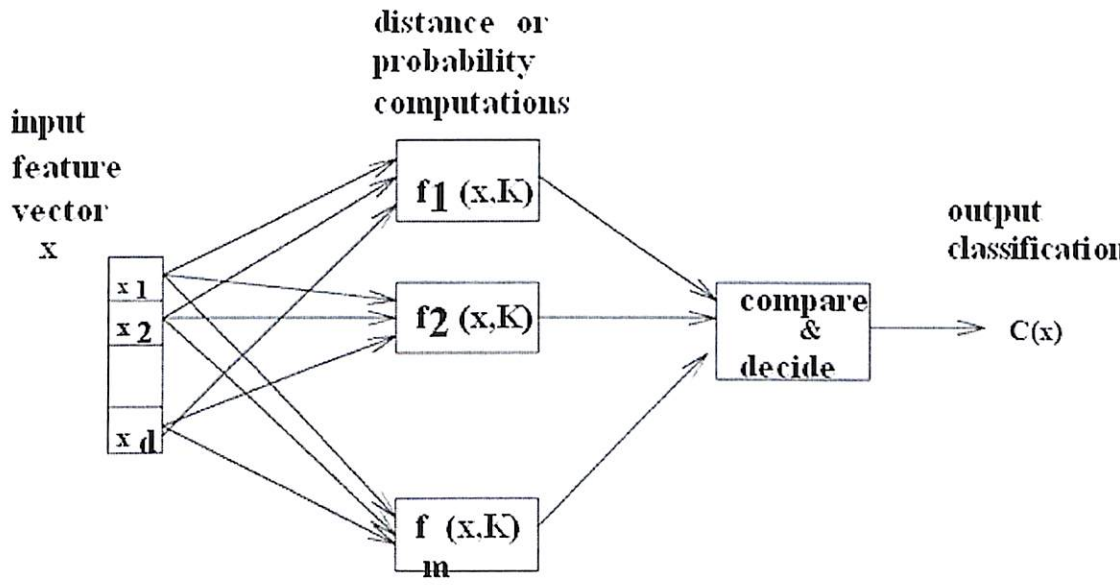

Figure 6 – Featureextrator

Classifier

Classification system diagram; discriminant functions f (x; K* perform somecomputation on input feature vector x using some knowledgeK from training and pass results toafinal stage that determines theclass.

Theclassifier uses the features extracted from the sensed object data toassign theobject tooneof the m designated classes C^1- C^2- ... - Cm ^{1}- Cm = Cr - whereCr denotes the reject class.

A d-dimensional feature vector x is the input representing theobject to beclassified. The system has one block for eachpossibleclass- whichcontains some knowledgeKabout theclass and someprocessing capability. Results from the m computations arepassed to thefinal classification stage- which decides theclass of theobject. The diagram is general enough to model the three diPtserent types of classification discussed below; (a* classification using the nearest mean- (b* classification by maximum aposteriori probability- and (c* classification using a feed.forward artificial neural network.

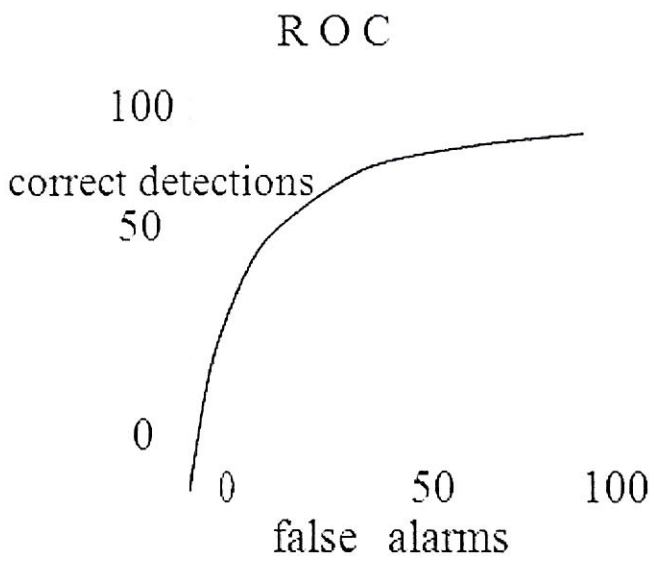

Figure 7 – ROC

The receiver operating curveor \ROC" plots correct detection rate versus falsealarm rate. Generally- the number of falsealarms will go upas the system attempts to detect higher percentages of known objects. Modest detection performancecan often beachieved at a low cost in falsealarms- but correct detection of nearlyall objects will causea largepercentageof unknown objects to be incorrectlyclassified into some known class.

Classification is usually somewhat arbitraryand more due toengineering concerns than to some inherent properties of an application. Indeed- we will look at neural nets- whichcan doone stepclassification directly from the input image.

Evaluation of system error

Theerror rateof aclassification system is one measureof how well the system solves theproblem for which it was designed. Other measures are speed- in terms of how manyobjects can beprocessed per unit time- and expense- in terms of hardware- software- and development cost. Performance is determined by both theerrors and rejections made; clas. sifying all inputs into the reject class means that the system makes noerrors but is useless.

Classification error – Theclassifier makes aclassification error when- ever it

lassifes the input object as class Ciwhen the trueclass is class Cj; i= j andCi= Cr, the reject class.

Empirical error rate – Theempirical error rateof aclassification sys-tem is the number of errors madeon independent test datadivided by the number of classifications attempted.

Empirical reject rate – Theempirical reject rateof aclassification sys-tem is the number of rejects madeon independent test data divided by the number of classifications attempted.

Independent Test Data – Independent test dataare sampleobjects with trueclass known, including objects from the \reject class", that were not used in designingthe featureextraction and classification algorithms.

Theabove definitions can be used in practice to test theperformanceof aclassification system. We must be verycareful to insure that the samples used to design and those used to test the system are representativeof the samples that the system will have toprocess in the future; and- samples used to test the system must be independent of those used to design it. Sometimes- weassume that our data follows some theoretical distribution. With this assumption- weareable tocomputea theoretical probabilityof error for futureperformance- rather than just an empirical error rate from testing. This concept will be discussed below.

Supposeahandprinted character recognition module for ahand.held computer correctly recognizesof a user's input characters. Given that the user mayhave toedit an input document anyway- a 6&error rate may beacceptable. Interestingly- sucha system might actually train the user- as well as the user training the system- so that performance gradually improves. For example- perhaps the user learns to morecarefullyclose up 9's so that theyare not confused with 7's. For a banking system that must read hand.printed digits on deposit slips- a 6&error rate might be intolerable.

Falsealarmsandfalsedismissals

Someproblemsarespecialtwo-classproblemswherethemeaningoftheclassesmightbe (a* goodobjectversusbadobject-

(b* objectpresentintheimageversusobjectabsent-or (c* personhasdiseaseDversuspersondoesnothavediseaseD. Here-theerrorstakeonspecialmeaningandarenotsymmetric. Case (c* ismostinstructive; ifthesystemincor.

rectlysaysthatthepersondoeshavediseaseDthentheerroriscalledafalsealarmorfalsepositive; whereas- ifthesystemincorrectlysaysthatthepersondoesnothavediseaseD-thentheerroriscalledafalsedismissalorfalsenegative. Inthecaseofthefalsealarm-itprobablymeansthatthepersonwillundergothecostofmoretests-oroftakingmedicinethatisnotneeded. Inthecaseoffalsedismissal-thediagnosisismissedandthepersonwillnotbetreated-possiblyleadingtogravecircumstances.

BecausethecostoftheerrorsdiPtsergreatly-itmaymakesensetobiasthedecisioninordertominimizefalsedismissalsatthecostofincreasingthenumberoffalsealarms. Case (a* islessdramaticwhentheproblemistoculloutbruisedcherries; afalsealarmmaymeanthatthecherrygoesintoapieratherthanintheproducebinwhereitwouldhavehadmorevalue. Falsealarmsincase (b* maymeanwewasteenergybyturningonalightwhenthereareallywasnomotioninthesceneorthatwecountedanautoonthehighwaywhenonereallydidnotpassby; falsedismissalsincase (b* alsohaveinterestingconsequences. Typicalreceiveroperatingcurve-whichrelatesfalsealarmratetodetectionrate.

Inordertoincreasethepercentageofobjectscorrectlyrecognized-oneusuallyhastopayacostofincorrectlypassingalongobjectsthatshouldberejected.

3.4.FEATUREEXTRACTION

In pattern recognition and in imageprocessing, featureextraction is a special form of dimensionality reduction.

When the input data toan algorithm is too large to beprocessed and it is suspected to be notoriously redundant (e.g. the same measurement in both feet and meters) then the input data will be transformed intoa reduced representation set of

features (also named features vector). Transforming the input data into the set of features is called featureextraction. If the features extracted arecarefullychosen it is expected that the features set will extract the relevant information from the input datain order toperform the desired task using this reduced representation instead of the full size input. [11]

Featureextraction involves simplifying theamount of resources required to describea large set of dataaccurately. When performing analysis of complex dataoneof the major problems stems from the number of variables involved. Analysis witha large number of variables generally requires a largeamount of memoryand computation power or aclassification algorithm whichoverfits the training sampleand generalizes poorly to new samples. Featureextraction is a general term for methods of constructing combinations of the variables to get around theseproblems while still describing the data with sufficient accuracy.

Best results areachieved when an expert constructs a set of application-dependent features. Nevertheless, if no suchexpert knowledge is available general dimensionality reduction techniques mayhelp. These include:

- Principal component analysis
- Semidefiniteembedding
- Multifactor dimensionality reduction
- Multilinear subspace learning
- Nonlinear dimensionality reduction
- Isomap
- Kernel PCA
- Multilinear PCA
- Latent semanticanalysis
- Partial least squares
- Independent component analysis
- Autoencoder

3.5.HAAR-like features

Haar-like features are digital image features used in object recognition. Theyowe their name to their intuitive similarity withHaar wavelets and were used in the first real-time face detector.

Historically, working withonly image intensities (i.e., the RGBpixel values at eachand everypixel of image) made the task of featurecalculation computationallyexpensive. Apublication byPapageorgiou et discussed working withan alternate feature set based on Haar wavelets instead of the usual image intensities. Violaand Jonesadapted the ideaof using Haar wavelets and developed the so-called Haar-like features. AHaar-like featureconsiders adjacent rectangular regions at a specific location in a detection window, sums up thepixel intensities in each region and calculates the difference between these sums. This difference is then used tocategorize subsections of an image. For example, let us say wehavean image database withhuman faces. It is acommon observation that among all faces the region of theeyes is darker than the region of thecheeks. Thereforeacommon haar feature for face detection is a set of twoadjacent rectangles that lieabove theeyeand thecheek region. Theposition of these rectangles is defined relative toa detection window that acts likea bounding box to the target object (the face in this case).

In the detection phaseof the Viola–Jones object detection framework, a window of the target size is moved over the input image, and for each subsection of the image theHaar-like feature is calculated. This difference is then compared toa learned threshold that separates non-objects from objects. Because suchaHaar-like feature is onlya weak learner or classifier (its detection quality is slightly better than random guessing) a large number of Haar-like features are necessary to describean object with sufficient accuracy. In the Viola–Jones object detection framework, theHaar-like features are thereforeorganized in something called aclassifier cascade to form a strong learner or classifier.

The keyadvantageof aHaar-like featureover most other features is its calculation speed. Due to the useof integral images, aHaar-like featureof any sizecan

becalculated in constant time (approximately 60 microprocessor instructions for a 2-rectangle feature).

A simple rectangular Haar-like featurecan be defined as the differenceof the sum of pixels of areas inside the rectangle, whichcan beat anyposition and scale within theoriginal image. This modified feature set is called *2-rectangle feature*. Violaand Jones also defined 3-rectangle features and 4-rectangle features. The values indicatecertain characteristics of aparticular areaof the image. Each feature typecan indicate theexistence (or absence) of certain characteristics in the image, suchas edges or changes in texture. For example, a 2-rectangle featurecan indicate where the border lies between a dark region and a light region.

Oneof thecontributions of Violaand Jones was to use summed area tables, which theycalled *integral images*. Integral images can be defined as two-dimensional lookup tables in the form of a matrix with the same sizeof theoriginal image. Eachelement of the integral imagecontains the sum of all pixels located on the up-left region of theoriginal image (in relation to theelement's position). This allows tocompute sum of rectangular areas in the image, at anyposition or scale, using only four lookups:

$$\text{sum} = I(C) + I(A) - I(B) - I(D).$$

wherepoints A, B, C, D belong to the integral image I, as shown in the figure.EachHaar-like feature may need more than four lookups, depending on how it was defined. Violaand Jones's 2-rectangle features need six lookups, 3-rectangle features need eight lookups, and 4-rectangle features need nine lookups.

LienhartandMaydt introducedtheconceptofatilted (45°) Haar-likefeature. Thiswasusedtoincreasethedimensionality ofthesetoffeaturesinanattempttoimprovethed etectionofobjectsinimages. Thiswassuccessful, assomeofthesefeaturesareabletodescribetheobjectinabetterway. For example, a 2-rectangle tilted Haar-like featurecan indicate theexistenceof an edgeat 45°.

Messom and Barczak extended the idea toa generic rotated Haar-like feature. Although the idea sounds mathematically sound, practical problems prevented the useof Haar-like features at anyangle. In order to be fast, detection algorithms use low resolution images, causing rounding errors. For this reason, rotated Haar-like features are not commonly used.

3.6.LOCAL BINARY PATTERNS

Local BinaryPattern (LBP) is a simpleyet veryefficient textureoperator which labels thepixels of an image by thresholding the neighborhood of eachpixel and considers the result as a binary number. Due to its discriminativepower and computational simplicity, LBP textureoperator has becomeapopular approach in various applications. It can be seen as a unifying approach to the traditionally divergent statistical and structural models of textureanalysis. Perhaps the most important propertyof the LBPoperator in real-world applications is its robustness to monotonic gray-scalechanges caused, for example, by illumination variations. Another important property is its computational simplicity, which makes it possible toanalyze images in challenging real-time settings.

The basic idea for developing the LBPoperator was that two-dimensional surface textures can be described by twocomplementary measures: local spatial patterns and gray scalecontrast. Theoriginal LBPoperator (Ojalaet al. 1996) forms labels for the imagepixels by thresholding the 3 x 3 neighborhood of eachpixel with thecenter valueand considering the result as a binary number. Thehistogram of these $2^8 = 256$ different labels can then be used as a texture descriptor. This operator used jointly witha simple local contrast measureprovided very good performance in unsupervised texture segmentation After this, many related approaches have been developed for textureand color texture segmentation.

The LBPoperator was extended to use neighborhoods of different sizes (Ojalaet al. 2002). Using acircular neighborhood and bilinearly interpolating values at non-integer pixel coordinates allow any radius and number of pixels in the neighborhood. The gray scale varianceof the local neighborhood can be used as

thecomplementarycontrast measure. In the following, the notation (P,R) will be used for pixel neighborhoods which means P sampling points on acircleof radius of R. See Fig. 2 for an exampleof LBPcomputation.

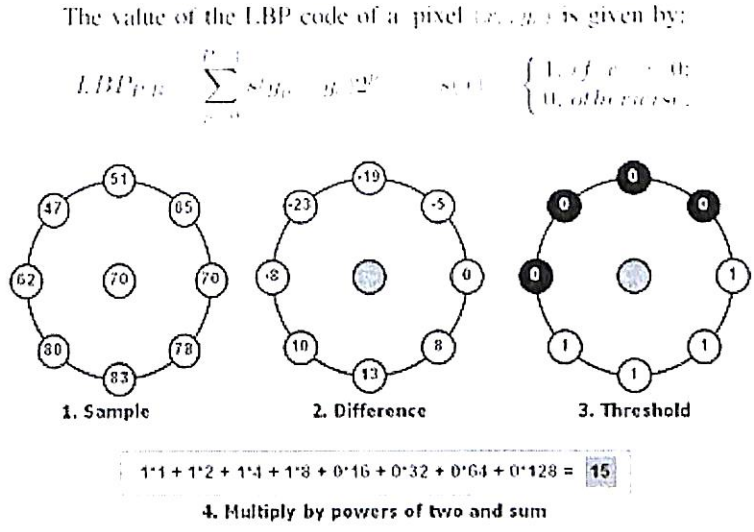

Figure8 –An exampleof LBPcomputation.

Another extension to theoriginal operator is the definition of socalled *uniform patterns*, whichcan be used to reduce the lengthof the feature vector and implement a simple rotation-invariant descriptor. This extension was inspired by the fact that some binarypatterns occur morecommonly in texture images than others.

A local binarypattern is called uniform if the binarypattern contains at most two bitwise transitions from 0 to 1 or vice versa when the bit pattern is traversed circularly. For example, thepatterns 00000000 (0 transitions), 01110000 (2 transitions) and 11001111 (2 transitions) are uniform whereas thepatterns 11001001 (4 transitions) and 01010010 (6 transitions) are not.

In thecomputation of the LBP labels, uniform patterns are used so that there is a separate label for each uniform pattern and all the non-uniform patterns are labeled witha single label. For example, when using (8,R) neighborhood, therearea total of 256 patterns, 58 of whichare uniform, whichyields in 59 different labels.

Ojalaet al. (2002) noticed in their experiments with texture images that uniform patterns account for a little less than 90% of all patterns when using the (8,1)

neighborhood and for around 70% in the (16,2) neighborhood. Each bin (LBPcode) can be regarded as a micro-texton. Local primitives whicharecodified by these bins include different types of curved edges, spots, flat areas etc.

The following notation is used for the LBPoperator: $LBP_{P,R}^{u2}$. The subscript represents using theoperator in a (P,R) neighborhood. Superscript u2 stands for using only uniform patterns and labeling all remaining patterns witha single label. After the LBP labeled image $f_l(x,y)$ has been obtained, the LBPhistogram can be defined as

$$Hi=\sum x,yI\{fl(x,y)=i\},i=0,\ldots,n-1,(1)$$

inwhichnisthenumberofdifferentlabelsproducedbytheLBPoperator, andI{A} is 1 ifAistrueand 0 ifAisfalse.

Whentheimagepatcheswhosehistogramsaretobecomparedhavedifferentsizes, thehistogramsmustbenormalizedtogetacoherentdescription:

$$Ni=Hi\sum n-1j=0Hj.(2)$$

Theoriginal LBPoperator was defined toonly deal with the spatial information. Later, it was extended toa spatiotemporal representation for dynamic textureanalysis. For this purpose, the socalled Volume Local BinaryPattern (VLBP) operator was proposed (Zhaoand Pietikäinen 2007). The idea behind VLBPconsists of looking at dynamic textureas a set of volumes in the (X,Y,T) space whereXand Y denote the spatial coordinates and T denotes the frame index (time). The neighborhood of eachpixel is thus defined in three dimensional space. Then, similarly to LBP in spatial domain, volume textons can be defined and extracted intohistograms. Therefore, VLBPcombines motion and appearance together to describe dynamic textures.

To make VLBPcomputationally simpleand easy toextend, an operator based on co-occurrences of local binarypatterns on threeorthogonal planes (LBP-TOP) was also introduced. LBP-TOPconsiders threeorthogonal planes: XY, XTand YT, and concatenates local binarypattern co-occurrence statistics in these threedirections. Thecircular neighborhoods are generalized toelliptical sampling to fit to the space-time statistics.

directions. The circular neighborhoods are generalized to elliptical sampling to fit to the space-time statistics.

The XY plane represents appearance information, while the XT plane gives a visual impression of one row changing in time and YT describes the motion of one column in temporal space. The LBP codes are extracted for all pixels from the XY, XT and YT planes, denoted as XY-LBP, XT-LBP and YT-LBP, and histograms from these planes are computed and concatenated into a single histogram. In such a representation, a dynamic texture is encoded by an appearance (XY-LBP) and two spatial temporal (XT-LBP and YT-LBP) co-occurrence statistics.

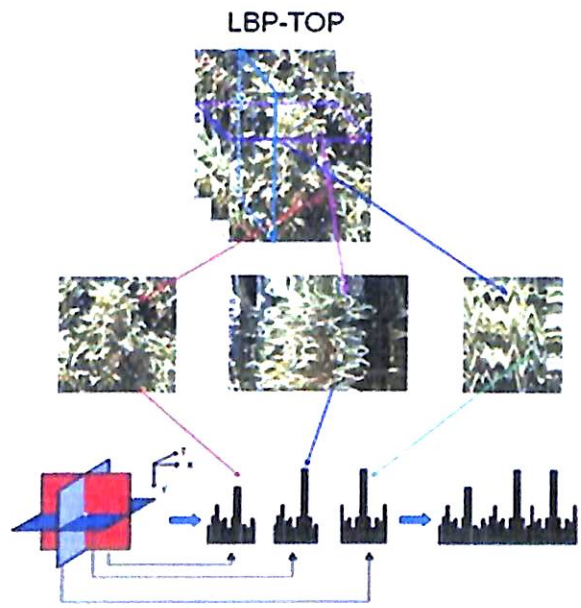

Figure 9 – LBP from three orthogonal planes.

Setting the radius in the time axis to be equal to the radius in the space axis is not reasonable for dynamic textures. So we have different radius parameters in space and time to set. In the XT and YT planes, different radii can be assigned to sample neighboring points in space and time. More generally, the radii in axes X, Y and T, and the number of neighboring points in the XY, XT and YT planes can also be different.

The LBP methodology has led to significant progress in texture analysis. It is widely used all over the world both in research and applications. Due to its discriminative power and computational simplicity, the method has been very

regarded as textureproblems, suchas faceanalysis and motion analysis (Pietikäinen et al. 2011). For a bibliographyof LBP-related researchand links to manypapers, see http://www.cse.oulu.fi/MVG/LBP_Bibliography.

To increase theapplicabilityof LBP, various extensions and modifications of it have been proposed. For example, Liaoet al. (2009) proposed dominant local binarypatterns which make useof the most frequentlyoccurred patterns of LBP to improve the recognition accuracy. The useof interest region descriptors (suchas SIFT) to various computer vision problems has been of great interest recently. For this purpose, a novel descriptor combining the strengths of SIFTand LBP was proposed (Heikkilä et al. 2009), in whichcenter-symmetric local binarypatterns (CS-LBP) were used to replace the gradient operator used by the SIFToperator. Mäenpää and Pietikäinen (2004) proposed an opponent color LBP, and investigated joint and separate useof color and texture in classification. Thecombination of the LBPs and Gabor features has been investigated (Tan et al. 2007, Wang et al. 2009).

The first texture-based method for background subtraction was proposed byHeikkilä and Pietikäinen (2006). Eachpixel is modeled as a groupof adaptive local binarypattern histograms that arecalculated over acircular region around thepixel. The was shown to be tolerant to illumination variations, the multimodalityof the background, and the introduction or removal of background objects. Furthermore, the method is capable for real-timeprocessing. Apreprocessing algorithm based on the LBPs has been developed tohandle variations in illumination in a faceauthentication system (Heuschet al. 2006). The useof LBPs in the recognition of actions was considered by (Kellokumpu et al. 2010). The useof LBPs in the facial ageclassification has been investigated (Wang et al. 2009). Other related LBP-based approaches to theseproblems have been proposed recently.

In addition to faceand facial expression recognition, the LBPhas also been used in manyother applications of biometrics, including eye localization, iris recognition, fingerprint recognition, palmprint recognition, gait recognition and facial ageclassification. References to manyof these works can be found from the LBP bibliography.[12]

3.1.COMPUTER VISION

Computer vision is a field that includes methods for acquiring, processing, analyzing, and understanding images and, in general, high-dimensional data from the real world in order toproduce numerical or symbolic information, e.g., in the forms of decisions. A theme in the development of this field has been to duplicate theabilities of human vision byelectronicallyperceiving and understanding an image. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with theaid of geometry, physics, statistics, and learning theory. Computer vision has also been described as theenterpriseof automating and integrating a wide rangeof processes and representations for vision perception.

Applications range from tasks suchas industrial machine vision systems which, say, inspect bottles speeding byon aproduction line, to research intoartificial intelligenceand computers or robots that can comprehend the world around them. Thecomputer vision and machine vision fields have significant overlap. Computer vision covers thecore technologyof automated imageanalysis which is used in many fields. Machine vision usually refers toaprocess of combining automated imageanalysis withother methods and technologies toprovideautomated inspection and robot guidance in industrial applications.

As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images. The image datacan take many forms, suchas video sequences, views from multiplecameras, or multi-dimensional data from a medical scanner.

As a technological discipline, computer vision seeks toapply its theories and models to theconstruction of computer vision systems. Examples of applications of computer vision include systems for:

- Controlling processes, e.g., an industrial robot;
- Navigation, e.g., byan autonomous vehicleor mobile robot;

- Detecting events, e.g., for visual surveillanceor peoplecounting;
- Organizing information, e.g., for indexing databases of images and image sequences;
- Modeling objects or environments, e.g., medical imageanalysis or topographical modeling;
- Interaction, e.g., as the input toa device for computer-human interaction, and
- Automatic inspection, e.g., in manufacturing applications.

Sub-domains of computer vision include scene reconstruction, event detection, video tracking, object recognition, learning, indexing, motion estimation, and image restoration.

In most practical computer vision applications, thecomputers arepre-programmed to solveaparticular task, but methods based on learning are now becoming increasinglycommon.

Areas of artificial intelligence deal withautonomous planning or deliberation for robotical systems to navigate throughan environment. A detailed understanding of theseenvironments is required to navigate through them. Information about theenvironment could beprovided byacomputer vision system, acting as a vision sensor and providing high-level information about theenvironment and the robot.

Artificial intelligenceand computer vision shareother topics suchas pattern recognition and learning techniques. Consequently, computer vision is sometimes seen as apart of theartificial intelligence field or thecomputer science field in general.

Solid-statephysics is another field that is closely related tocomputer vision. Most computer vision systems relyon image sensors, which detect electromagnetic radiation which is typically in the form of either visibleor infra-red light. The sensors are designed using quantum physics. Theprocess by which light interacts withsurfaces is explained using physics. Physics explains the behavior of optics whichareacorepart of most imaging systems. Sophisticated image sensors even require quantum mechanics toprovideacomplete understanding of the image

formation process. Also, various measurement problems in physics can beaddressed using computer vision, for example motion in fluids.

A third field whichplays an important role is neurobiology, specifically the studyof the biological vision system. Over the last century, therehas been an extensive studyof eyes, neurons, and the brain structures devoted toprocessing of visual stimuli in bothhumans and various animals. This has led toacoarse, yet complicated, description of how "real" vision systems operate in order to solvecertain vision related tasks. These results have led toa subfield within computer vision whereartificial systems are designed to mimic theprocessing and behavior of biological systems, at different levels of complexity. Also, someof the learning-based methods developed within computer vision (e.g. neural net based imageand featureanalysis and classification) have their background in biology.

Some strands of computer vision researchareclosely related to the studyof biological vision – indeed, just as many strands of AI researchareclosely tied with research intohuman consciousness, and the useof stored knowledge to interpret, integrateand utilize visual information. The field of biological vision studies and models thephysiological processes behind visual perception in humans and other animals. Computer vision, on theother hand, studies and describes theprocesses implemented in softwareand hardware behind artificial vision systems. Interdisciplinaryexchange between biological and computer vision has proven fruitful for both fields.

Yet another field related tocomputer vision is signal processing. Many methods for processing of one-variable signals, typically temporal signals, can beextended in a natural way toprocessing of two-variable signals or multi-variable signals in computer vision. However, becauseof the specific natureof images thereare many methods developed within computer vision whichhave nocounterpart in processing of one-variable signals. Together with the multi-dimensionalityof the signal, this defines a subfield in signal processing as apart of computer vision.

Beside theabove mentioned views on computer vision, manyof the related research topics can also be studied from apurely mathematical point of view. For example, many methods in computer vision are based on statistics, optimization or geometry. Finally, a significant part of the field is devoted to the implementation aspect of computer vision; how existing methods can be realized in various combinations of softwareand hardware, or how these methods can be modified in order to gain processing speed without losing too muchperformance.

The fields most closely related tocomputer vision are imageprocessing, imageanalysis and machine vision. There is a significant overlap in the rangeof techniques and applications that thesecover. This implies that the basic techniques that are used and developed in these fields are moreor less identical, something whichcan be interpreted as there is onlyone field with different names. On theother hand, it appears to be necessary for research groups, scientific journals, conferences and companies topresent or market themselves as belonging specifically tooneof these fields and, hence, various characterizations which distinguisheachof the fields from theothers have been presented.

Computer vision is, in some ways, the inverseof computer graphics. Whilecomputer graphics produces image data from 3D models, computer vision often produces 3D models from image data. There is alsoa trend towards acombination of the two disciplines, e.g., as explored in augmented reality.

The following characterizations appear relevant but should not be taken as universallyaccepted:

Imageprocessing and imageanalysis tend to focus on 2D images, how to transform one image toanother, e.g., bypixel-wiseoperations suchas contrast enhancement, local operations suchas edgeextraction or noise removal, or geometrical transformations suchas rotating the image. This characterization implies that imageprocessing/analysis neither requireassumptions nor produce interpretations about the imagecontent.

Computer vision includes 3D analysis from 2D images. This analyzes the 3D sceneprojected ontooneor several images, e.g., how to reconstruct structureor other information about the 3D scene from oneor several images. Computer vision often relies on moreor less complexassumptions about the scene depicted in an image.

Machine vision is theprocess of applying a rangeof technologies & methods toprovide imaging-based automatic inspection, process control and robot guidance in industrial applications.Machine vision tends to focus on applications, mainly in manufacturing, e.g., vision based autonomous robots and systems for vision based inspection or measurement. This implies that image sensor technologies and control theoryoften are integrated with theprocessing of image data tocontrol a robot and that real-timeprocessing is emphasised by means of efficient implementations in hardwareand software. It also implies that theexternal conditions suchas lighting can beand areoften morecontrolled in machine vision than theyare in general computer vision, whichcan enable the useof different algorithms.

There is alsoa field called imaging whichprimarily focus on theprocess of producing images, but sometimes also deals withprocessing and analysis of images. For example, medical imaging includes substantial work on theanalysis of image data in medical applications.

Finally, pattern recognition is a field which uses various methods toextract information from signals in general, mainly based on statistical approaches. A significant part of this field is devoted toapplying these methods to image data.

Typical tasks of computer vision

Eachof theapplication areas described aboveemploya rangeof computer vision tasks; moreor less welldefined measurement problems or processing problems, whichcan be solved using a varietyof methods. Someexamples of typical computer vision tasks arepresented below.

Recognition

Theclassical problem in computer vision, imageprocessing, and machine vision is that of determining whether ornot the image datacontains some specificobject, feature, or activity. This task can normally be solved robustlyand without effort byahuman, but is still not satisfactorily solved in computer vision for the general case –arbitraryobjects in arbitrary situations. Theexisting methods for dealing with this problem can at best solve itonly for specificobjects, suchas simple geometricobjects (e.g., polyhedra), human faces, printed or handwritten characters, or vehicles, and in specific situations, typically described in terms of well-defined illumination,background, and poseof theobject relative to thecamera.

Different varieties of the recognition problem are described in the literature:

- Object recognition – oneor several pre-specified or learned objects or object classes can berecognized, usually together with their 2D positions in the imageor 3D poses in the scene. GoogleGoggles provides a stand-aloneprogram illustration of this function.
- Identification – an individual instanceof an object is recognized. Examples include identification of aspecificperson's faceor fingerprint, or identification of a specific vehicle.
- Detection – the image dataare scanned for a specificcondition. Examples include detection of possibleabnormal cells or tissues in medical images or detection of a vehicle in an automatic road tollsystem.Detection based on relatively simpleand fast computations is sometimes used for finding smaller regionsof interesting image data whichcan be further analyzed by morecomputationally demanding techniques toproduceacorrect interpretation.

Severalspecialized tasks based on recognition exist, suchas:

- Content-based image retrieval – finding all images in a larger set of images whichhavea specificcontent. Thecontent can be specified in different ways, for example in terms ofsimilarity relativea targetimage (give meall images similar to imageX), or in terms of high-levelsearchcriteria given as text input(give

meall images whichcontains manyhouses, are taken during winter, and have nocars in them).

- Poseestimation – estimating theposition or orientation of a specificobject relative to thecamera. Anexampleapplication for this technique would beassisting a robot arm in retrieving objects from aconveyor belt in an assembly line situation or picking parts from a bin.
- Optical character recognition (OCR) – identifying characters in images of printed or handwritten text,usually witha view toencoding the text in a format moreamenable toediting or indexing (e.g. ASCII).
- 2D Code reading Reading of 2D codes suchas data matrixand QR codes.
- Facial recognition

Motion analysis

Several tasks relate to motion estimation wherean image sequence is processed toproducean estimateof thevelocityeither at eachpoints in the imageor in the 3D scene, or even of thecamera that produces the images .

- Egomotion – determining the 3D rigid motion (rotation and translation) of thecamera from an imagesequenceproduced by thecamera.
- Tracking – following the movements of a (usually) smaller set of interest points or objects (e.g., vehiclesor humans) in the image sequence.
- Optical flow – to determine, for eachpoint in the image, how that point is moving relative to the imageplane, i.e., its apparent motion. This motion is a result bothof how thecorresponding 3D point is movingin the sceneand how thecamera is moving relative to the scene.

Scene reconstruction

Given oneor (typically) more images of a scene, or a video, scene reconstruction aims at computing a 3D modelof the scene. In the simplest case the model can bea set of 3D points. More sophisticated methods produceacomplete 3D surface model. Theadvent of 3D imaging not requiring motion or scanning, and

related processingalgorithms is enabling rapid advances in this field. Grid-based 3D sensing can be used toacquire 3D imagesfrom multipleangles. Algorithms are now available to stitch multiple 3D images together intopoint clouds and3D models.

Image restoration

Theaim of image restoration is the removal of noise (sensor noise, motion blur, etc.) from images. The simplestpossibleapproach for noise removal is various types of filters suchas low-pass filters or median filters. Moresophisticated methods assumea model of how the local image structures look like, a model which distinguishesthem from the noise. By first analysing the image data in terms of the local image structures, suchas lines oredges, and then controlling the filtering based on local information from theanalysis step, a better level of noiseremoval is usuallyobtained compared to the simpler approaches.

An example in this field is the inpainting.

Computer vision system methods

Theorganization of acomputer vision system is highlyapplication dependent. Some systems are stand-aloneapplications which solvea specific measurement or detection problem, whileothers constitutea sub-system of a larger design which, for example, alsocontains sub-systems for control of mechanical actuators, planning, information databases, man-machine interfaces, etc. The specific implementation of acomputer vision system also depends on if its functionality is pre-specified or if somepart of it can be learned or modified during operation. Many functions are unique to theapplication. Thereare, however, typical functions whichare found in manycomputer vision systems.

- **Imageacquisition** – A digital image is produced byoneor several image sensors, which, besides various types of light-sensitivecameras, include range sensors, tomography devices, radar, ultra-soniccameras, etc. Depending on the typeof sensor, the resulting image data is an ordinary 2D image, a 3D volume, or an image sequence. Thepixel values typicallycorrespond to light intensity in oneor

several spectral bands (gray images or colour images), but can also be related to various physical measures, suchas depth, absorption or reflectanceof sonicor electromagnetic waves, or nuclear magnetic resonance.

- **Pre-processing** – Beforeacomputer vision method can beapplied to image data in order toextract some specificpieceof information, it is usually necessary toprocess the data in order toassure that it satisfies certain assumptions implied by the method. Examples are
    - Re-sampling in order toassure that the imagecoordinate system is correct.
    - Noise reduction in order toassure that sensor noise does not introduce false information.
    - Contrast enhancement toassure that relevant information can be detected.
    - Scale space representation toenhance image structures at locallyappropriate scales.

- **Featureextraction** – Image features at various levels of complexityareextracted from the image data. Typical examples of such features are
    - Lines, edges and ridges.
    - Localized interest points suchas corners, blobs or points.

      Morecomplex features may be related to texture, shapeor motion.

- **Detection/segmentation** – At somepoint in theprocessing a decision is madeabout which imagepoints or regions of the imageare relevant for further processing. Examples are
    - Selection of a specific set of interest points
    - Segmentation of oneor multiple image regions whichcontain a specificobject of interest.

- **High-level processing** – At this step the input is typicallya small set of data, for examplea set of points or an image region which is assumed tocontain a specificobject. The remaining processing deals with, for example:
    - Verification that the data satisfy model-based and application specificassumptions.
    - Estimation of application specificparameters, suchas object poseor object size.
    - Image recognition – classifying a detected object into different categories.
    - Image registration – comparing and combining two different views of the sameobject.

- **Decision making** Making the final decision required for theapplication, for example:
    - Pass/fail on automatic inspection applications
    - Match / no-match in recognition applications
    - Flag for further human review in medical, military, securityand recognition applications

3.2. TRAFFICSIGN DETECTION APPLICATION

As I started working on theTraffic sign recognition (TSR), Ihave been facing a big job to do. First of all, Ihad no ideaabout imageprocessing and pattern detection. Therefore, this project opened a whole new world for me. Rather than building upown full of bugs classier I search web for existing libraries and I found a lot, like: CellProfiler, imageJ, FIJI, Endrov, Mango, OpenCV, GNU Octave, GemIdent, 3D Slicer, OsiriX, FreeSurfer, InVesalius, FMRIB Software Libraryand etc. But the best one for me was OpenCV. Because it is open sourceand has a lot of ports and rapidly developping.

3.3.OPENCV

OpenCV (Open SourceComputer Vision Library) is a libraryof programming functions mainlyaimed at real-timecomputer vision, developed byIntel, and now supported by Willow Garageand Itseez. It is free for use under theopen sourceBSD license. The library is cross-platform. It focuses mainlyon real-time imageprocessing. If the library finds Intel's Integrated PerformancePrimitives on the system, it will use theseproprietaryoptimized routines toaccelerate itself.

Officially launched in 1999, theOpenCV project was initiallyan Intel Research initiative toadvanceCPU-intensiveapplications, part of a series of projects including real-time ray tracing and 3D display walls. The main contributors to theproject included a number of optimization experts in Intel Russia, as well as Intel's Performance LibraryTeam. In theearly days of OpenCV, the goals of theproject were described as

Advance vision research byproviding not onlyopen but alsooptimized code for basic vision infrastructure. No more reinventing the wheel.

Disseminate vision knowledge byproviding acommon infrastructure that developers could build on, so that code would be more readily readableand transferable.

Advance vision-based commercial applications by making portable, performance-optimized codeavailable for free—witha license that did not require to beopen or free themselves.

The first alpha version of OpenCV was released to thepublicat theIEEEConferenceon Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. In mid-2008, OpenCV obtained corporate support from Willow Garage, and is now again under active development. A version 1.1 "pre-release" was released in October 2008.

The second major releaseof theOpenCV was on October 2009. OpenCV 2 includes major changes to theC++ interface, aiming at easier, more type-safepatterns, new functions, and better implementations for existing ones in terms of performance (especiallyon multi-core systems). Official releases now occur every six months and development is now done byan independent Russian team supported bycommercial corporations.

In August 2012, support for OpenCV was taken over bya non-profit foundation, OpenCV.org, which maintains a developer and user site.

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive thoughextensiveolder C interface. Thereare now full interfaces in Python, Javaand MATLAB/OCTAVE (as of version 2.5). TheAPI for these interfaces can be found in theonline documentation. Wrappers in other languages suchas C#, Ch, Rubyhave been developed toencourageadoption bya wider audience.

All of the new developments and algorithms in OpenCV are now developed in theC++ interface.

ACUDA-based GPU interfacehas been in progress since September 2010.

An OpenCL-based GPU interfacehas been in progress sinceOctober 2012, documentation for version 2.4.5 can be found here.

OpenCV runs on Windows, Android, Maemo, FreeBSD, OpenBSD, iOS, BlackBerry 10, Linuxand OS X. The user can get official releases from SourceForge, or take thecurrent snapshot under SVN from there. OpenCV uses CMake.

OpenCV has a modular structure, which means that thepackage includes several shared or static libraries. The following modules areavailable:

- core - acompact module defining basic data structures, including the dense multi-dimensional arrayMat and basic functions used byall other modules.

- imgproc - an imageprocessing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affineand perspective warping, generic table-based remapping), color spaceconversion, histograms, and soon.
- video - a videoanalysis module that includes motion estimation, background subtraction, and object tracking algorithms.
- calib3d - basic multiple-view geometryalgorithms, singleand stereocameracalibration, object poseestimation, stereocorrespondencealgorithms, and elements of 3D reconstruction.
- features2d - salient feature detectors, descriptors, and descriptor matchers.
- objdetect - detection of objects and instances of thepredefined classes (for example, faces, eyes, mugs, people, cars, and soon).
- highgui - an easy-to-use interface to videocapturing, imageand videocodecs, as well as simple UIcapabilities.
- gpu - GPU-accelerated algorithms from different OpenCV modules.

... someother helper modules, suchas FLANN and Google test wrappers, Python bindings, and others.

The further chapters of the document describe functionalityof each module. But first, make sure to get familiar with thecommon APIconcepts used thoroughly in the library.

3.4.OPENCV ANDROID

OpenCV Manager is an Android service targeted to manageOpenCV library binaries on end users devices. It allows sharing theOpenCV dynamic libraries between applications on the same device. TheManager provides the following benefits:

1. Less memory usage. All apps use the same binaries from serviceand do not keep native libs inside themselves;
2. Hardware specificoptimizations for all supported platforms;

3. Trusted OpenCV library source. All packages withOpenCV arepublished on GooglePlay market;
4. Regular updates and bug fixes;

Usage model for end user

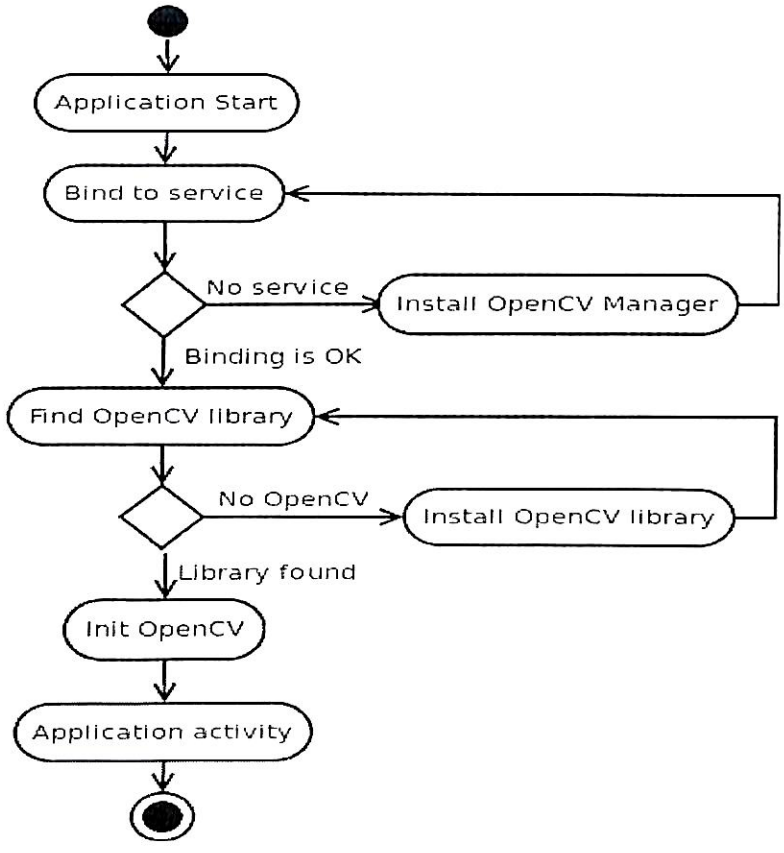

Figure10 – model for end user

First OpenCV app:

1. AnyOpenCV-dependent app is installed from GooglePlay marketplaceor manually;
2. At the first launch, it suggests installation of OpenCV Manager;
3. Then OpenCV Manager is downloaded and installed, using the GooglePlayapplication.
4. When Manager has been started, theapplication suggests installation of OpenCV library for the target devicearchitecture if it is necessary;
5. After the installation is finished, theapp may be launched.

Subsequent launches of OpenCV apps:

1. AnyOpenCV-dependent app is installed from GooglePlay market or manually;

2. At the first launch, the app starts as usually;
3. If the selected OpenCV version is not installed, OpenCV Manager suggests installing OpenCV library for the target device through Google Play marketplace;
4. After the installation is finished, the app may be launched.

Architecture of OpenCV Manager

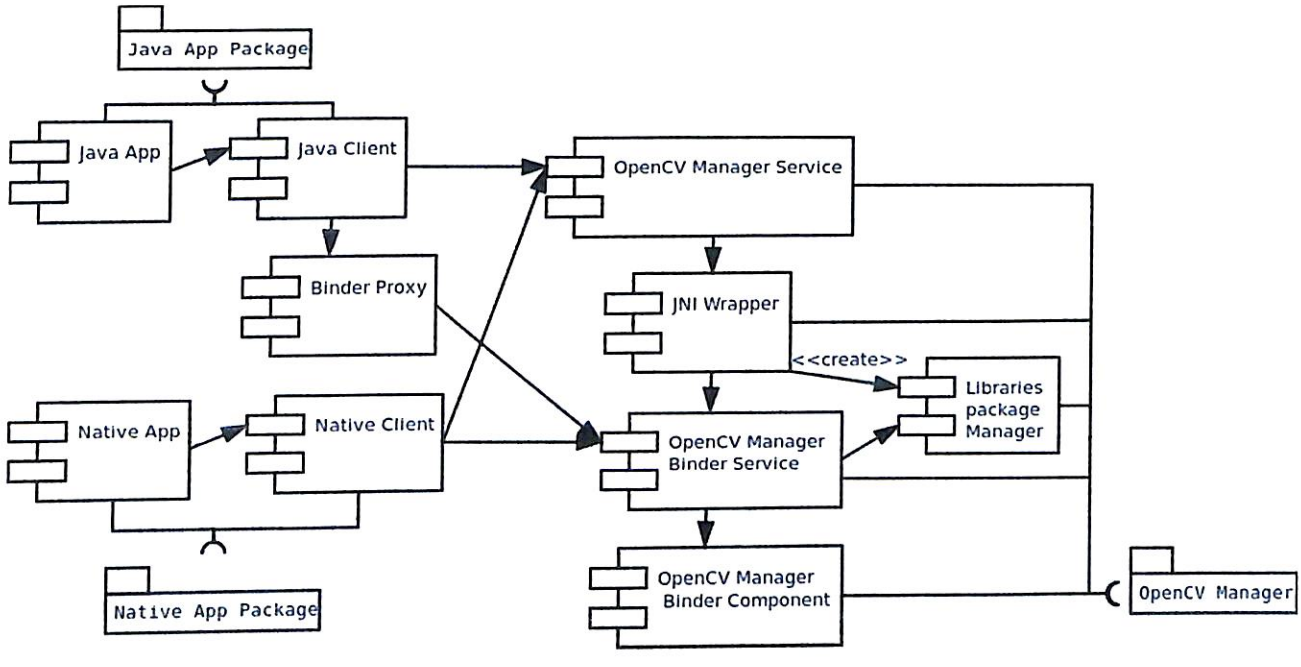

Figure 11 – openCV manager

3.5.OPENCV ANDROID CASCADE CLASSIFER

Cascading is based on the concatenation of several classifiers, using all information collected from the output from a given classifier as additional information for the next classifier in the cascade. Unlike voting or stacking ensembles, which are multiexpert systems, cascading is a multistage one.

The work with a cascade classifier inlcudes two major stages: training and detection. Detection stage is described in a documentation of objdetect module of general OpenCV documentation. Documentation gives some basic information about cascade classifier. Current guide is describing how to train a cascade classifier: preparation of a training data and running the training application.

There are two applications in OpenCV to train cascade classifier: opencv_haartraining and opencv_traincascade. opencv_traincascade is a newer version, written in C++ in accordance to OpenCV 2.x API. But the main difference between this two applications is that opencv_traincascade supports both Haar and LBP (Local Binary Patterns) features. LBP features are integer in contrast to Haar features, so both training and detection with LBP are several times faster then with Haar features. Regarding the LBP and Haar detection quality, it depends on training: the quality of training dataset first of all and training parameters too. It's possible to train a LBP-based classifier that will provide almost the same quality as Haar-based one.

opencv_traincascade and opencv_haartraining store the trained classifier in different file formats. Note, the newer cascade detection interface support both formats.opencv_traincascade can save (export) a trained cascade in the older format. But opencv_traincascade and opencv_haartraining can not load (import) a classifier in another format for the futher training after interruption.

Note that opencv_traincascade application can use TBB for multi-threading. To use it in multicore mode OpenCV must be built with TBB.

Also there are some auxilary utilities related to the training.

- opencv_createsamples is used to prepare a training dataset of positive and test samples. opencv_createsamples produces dataset of positive samples in a format that is supported by both opencv_haartraining and opencv_traincascade applications. The output is a file with *.vec extension, it is a binary format which contains images.
- opencv_performance may be used to evaluate the quality of classifiers, but for trained by opencv_haartraining only. It takes a collection of marked up images, runs the classifier and reports the performance, i.e. number of found objects, number of missed objects, number of false alarms and other information.

Since opencv_haartraining is an obsolete application, only opencv_traincascade will be described futher. opencv_createsamples utility is needed to prepare a training data for opencv_traincascade, so it will be described too.

For training we need a set of samples. There are two types of samples: negative and positive. Negative samples correspond to non-object images. Positive samples correspond to images with detected objects. Set of negative samples must be prepared manually, whereas set of positive samples is created using opencv_createsamples utility.

Negative samples are taken from arbitrary images. These images must not contain detected objects. Negative samples are enumerated in a special file. It is a text file in which each line contains an image filename (relative to the directory of the description file) of negative sample image. This file must be created manually. Note that negative samples and sample images are also called background samples or background samples images, and are used interchangeably in this document. Described images may be of different sizes. But each image should be (but not nessesarily) larger then a training window size, because these images are used to subsample negative image to the training size.

An example of description file:

Directory structure:

```
/img
  img1.jpg
  img2.jpg
bg.txt
```

File bg.txt:

```
img/img1.jpg
img/img2.jpg
```

Positive samples are created by opencv_createsamples utility. They may be created from a single image with object or from a collection of previously marked up images.

Please note that you need a large dataset of positive samples before you give it to the mentioned utility, because it only applies perspective transformation. For example you may need only one positive sample for absolutely rigid object like an OpenCV logo, but you definetely need hundreds and even thousands of positive samples for faces. In the case of faces you should consider all the race and age groups, emotions and perhaps beard styles.

So, a single object image may contain a company logo. Then a large set of positive samples is created from the given object image by random rotating, changing the logo intensity as well as placing the logo on arbitrary background. The amount and range of randomness can be controlled by command line arguments ofopencv_createsamples utility.

Command line arguments:

- -vec <vec_file_name>

  Name of the output file containing the positive samples for training.

- -img <image_file_name>

  Source object image (e.g., a company logo).

- -bg <background_file_name>

  Background description file; contains a list of images which are used as a background for randomly distorted versions of the object.

- -num <number_of_samples>

  Number of positive samples to generate.

- -bgcolor <background_color>

  Background color (currently grayscale images are assumed); the background color denotes the transparent color. Since there might be compression artifacts, the amount of color tolerance can be specified by-bgthresh. All pixels withing bgcolor-bgthresh andbgcolor+bgthresh range are interpreted as transparent.

- -bgthresh <background_color_threshold>

- -inv

  If specified, colors will be inverted.

- -randinv

  If specified, colors will be inverted randomly.

- -maxidev <max_intensity_deviation>

  Maximal intensity deviation of pixels in foreground samples.

- -maxxangle <max_x_rotation_angle>

- -maxyangle <max_y_rotation_angle>

- -maxzangle <max_z_rotation_angle>

  Maximum rotation angles must be given in radians.

- -show

  Useful debugging option. If specified, each sample will be shown. Pressing Esc will continue the samples creation process without.

- -w <sample_width>

  Width (in pixels) of the output samples.

- -h <sample_height>

  Height (in pixels) of the output samples.

For following procedure is used to create a sample object instance: The source image is rotated randomly around all three axes. The chosen angle is limited my -max?angle. Then pixels having the intensity from [bg_color-bg_color_threshold;bg_color+bg_color_threshold] range are interpreted as transparent. White noise is added to the intensities of the foreground. If the -inv key is specified then foreground pixel intensities are inverted. If -randinv key is specified then algorithm randomly selects whether inversion should be applied to this sample. Finally, the obtained image is placed onto an arbitrary background from the background description file, resized to the desired size specified by -w and -h and stored to the vec-file, specified by the -vec command line option.

Positive samples also may be obtained from a collection of previously marked up images. This collection is described by a text file similar to background description file. Each line of this file corresponds to an image. The first element of the line is the filename. It is followed by the number of object instances. The following numbers are the coordinates of objects bounding rectangles (x, y, width, height).

An example of description file:

Directory structure:

```
/img
  img1.jpg
  img2.jpg
info.dat
```

File info.dat:

```
img/img1.jpg  1  140 100 45 45
img/img2.jpg  2  100 200 50 50   50 30 25 25
```

Image img1.jpg contains single object instance with the following coordinates of bounding rectangle: (140, 100, 45, 45). Image img2.jpg contains two object instances.

In order to create positive samples from such collection, -info argument should be specified instead of -img:

- -info <collection_file_name>

Description file of marked up images collection.

The scheme of samples creation in this case is as follows. The object instances are taken from images. Then they are resized to target samples size and stored in output vec-file. No distortion is applied, so the only affecting arguments are -w, -h, -show and -num.

opencv_createsamples utility may be used for examining samples stored in positive samples file. In order to do this only -vec, -w and -h parameters should be specified.

Note that for training, it does not matter how vec-files with positive samples are generated. But opencv_createsamples utility is the only one way to collect/create a vector file of positive samples, provided by OpenCV.

Example of vec-file is available hereopencv/data/vec_files/trainingfaces_24-24.vec. It can be used to train a face detector with the following window size: -w 24 -h 24.

The next step is the training of classifier. As mentioned aboveopencv_traincascade or opencv_haartraining may be used to train a cascade classifier, but only the newer opencv_traincascade will be described futher.

Command line arguments of opencv_traincascade application grouped by purposes:

1. Common arguments:
    - -data <cascade_dir_name>

      Where the trained classifier should be stored.

    - -vec <vec_file_name>

      vec-file with positive samples (created byopencv_createsamples utility).

    - -bg <background_file_name>

      Background description file.

    - -numPos <number_of_positive_samples>
    - -numNeg <number_of_negative_samples>

      Number of positive/negative samples used in training for every classifier stage.

    - -numStages <number_of_stages>

      Number of cascade stages to be trained.

    - -precalcValBufSize<precalculated_vals_buffer_size_in_Mb>

      Size of buffer for precalculated feature values (in Mb).

    - -precalcIdxBufSize<precalculated_idxs_buffer_size_in_Mb>

      Size of buffer for precalculated feature indices (in Mb). The more memory you have the faster the training process.

    - -baseFormatSave

      This argument is actual in case of Haar-like features. If it is specified, the cascade will be saved in the old format.

2. Cascade parameters:

   - -stageType <BOOST(default)>

     Type of stages. Only boosted classifier are supported as a stage type at the moment.

   - -featureType<{HAAR(default), LBP}>

     Type of features: HAAR - Haar-like features, LBP - local binary patterns.

   - -w <sampleWidth>

   - -h <sampleHeight>

     Size of training samples (in pixels). Must have exactly the same values as used during training samples creation (opencv_createsamples utility).

3. Boosted classifer parameters:

   - -bt <{DAB, RAB, LB, GAB(default)}>

     Type of boosted classifiers: DAB - Discrete AdaBoost, RAB - Real AdaBoost, LB - LogitBoost, GAB - Gentle AdaBoost.

   - -minHitRate <min_hit_rate>

     Minimal desired hit rate for each stage of the classifier. Overall hit rate may be estimated as (min_hit_rate^number_of_stages).

   - -maxFalseAlarmRate <max_false_alarm_rate>

     Maximal desired false alarm rate for each stage of the classifier. Overall false alarm rate may be estimated as (max_false_alarm_rate^number_of_stages).

   - -weightTrimRate <weight_trim_rate>

Specifies whether trimming should be used and its weight. A decent choice is 0.95.

- -maxDepth <max_depth_of_weak_tree>

  Maximal depth of a weak tree. A decent choice is 1, that is case of stumps.

- -maxWeakCount <max_weak_tree_count>

  Maximal count of weak trees for every cascade stage. The boosted classifier (stage) will have so many weak trees (<=maxWeakCount), as needed to achieve the given -maxFalseAlarmRate.

4. Haar-like feature parameters:

   - -mode <BASIC (default) | CORE | ALL>

     Selects the type of Haar features set used in training. BASIC use only upright features, while ALL uses the full set of upright and 45 degree rotated feature set.

5. Local Binary Patterns parameters:

   Local Binary Patterns don't have parameters.

After the opencv_traincascade application has finished its work, the trained cascade will be saved in cascade.xml file in the folder, which was passed as -dataparameter. Other files in this folder are created for the case of interrupted training, so you may delete them after completion of training.

Training is finished and you can test you cascade classifier!

3.6.ANDROID SDK & NDK

Android software development is the process by which new applications are created for the Android operating system. Applications are usually developed in the Java programming language using the Android Software Development Kit, but other

other development tools areavailable. As of October 2012, more than 700,000 applications have been developed for Android, withover 25 billion downloads. A June 2011 research indicated that over 67% of mobile developers used theplatform, at the timeof publication. In Q2 2012; around 105 million units of Android smartphones were shipped whichacquires a total shareof 68% in overall smartphones sale till Q2 2012.

TheAndroid software development kit (SDK) includes acomprehensive set of development tools. These includea debugger, libraries, ahandset emulator based on QEMU, documentation, samplecode, and tutorials. Currently supported development platforms includecomputers running Linux (any modern desktop Linux distribution), MacOS X 10.5.8 or later, Windows XPor later; for the moment onecannot developAndroid softwareon Android itself. Theofficially supported integrated development environment (IDE) is Eclipse using theAndroid Development Tools (ADT) Plugin, thoughIntelliJ IDEAIDE (all editions) fully supports Android development out of the box, and NetBeans IDEalso supports Android development viaaplugin. Additionally, developers may useany text editor toedit Javaand XML files, then usecommand line tools (Java Development Kit and ApacheAnt are required) tocreate, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing softwarepackage(s) remotely).

Enhancements toAndroid's SDK gohand in hand with theoverall Android platform development. The SDKalso supports older versions of theAndroid platform in case developers wish to target their applications at older devices. Development tools are downloadablecomponents, soafter onehas downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.

Android applications arepackaged in .apk format and stored under /data/app folder on theAndroid OS (the folder is accessibleonly to the root user for security reasons). APKpackagecontains .dex files (compiled bytecode files called Dalvik executables), resource files, etc.

TheAndroid Debug Bridge (ADB) is a toolkit included in theAndroid SDKpackage. It consists of bothclient and server-sideprograms that communicate withoneanother. TheADB is typicallyaccessed through thecommand-line interface.

The format for issuing commands through theADB is typically:

adb [-d|-e|-s <serialNumber>] <command>

In a security issue reported in March 2011, ADB was targeted as a vector toattempt to install a rootkit on connected phones using a "resourceexhaustion attack".

Fastboot is a diagnosticprotocol included with the SDKpackage used primarily to modify the flash filesystem viaa USBconnection from host computer. It requires that the device be started in a boot loader or Second Program Loader mode in whichonly the most basichardware initialization is performed. After enabling theprotocol on the device itself, it will accept a specific set of commands sent to it via USB using acommand line. Someof most commonly used fastboot commands include:

- flash - Rewrites apartition witha binary image stored on thehost computer.
- erase - Erases a specificpartition.
- reboot - Reboots the device intoeither the main operating system, the system recoverypartition or back into its bootloader.
- devices - Displays a list of all devices (with the serial number) connected to thehost computer.
- format - Format a specificpartition. The file system of thepartition must be recognized by the device.

Libraries written in Cand other languages can becompiled toARM, MIPS or x86 nativecodeand installed using theAndroid Native Development Kit. Nativeclasses can becalled from Javacode running under the Dalvik VM using the System.loadLibrarycall, which is part of the standard Android Javaclasses.

Completeapplications can becompiled and installed using traditional development tools. TheADB debugger gives a root shell under theAndroid Emulator whichallows nativeARM, MIPS or x86 code to be uploaded and executed. Nativecodecan becompiled using GCCon a standard PC. Running nativecode is complicated byAndroid's useof a non-standard C library (libc, known as Bionic). The underlying graphics device is availableas a framebuffer at /dev/graphics/fb0. The graphics library that Android uses toarbitrateand control access to this device is called the Skia Graphics Library (SGL), and it has been released under an open source licence. Skiahas backends for both Win32 and Unix, allowing the development of cross-platform applications, and it is the graphics engine underlying the GoogleChrome web browser.

Unlike Javaapplication development based on theEclipseIDE, the NDK is based on command-line tools and requires invoking them manually to build, deployand debug theapps. Several third-party tools allow integrating the NDK intoEclipseand Visual Studio.

The NDK is a toolset that allows you to implement parts of your app using native-code languages suchas Cand C++. For certain types of apps, this can behelpful soyou can reuseexisting code libraries written in these languages, but most apps do not need theAndroid NDK.

Before downloading the NDK, you should understand that the NDK will not benefit most apps. As a developer, you need to balance its benefits against its drawbacks. Notably, using nativecodeon Android generally does not result in a noticableperformance improvement, but it always increases your appcomplexity. In general, you should only use the NDK if it is essential toyour app—never becauseyou simplyprefer toprogram in C/C++.

TheAndroid framework provides two ways to use nativecode:

Writeyour application using theAndroid framework and use JNI toaccess theAPIs provided by theAndroid NDK. This techniqueallows you to takeadvantageof

theconvenienceof theAndroid framework, but still allows you to write nativecode when necessary. If you use this approach, your application must target specific, minimum Android platform levels, seeAndroid platform compatibility for more information.

Writea nativeactivity, whichallows you to implement the lifecyclecallbacks in nativecode. TheAndroid SDKprovides the NativeActivityclass, which is aconvenienceclass that notifies your nativecodeof anyactivity lifecyclecallbacks (onCreate(), onPause(), onResume(), etc). You can implement thecallbacks in your nativecode tohandle theseevents when theyoccur. Applications that use nativeactivities must be run on Android 2.3 (API Level 9) or later.

You cannot access features suchas Services and Content Providers natively, so if you want to use them or anyother framework API, you can still write JNIcode to do so.

Typical good candidates for the NDKare self-contained, CPU-intensiveoperations that don't allocate much memory, suchas signal processing, physics simulation, and soon. When examining whether or not you should develop in nativecode, think about your requirements and see if theAndroid framework APIs provide the functionality that you need.

The NDK includes a set of cross-toolchains (compilers, linkers, etc..) that can generate nativeARM binaries on Linux, OS X, and Windows (withCygwin) platforms.

It provides a set of system headers for stable nativeAPIs that are guaranteed to be supported in all later releases of theplatform:

- libc (C library) headers
- libm (math library) headers
- JNI interfaceheaders
- libz (Zlib compression) headers
- liblog (Android logging) header

- OpenGL ES 1.1 and OpenGL ES 2.0 (3D graphics libraries) headers
- libjnigraphics (Pixel buffer access) header (for Android 2.2 and above).
- AMinimal set of headers for C++ support
- OpenSL ES nativeaudio libraries
- Android nativeapplication APIS

The NDKalsoprovides a build system that lets you work efficiently withyour sources, without having tohandle the toolchain/platform/CPU/ABI details. You create very short build files to describe which sources tocompileand whichAndroid application will use them — the build system compiles the sources and places the shared libraries directly in your application project.

TheAndroid 3.1 platform (also backported toAndroid 2.3.4) introduces Android Open Accessory support, whichallows external USBhardware (an Android USBaccessory) to interact withan Android-powered device in a special "accessory" mode. When an Android-powered device is in accessory mode, theconnected accessoryacts as the USBhost (powers the bus and enumerates devices) and theAndroid-powered deviceacts as the USB device. Android USBaccessories are specifically designed toattach toAndroid-powered devices and adhere toa simpleprotocol (Android accessoryprotocol) that allows them to detect Android-powered devices that support accessory mode.

Android is created by theOpen Handset Alliance which is led by Google. Theearly feedback on developing applications for theAndroid platform was mixed. Issues cited include bugs, lack of documentation, inadequate QA infrastructure, and nopublic issue-tracking system. (Googleannounced an issue tracker on 18 January 2008.) In December 2007, MergeLab mobile startup founder Adam MacBeth stated, "Functionality is not there, is poorly documented or just doesn't work... It's clearly not ready for prime time." Despite this, Android-targeted applications began toappear the week after theplatform was announced. The first publiclyavailableapplication was the Snake game. TheAndroid Dev Phone is a SIM-unlocked and hardware-unlocked device that is designed for advanced developers. Whiledevelopers can use

regular consumer devices purchased at retail to test and use their applications, some developers maychoose not to usea retail device, preferring an unlocked or no-contract device.

Apreview releaseof theAndroid SDK was released on 12 November 2007. On 15 July 2008, theAndroid Developer ChallengeTeam accidentally sent an email toall entrants in theAndroid Developer Challengeannouncing that a new releaseof the SDK was available in a "private" download area. Theemail was intended for winners of the first round of theAndroid Developer Challenge. The revelation that Google was supplying new SDK releases to some developers and not others (and keeping this arrangement private) led to widely reported frustration within theAndroid developer communityat the time.

On 18 August 2008, theAndroid 0.9 SDK beta was released. This releaseprovided an updated and extended API, improved development tools and an updated design for thehome screen. Detailed instructions for upgrading areavailable to thosealready working withan earlier release. On 23 September 2008, theAndroid 1.0 SDK (Release 1) was released.According to the release notes, it included "mainly bug fixes, although some smaller features wereadded." It also included several APIchanges from the 0.9 version. Multiple versions have been released since it was developed .

3.7.OpenCV traincascade

TheOpenCV libraryprovides us a greatly interesting demonstration for a face detection. Furthermore, it provides us programs (or functions) that they used to train classifiers for their face detection system, called HaarTraining, so that wecan createour own object classifiers using these functions. It is interesting.

However, Icould not follow how OpenCV developers performed thehaartraining for their face detection system exactly because they did not provide us several information suchas what images and parameters they used for training. Theobjectiveof this report is toprovide step-by-stepprocedures for following people.

My working environment is Visual Studio + cygwin on Windows 7, or on Ubuntu. The cygwin is required because I use several UNIX commands. I am sure that you will use the cygwin (especially I mean UNIX commands) not only for this haartraining but also for others in the future if you are one of engineer or science people.

3.7.1. Data Preperation

For data I search up the internet and found wonderful dataset of traffic signs. Just here http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset

Positive Images

We need to collect positive images that contain only objects of interest, e.g., signs. For tha we use linux command which goes throu folder and saves name, width and height of images.

```
find <dir> -name '*.<ext>' -exec identify -format '%i 1 0
0 %w %h' \{\} \; > <description_file>
```

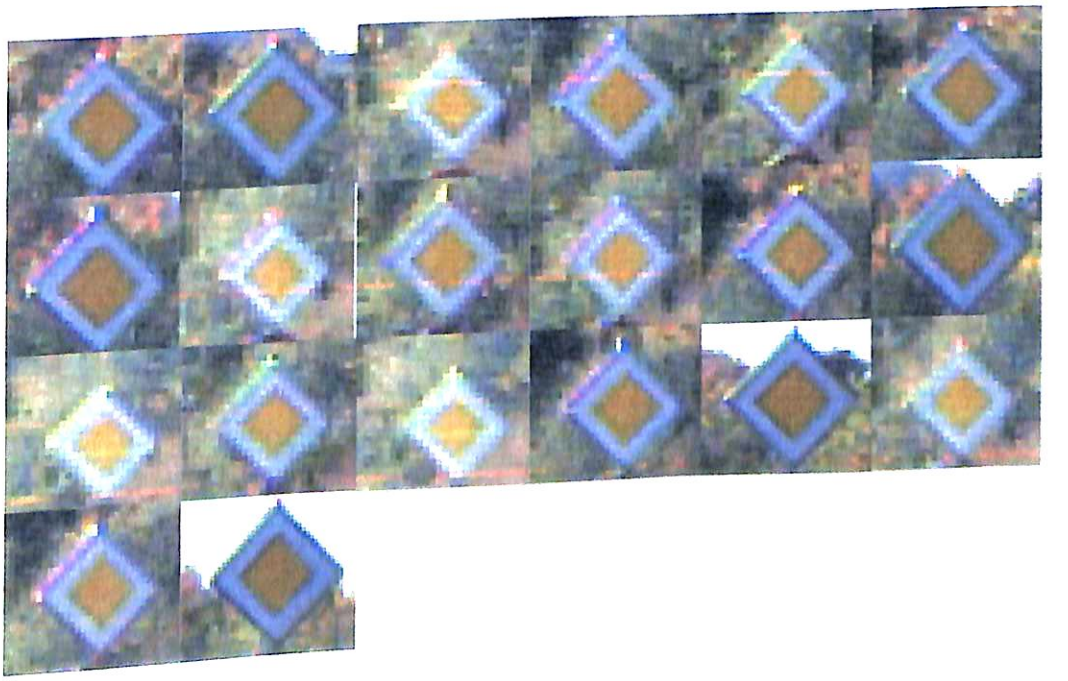

Figure 12 – postivie samples

3.7.2. NegativeImages

Alsocalled backgroud images. We need tocollect negative images that does not contain objects of interest, e.g., faces to train haarcascadeclassifier.It contains various places which does not contain signs.

```
find negative -name '*.jpg' > negative.dat
```

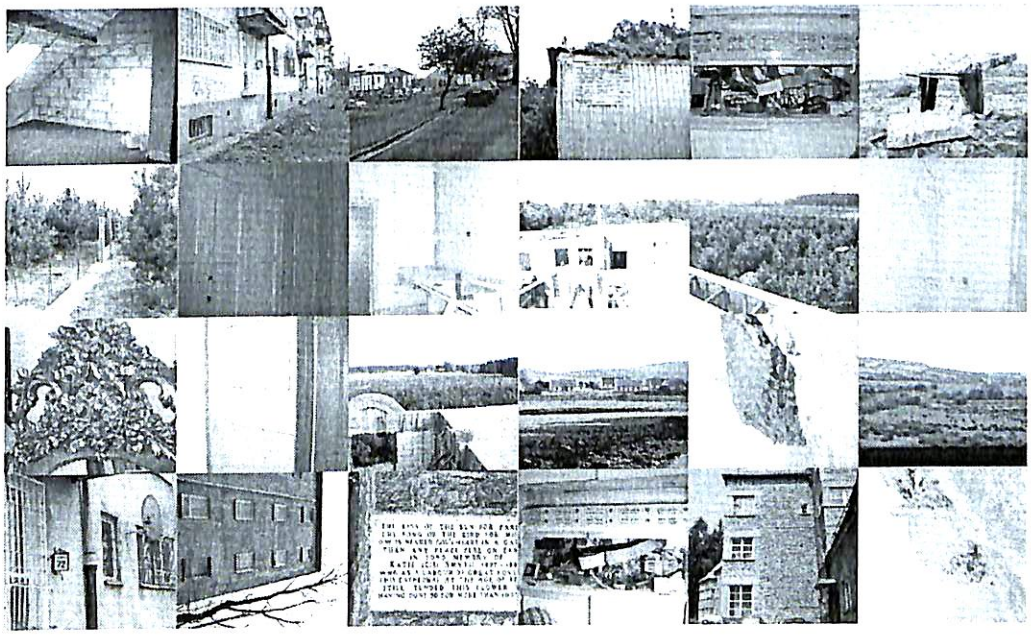

Figure 13 –negative samples

3.7.3. Create Samples

This is a list of options, but thereare mainly four functions and the meanings of options become different in different functions. It confuses us.

```
Usage: ./createsamples

  [-info<description_file_name>]

  [-img <image_file_name>]

  [-vec<vec_file_name>]

  [-bg <background_file_name>]
```

```
   [-num <number_of_samples = 1000>]

   [-bgcolor <background_color = 0>]

   [-inv] [-randinv] [-bgthresh<background_color_threshold
= 80>]

   [-maxidev <max_intensity_deviation = 40>]

   [-maxxangle<max_x_rotation_angle = 1.100000>]

   [-maxyangle<max_y_rotation_angle = 1.100000>]

   [-maxzangle<max_z_rotation_angle = 0.500000>]

   [-show [<scale = 4.000000>]]

   [-w <sample_width = 24>]

   [-h<sample_height = 24>]
```

For example,

```
$ createsamples -info samples.dat -vec samples.vec -w 20 -h 20
```

This generates samples without applying distortions. You may think this function as a file format conversion function.

The format of the<description_file_of_samples> is as follows:

```
[filename] [# of objects] [[xy widthheight] [... 2nd object] ...]
[filename] [# of objects] [[xy widthheight] [... 2nd object] ...]
[filename] [# of objects] [[xy widthheight] [... 2nd object] ...]
```

```
H:\training>opencv_createsamplesd   -infopositive.dat   -
vecpositive.vec -h 20 -w 20
Info file name: positive.dat
Img file name: (NULL)
Vec file name: positive.vec
BG  file name: (NULL)
Num: 1000
BG color: 0
BG threshold: 80
Invert: FALSE
Max intensity deviation: 40
Maxxangle: 1.1
Maxyangle: 1.1
Maxzangle: 0.5
Showsamples: FALSE
Width: 20
Height: 20
Createtrainingsamplesfromimagescollection...
positive.dat(701) : parseerrorDone. Created 700 samples
```

5.5.2. CascadeTraining

```
H:\training>opencv_traincascaded
Usage: opencv_traincascaded
  -data<cascade_dir_name>
  -vec<vec_file_name>
  -bg <background_file_name>
  [-numPos <number_of_positive_samples = 2000>]
  [-numNeg <number_of_negative_samples = 1000>]
  [-numStages <number_of_stages = 20>]
```

```
  [-
precalcValBufSize<precalculated_vals_buffer_size_in_Mb   =
256>]
  [-
precalcIdxBufSize<precalculated_idxs_buffer_size_in_Mb   =
256>]
  [-baseFormatSave]
--cascadeParams--
  [-stageType<BOOST(default)>]
  [-featureType<{HAAR(default), LBP, HOG}>]
  [-w <sampleWidth = 24>]
  [-h<sampleHeight = 24>]
--boostParams--
  [-bt <{DAB, RAB, LB, GAB(default)}>]
  [-minHitRate<min_hit_rate> = 0.995>]
  [-maxFalseAlarmRate<max_false_alarm_rate = 0.5>]
  [-weightTrimRate<weight_trim_rate = 0.95>]
  [-maxDepth<max_depth_of_weak_tree = 1>]
  [-maxWeakCount <max_weak_tree_count = 100>]
--haarFeatureParams--
  [-mode<BASIC(default) | CORE | ALL
--lbpFeatureParams--
--HOGFeatureParams--


H:\training>opencv_traincascaded      -datacascade      -
vecpositive.vec -bg negative.dat -numPos 500 -numNeg 700
-numStages 5 -featureType LBP -w 20 -h 20 -modeALL
PARAMETERS:
cascadeDirName: cascade
vecFileName: positive.vec
```

```
bgFileName: negative.dat
numPos: 500
numNeg: 700
numStages: 5
precalcValBufSize[Mb] : 256
precalcIdxBufSize[Mb] : 256
stageType: BOOST
featureType: LBP
sampleWidth: 20
sampleHeight: 20
boostType: GAB
minHitRate: 0.995
maxFalseAlarmRate: 0.5
weightTrimRate: 0.95
maxDepth: 1
maxWeakCount: 100

===== TRAINING 0-stage =====
<BEGIN
POS count : consumed   500 : 500
NEG count : acceptanceRatio    700 : 1
Precalculation time: 0.928
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|    0.996|     0.04|
+----+---------+---------+
END>
```

```
===== TRAINING 1-stage =====
<BEGIN
POS count : consumed   500 : 502
NEG count : acceptanceRatio    700 : 0.185185
Precalculation time: 0.931
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|        1|     0.09|
+----+---------+---------+
END>

===== TRAINING 2-stage =====
<BEGIN
POS count : consumed   500 : 502
NEG count : acceptanceRatio    700 : 0.0590369
Precalculation time: 0.932
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|    0.998| 0.121429|
+----+---------+---------+
END>

===== TRAINING 3-stage =====
```

```
<BEGIN
POS count : consumed   500 : 503
NEG count : acceptanceRatio    700 : 0.0226537
Precalculation time: 0.928
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|        1| 0.114286|
+----+---------+---------+
END>

===== TRAINING 4-stage =====
<BEGIN
POS count : consumed   500 : 503
NEG count : acceptanceRatio    700 : 0.00519211
Precalculation time: 0.932
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|    0.996| 0.138571|
+----+---------+---------+
END>

===== TRAINING 5-stage =====
<BEGIN
POS count : consumed   500 : 505
```

```
+----+---------+---------+
|    1|        1|        1|
+----+---------+---------+
|    2|    0.996| 0.185714|
+----+---------+---------+
END>
Stages 0-4 are loaded
```

After this training we will get cascade.xml file, which contains all traingin data in it.

We will use it futher work with openCV android.

5.6.Android Project

Android application uses opencv 2.4.5 version for android. For the base i took face dtection example, it is very simmilar to my application. At first time i use java code for camera detection, but it was too slow, and i managed to write native code which implements camera view. As it was expected, performance grew up very high it was approximetly 10-20 fps.

Cascade files, images, and audio notification of signs were placed into assets folder in project and loaded like this:

```
// load cascade file
String [] cascadeFiles = assetManager.list(CASCADE_DIR);
for(String s : cascadeFiles){
Log.i("Raw Asset: ", s);
InputStream is = assetManager.open(CASCADE_DIR+File.separator+s);
File cascadeDir = getDir(CASCADE_DIR, Context.MODE_PRIVATE);
File mCascadeFile = new File(cascadeDir, s);
FileOutputStream os = new FileOutputStream(mCascadeFile);
byte[] buffer = new byte[4096];
int bytesRead2;
while ((bytesRead2 = is.read(buffer)) != -1) {
os.write(buffer, 0, bytesRead2);
}
is.close();
os.close();
DetectionBasedTracker cc = new DetectionBasedTracker(mCascadeFile.getAbsolutePath(), 0);
cascades.put(cc, s);
cascadeDir.delete();
}
```

Detection phase was in native side for better perfornce of mobile phone. It written in on C++ code, using C++ library of opencv 2.4.5. Native C++ is located in jni folder.

```
JNIEXPORT void JNICALL Java_kz_balancy_leadmyway_DetectionBasedTracker_nativeDetect
(JNIEnv * jenv, jclass, jlong thiz, jlong imageGray, jlong faces){
LOGD("Java_kz_balancy_leadmyway_DetectionBasedTracker_nativeDetect enter");
try{
    vector<Rect> RectFaces;
    ((DetectionBasedTracker*)thiz)->process(*((Mat*)imageGray));
    ((DetectionBasedTracker*)thiz)->getObjects(RectFaces);
    vector_Rect_to_Mat(RectFaces, *((Mat*)faces));
  } catch(cv::Exception& e){
    LOGD("nativeCreateObject caught cv::Exception: %s", e.what());
    jclass je = jenv->FindClass("org/opencv/core/CvException");
    if(!je) {
      je = jenv->FindClass("java/lang/Exception");
    }
    jenv->ThrowNew(je, e.what());
  } catch (...) {
    LOGD("nativeDetect caught unknown exception");
    jclass je = jenv->FindClass("java/lang/Exception");
    jenv->ThrowNew(je, "Unknown exception in JNI code
{highgui::VideoCapture_n_1VideoCapture__()}");
  }
  LOGD("Java_kz_balancy_leadmyway_DetectionBasedTracker_nativeDetect exit");
}
```

5.7.REAL – TIME TESTING

After traning some signs samples. I make my application run. It was wonderfull how ti recognize the signs. I make linerar layout for left bar to show current recognized sign and some voice notification. Vioce notification is becaouse the driver cant always look to phone it rather be good if there will be some voice notificaitons. So her are some examples how it recognize the signs.

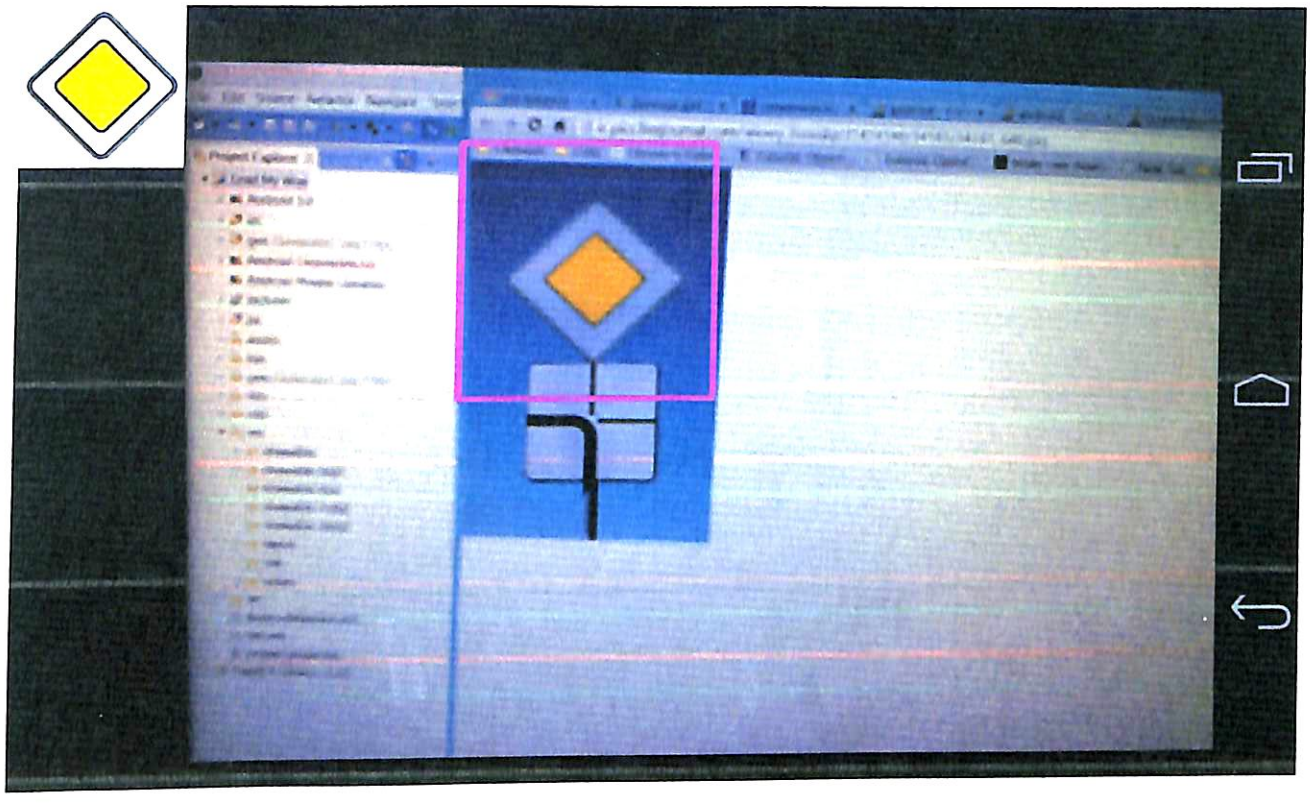

Figure 14 – main road recognized

Here my phone recognized main road sign. And draw some rectangular stroke in order to show what it recognize.

Figure 15 – Traffic sign

Here it recognizes the Give way. By the way, its not complete project. So there were some bugs while recognizing signs. Some time it just not recognize, some times it recognizse something that is not what i want.

The solution to this is more and more positive samples. This samples are need to traing a lot. The more positiive samples we have, the more presize it will recognize the sings.

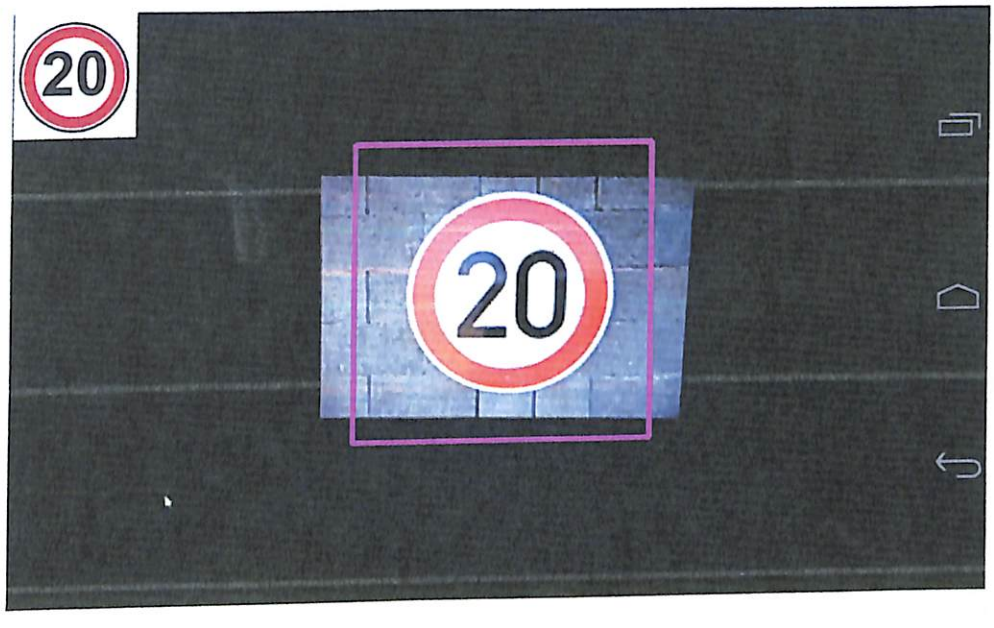

Figure 16 – 20km

CONCLUSION

Throughout thepast academicyear the developer has put his efforts into this application, which is not finished but has made some important steps forward. At this moment theproject is not in an ideal form but developers plan to keep improving it until the final presentation.

Thepurposeof theproject is toadd essential features that can increase safety for drivers, passengers and other members of thepublic. Therefore it would be beneficial to finishand keep improving theproject in future. So this project could beconsidered as apioneer in the trafficsign detection field. While designing and implementing this application, the developer has gained a valuableexperienceand knowledge in imageanalysis and imageprocessing. Theproject has covered many different fields likeAndroid application development, Hough transform implementation, different types of edge detection, pattern recognition and of course testing of theapplication.

So far, theapplication works well on thehand phone. It can detect standard trafficsign during day time. If mobilesareequipped with faster processor and larger amounts of memory, theapplication might work on thecar withadditional web cameraattached to the front of thecar. It is possible to implement theproject using other programming language for faster running, or useother moreefficient imageprocessing technique that uses less CPU resources and memory to make theproject work faster.

Also, it is planned to work on timecalculation and prediction of time that the next signal will light up. It is not useful to use fixed timing because thepattern of signals and timeof the signals can be different in different countries, cities and even streets, but wecould makeadaptiveor learning systems.

REFERENCES

1. ImageProcessing And Pattern Recognition Fundamentals and Techniques Frank Y. Shih, 2010.
2. О состоянии учета дорожно-транспортных происшествий за 4 месяца 2013 года -http://goo.gl/AgUAb, 17 may 2013.
3. Digital Picture Processing, Azriel Rosenfeld and Avinash K.,AcademicPress, 1982.
4. Robot Vision, Berthold Klaus Paul Horn,MITPress. 1986
5. DominantLocal Binary Patternsfor Texture Classification , LiaoS.,LawM.W.K. andChungA.C.S., 2009.
6. Dynamic Texture RecognitionUsingLocal Binary Patternswith an Applicationto Facial Expressions, Zhao, G. AndPietikäinenM.,IEEETrans. Pattern Analysis and MachineIntelligence, 2007.
7. Local Gabor Binary Pattern Histogram Sequence (LGBPHS), Zhang, W., Shan, S., Gao, W., Chen, X. And Zhang H.,A Novel Non-statistical Model for Face Representation and Recognition. In Proc. TenthIEEEInternational Conferenceon Computer Vision, 2005.
8. Lipreading with Local Spatiotemporal Descriptors, Zhao, G., Barnard, M. And Pietikäinen, M., IEEETrans. Multimedia, 2009.
9. Overview Of Image Processing, K.M.M. Rao,Deputy Director,NRSA,Hyderabad-500 037, 2006.
10. Image processing, http://en.wikipedia.org/wiki/Image_processing
11. Feature extraction, http://en.wikipedia.org/wiki/Feature_extraction
12. Local binary patterns, http://en.wikipedia.org/wiki/Local_binary_patterns
13. Tinku Acharya and Ajoy K. Ray (2006). Image Processing - Principles and Applications. Wiley InterScience.
14. Wilhelm Burger and Mark J. Burge (2007). Digital Image Processing: An Algorithmic Approach Using Java. Springer. ISBN 1-84628-379-5 and ISBN 3-540-30940-3 Check |isbn= value (help).

15. R. Fisher, K (2002). Springer. ISBN 3-540-67754-2. Missing or empty |title= (help)
16. Tim Morris (2004). Computer Vision and Image Processing. Palgrave Macmillan. ISBN 0-333-99451-5.
17. Tony F. Chan and Jackie (Jianhong) Shen (2005). Image Processing and Analysis - Variational, PDE, Wavelet, and Stochastic Methods. Society of Industrial and Applied Mathematics. ISBN 0-89871-589-X.
18. Milan Sonka, Vaclav Hlavac and Roger Boyle (1999). Image Processing, Analysis, and Machine Vision. PWS Publishing. ISBN 0-534-95393-X.
19. Chris Solomon and Toby Breckon (2010). Fundamentals of Digital Image Processing: A Practical Approach with Exp. in Matlab. Wiley-Blackwell. ISBN 978-0470844731.
20. DC. He and L. Wang (1990), "Texture Unit, Texture Spectrum, And Texture Analysis", Geoscience and Remote Sensing, IEEE Transactions on, vol. 28, pp. 509 - 512.
21. L. Wang and DC. He (1990), "Texture Classification Using Texture Spectrum", Pattern Recognition, Vol. 23, No. 8, pp. 905 - 910.
22. T. Ojala, M. Pietikäinen, and D. Harwood (1994), "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions", Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR 1994), vol. 1, pp. 582 - 585.
23. T. Ojala, M. Pietikäinen, and D. Harwood (1996), "A Comparative Study of Texture Measures with Classification Based on Feature Distributions", Pattern Recognition, vol. 29, pp. 51-59.
24. "An HOG-LBP Human Detector with Partial Occlusion Handling", Xiaoyu Wang, Tony X. Han, Shuicheng Yan, ICCV 2009
25. "The Android mascot was not always a little green guy". NBC News, Rosa Golijan.
26. "Original Android Logo Was A Truly Frightening Robot". Huffington Post, Betsy Isaacsen, January 4, 2013.

27.Blok, Irina. "Android Logo". Retrieved 5 April 2013.

28."Android Code Analysis". Retrieved June 6, 2012.

29."MIPS gets sweet with Honeycomb". Eetimes.com. Retrieved 2012-02-20.

30.a b Shah, Agam (December 1, 2011). "Google's Android 4.0 ported to x86 processors". Computerworld. International Data Group. Retrieved 2012-02-20.

31.a b "Android OS for i.MX Applications Processors Product Summary Page". freescale Inc.

32."Licenses". Android Open Source Project. Open Handset Alliance. Retrieved 2012-09-09.

33.Chris Welch (2013-04-16). "Before it took over smartphones, Android was originally destined for cameras". The Verge. Retrieved 2013-05-01.

34.Vance, Ashlee (27 July 2011). "Steve Perlman's Wireless Fix". Bloomberg Businessweek. Bloomberg. Retrieved 3 November 2012.

35.Block, Ryan (August 28, 2007). "Google is working on a mobile OS, and it's due out shortly". Engadget. Retrieved 2012-02-17.

36.Claburn, Thomas (September 19, 2007). "Google's Secret Patent Portfolio Predicts gPhone". InformationWeek. Retrieved 2012-02-17.[dead link]

37.Reinhard Klette, Karsten Schluens and Andreas Koschan (1998). Computer Vision – Three-Dimensional Data from Images. Springer, Singapore. ISBN 981-3083-71-9.

38.Bernd Jähne (2002). Digital Image Processing. Springer. ISBN 3-540-67754-2.

39.Gérard Medioni and Sing Bing Kang (2004). Emerging Topics in Computer Vision. Prentice Hall. ISBN 0-13-101366-1.

40.Nikos Paragios and Yunmei Chen and Olivier Faugeras (2005). Handbook of Mathematical Models in Computer Vision. Springer. ISBN 0-387-26371-3.

41.Wilhelm Burger and Mark J. Burge (2007). Digital Image Processing: An Algorithmic Approach Using Java. Springer. ISBN 1-84628-379-5.

42.Pedram Azad, Tilo Gockel, Rüdiger Dillmann (2008). Computer Vision – Principles and Practice. Elektor International Media BV. ISBN 0-905705-71-8.