ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS

Erhan TURAN

INDOOR INTELLIGENT VIDEO SURVEILLANCE SYSTEM APPLICATION

DEPARTMENT OF COMPUTER ENGINEERING

ADANA, 2014

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

INDOOR INTELLIGENT VIDEO SURVEILLANCE SYSTEM APPLICATION

Erhan TURAN

MSc THESIS

DEPARTMENT OF COMPUTER ENGINEERING

We certify that the thesis titled above was reviewed and approved for the award of degree of the Master of Science by the board of jury on 08/01/2014.

…….……..	…….……..	……..
Assoc. Prof. Dr. Mustafa GÖK	Asst. Prof. Dr. Murat AKSOY	Asst. Prof. Dr. Çiğdem İnan ACI
SUPERVISOR	MEMBER	MEMBER

This MSc Thesis is written at the Department of Institute of Natural And Applied Sciences of Çukurova University.
Registration Number:

Prof. Dr. Mustafa GÖK
Director
Institute of Natural and Applied Sciences

Not: The usage of the presented specific declarations, tables, figures, and photographs either in this thesis or in any other reference without citation is subject to "The law of Arts and Intellectual Products" number of 5846 of Turkish Republic.

ABSTRACT

MSc THESIS

INDOOR INTELLIGENT VIDEO SURVEILLANCE SYSTEM APPLICATION

Erhan TURAN

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER ENGINEERING

Supervisor : Assoc. Prof. Dr. Mustafa GÖK
Year: 2014, Pages: 61
Jury : Assoc. Prof. Dr. Mustafa GÖK
: Asst. Prof. Dr. Murat AKSOY
: Asst. Prof. Dr. Çiğdem İnan ACI

In this study, an application system of indoor intelligent distributed video surveillance designed and developed with present image processing algorithms and application programming interfaces.

This study aims to develop a supplementary video surveillance system on a pre-installed passive video surveillance system. The passive video surveillance systems record surveillance video feed with movement detection or plain full time interface. In this study, video feeds recorded by movement detection, will be analyzed for pedestrian and face extraction. After pedestrian and face data extracted from video feeds, pedestrian data will be grouped for every entering and leaving moments with matching process, then each image group of pedestrian's faces will be analyzed to find reappearance of the same pedestrian. After all image processing, system will predict each unique pedestrian's every entering and leaving time.

This study used OpenCV as primary API to manage grabbing, processing and post-processing of video and images. For application software architecture, Qt application programming framework utilized on thread and GUI programming. And for gathered information and statistical purposes SQLite database used. In this study, for human detection, Histogram of Oriented Gradients and face detection algorithms tested to find out optimum algorithm for human counting and recognition on videos of indoor video surveillance.

Key Words: Image processing, HOG, OpenCV, Face detection, Face recognition

ÖZ

YÜKSEK LİSANS TEZİ

BİNA İÇİ AKILLI GÖRÜNTÜ GÜVENLİK SİSTEMİ UYGULAMASI

Erhan TURAN

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Doç. Dr. Mustafa GÖK
Yıl: 2014, Sayfa:61
Jüri : Doç. Dr. Mustafa GÖK
: Yrd. Doç. Dr. Murat AKSOY
: Yrd. Doç. Dr. Çiğdem İnan ACI

Bu çalışmada, güncel görüntü işleme algoritmaları ve uygulama geliştirme arayüzleri ile bina içi akıllı görüntü güvenlik sistemi uygulaması tasarlandı ve geliştirildi.

Bu çalışmanın amacı hali hazırda çalışmakta olan pasif video güvenlik sistemlerine ek olarak çalışabilecek bir video güvenlik sistemi geliştirmektir. Pasif video güvenlik sistemleri, hareket bulma veya tam zamanlı kayıt yöntemleri ile video kayıtları almaktadır. Bu çalışmada, hareket bulma ile kaydedilen video verileri, yaya ve yüz bulma işleminden geçirilip elde edilen yaya görüntüleriyle her bir yayayı bulmak için eşleştirilme işlemi uygulanacaktır. Her bir yayanın grup halinde tanımlanan yüz görüntüleri, yüz tanıma işlemleri uygulanarak farklı yayaların tek bir grup altında tanımlanması sağlanacaktır. Bu şekilde aynı yayanın birden fazla giriş çıkış yaptığı tahmin edilerek, gerçek ve özgün yaya giriş çıkış bilgileri rapor edilebilecektir.

Bu çalışmada, video ve görüntü alma, işleme ve kaydetme için OpenCV kütüphanelerinden faydalanılmıştır. Uygulamanın yazılım mimarisi için, Qt uygulama programlama ortamı iplik ve GUI programlamada kullanılmıştır. Ve elde edilen verilerle istatiksel amaçlar için SQLite veritabanı kullanılmıştır. Bu çalışmada, kapalı mekan video güvenlik görüntüleri ile insan izleme ve sayma işlemlerine en uygun algoritmayı bulmak amacıyla HOG algoritması ile yüz bulma algoritmaları test edilmiştir.

Anahtar Kelimeler: Görüntü işleme, HOG, OpenCV, Yüz bulma, Yüz tanıma

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Assoc. Prof. Dr. Mustafa GÖK, for his supervision guidance, encouragements, patience, motivation, constructive and useful suggestions and his valuable time for this work. I am grateful to him for all the things that I have learned under his supervision. This thesis would not have been possible without the help of my advisor. He never deprived his help at any stage of my work.

I would like to thank members of MSc thesis jury, Asst. Prof. Dr. Murat AKSOY and Asst. Prof. Dr. Çiğdem İnan ACI for their constructive and lead-in suggestions and corrections.

And while gathering video surveillance data, thanks to my colleagues, İsmail YILDIZ, Erçin ÖZCAN, Ercan AVŞAR and Muhammed BARGHI for gladly taking part in the video sequences and tolerating the recording process with great kindness.

Finally, I would like to thank my family for their love, support and patience especially during the time I was working on this thesis.

CONTENTS **PAGE**

ABSTRACT I
ÖZ II
ACKNOWLEDGEMENTS III
CONTENTS IV
LIST OF FIGURES VI
LIST OF ABBREVIATONS VII
LIST OF ABBREVIATONS X
1. INTRODUCTION 1
2. PRELIMINARY WORK 5
2.1. Video Surveillance Systems 5
2.1.1. Video Surveillance System Components 6
2.1.1.1. Hardware 6
2.1.1.2. Software 7
2.2. Video and Image Processing 8
2.2.1. Background Subtraction Algorithms 8
2.2.2. Histogram of Oriented Gradients 9
2.2.3. FAST: Features from Accelerated Segment Test 11
2.2.4. BRIEF: Binary Robust Independent Elementary Features 14
2.2.5. ORB: An Efficient Alternative to SIFT or SURF 15
2.2.6. Haar-like Feature Based Cascade Classifier 18
2.2.7. Eigenfaces 19
3. MATERIAL AND METHOD 21
3.1. Material 21
3.1.1. Visual C++ 2010 C++ Compiler 21
3.1.2. Qt Framework 21
3.1.3. OpenCV 22
3.1.4. SQLite 22
3.1.5. Dataset 22
3.2. Method 25

3.2.1. Histogram of Oriented Gradients 25
3.2.2. FAST: Features from Accelerated Segment Test 25
3.2.3. ORB: An Efficient Alternative to SIFT or SURF 26
3.2.4. Haar-like Feature Based Cascade Classifier 26
3.2.5. Eigenfaces 27
4. PEDESTRIAN WATCHER SOFTWARE 29
4.1. The Hardware Implementation of Pedestrian Watcher 29
4.2. The Software Architecture of Pedestrian Watcher 29
4.2.1. Database Design of Pedestrian Watcher 33
4.3. Image Processing in Pedestrian Watcher 34
5. RESEARCH AND DISCUSSION 39
5.1. Pedestrian and Face Extraction Benchmarks 39
5.2. Pedestrian Matching Benchmarks 50
5.3. Pedestrian Recognition Benchmarks 51
6. CONCLUSION 55
REFERENCES 57
CURRICULUM VITAE 61

LIST OF TABLES **PAGE**

Table 3.1. Video dataset list of positioned indoor of the entrance 23
Table 3.2. Video dataset list of positioned outside of the entrance 24
Table 5.1. Extraction results using haarcascade_frontalface_default.xml for indoor video feeds 40
Table 5.2. Extraction results using haarcascade_frontalface_default.xml for outside video feeds 41
Table 5.3. Extraction results using haarcascade_frontalface_alt.xml for indoor video feeds 42
Table 5.4. Extraction results using haarcascade_frontalface_alt.xml for outside video feeds 43
Table 5.5. Extraction results using haarcascade_frontalface_alt2.xml for indoor video feeds 44
Table 5.6. Extraction results using haarcascade_frontalface_alt2.xml for outside video feeds 45
Table 5.7. Extraction results using haarcascade_frontalface_alt_tree.xml for indoor video feeds. 46
Table 5.8. Extraction results using haarcascade_frontalface_alt_tree.xml for outside video feeds 47
Table 5.9. Extraction results using lbpcascade_frontalface.xml for indoor video feeds 48
Table 5.10. Extraction results using lbpcascade_frontalface.xml for outside video feeds 49
Table 5.11. Grouping results after pedestrian matching process for each XML file 50
Table 5.12. Face Recognizer training results for indoor video feeds 51
Table 5.13. Face Recognizer predicts for outside video feeds 52

LIST OF FIGURES **PAGE**

Figure 2.1. Circle of sixteen pixels around candidate corner *p* (Rosten et al.,2006) 11
Figure 2.2. Generated vector samples with methods (Calonder et al., 2010) 15
Figure 2.3. From human retina to computer vision (Alahi et al., 2012) 18
Figure 4.1. Video enlisting in Pedestrian Watcher 30
Figure 4.2. Settings for pedestrian extraction in Pedestrian Watcher 31
Figure 4.3. Running pedestrian matching in Pedestrian Watcher 32
Figure 4.4. Pedestrian recognition and logging interface of Pedestrian Watcher 33
Figure 4.5. Pedestrian Watcher flow diagram 35

LIST OF ABBREVIATONS

API	: Application Programming Interface
CCD	: Charge Coupled Device
CCTV	: Closed Circuit Television
CMOS	: Complementary Metal Oxide Semiconductor
CPU	: Central Processing Unit
FPGA	: Field Programmable Gate Array
GHz	: Gigahertz
GUI	: Graphical User Interface
OS	: Operating System
PCA	: Principle Component Analysis
RAM	: Random Access Memory
SDK	: Software Development Kit
SoC	: System on Chip
SQL	: Structured Query Language
USB	: Universal Serial Bus

1. INTRODUCTION

Video surveillance systems running since 1940's, beginning with analogue hardware, to present day, with computer based digital hardware with sophisticated video and image processing algorithms and supplementary database and analysis software. Last decades shows that the hardware and software developments in computer science become feasible for designing and implementing effective video and image processing on video surveillance systems.

Switching from analogue hardware to computer based digital hardware systems was the milestone for video surveillance systems. This change, let two main difference from analogue systems: an extensive network system between hardware and high performance modifiable programming interface for any image processing task. Before these improvements, providing multiple surveillance images from distributed analogue cameras to people in charge was the primary purpose of video surveillance systems.

Increasing amount of security and traffic cameras and their counterpart, video matrix and monitors becomes more laboring for human officers to do their work(Desurmont et al, 2008), so this insufficiency urges surveillance technologies for automated analysis of video for movement detection, human and/or vehicle tracking and face detection with recognition.

OpenCV (http://www.opencv.org) is one of the most advanced application programming interfaces undertakes the demand on computer based image processing with robust and up-to-date algorithms. Besides the algorithms inside OpenCV, it also has an effective infrastructure, capable of appending new convenient algorithms and methods for computer vision and machine learning. And this let OpenCV the very thing for implementing and testing computer vision algorithms for real-time performance in video surveillance applications.

This thesis presents an indoor intelligent video surveillance system with the substantial image processing algorithms and methods. Before the image processing tasks, there are essential programming substructure to maintain image processing on a concurrent software system. Qt Framework is one of the fit, all in one API for the

software architecture design needed to implement a concurrent video surveillance system. Both OpenCV and Qt Framework is based on C++ programming language and OpenCV has a native GUI option based on Qt framework, although in this thesis OpenCV and Qt framework implemented with a heterogeneous approach.

Threading module of Qt Framework is vital interface part of the software system. GUI programming goes along with thread programming because of concurrency needs between heavy background image processing tasks and graphical user interface. Due to thread programming is based on operating system programming interfaces, which makes it particular to the operating system, hard to design operating system independent software that needs threading. Qt Threading class comes handy in this situation with a thin wrapper API over operating system's own threading interface. Withal it has a simple and effective inter-thread communication based on Qt Signals and Slots language construct.

Qt GUI module is a convenient interface for programming GUI for OpenCV based image processing. Even though OpenCV has its own module for user interface called HighGui, Qt GUI module has a comprehensive widgets and tools that can easily link to OpenCV code.

Additionally connecting to databases with Qt framework with threading and GUI support makes data processing reliable and untroubled. Database design and implementation brings intelligence to the video surveillance systems with data mining and information retrieval methods.

SQLite database engine used for the system to save collected data from image processing tasks. These records will be used to count and manage people in the building with a probabilistic approach.

For the computer vision tasks OpenCV used with multiple modules. OpenCV has varied specialized modules including matrix calculations, image processing, video processing, feature extraction, 3d calibration and machine learning, etc.

In this study, video and image processing modules used extensively for Histogram of Oriented Gradients algorithm (Dalal et al., 2005) to detect pedestrian in videos, Haar Feature-based Cascade Classifier algorithm (Viola et al., 2001 and Lienhart et al., 2002) for face detection on video frames, FAST (Rosten et al., 2010)

and ORB (Rublee et al., 2011) binary descriptor algorithm for feature comparison between founded pedestrians, Eigenfaces algorithm (Turk et al., 1991) for face recognition of founded pedestrian faces.

2. PRELIMINARY WORK

2.1. Video Surveillance Systems

Video-based surveillance system technological evolution begins with analogue CCTV systems. First CCTV systems used for military and governmental demands. These systems made from cameras located on desired points and connected to a centrically located surveillance monitors with video matrix. Today they are capable to gather video inputs from large amount of cameras with the choice of normal and infrared sensors, and digital or analogue connections. Video surveillance system technologies can be labeled in three generations (Valera et al., 2006).

First generation of video surveillance systems used analogue technologies for gathering, transferring, viewing and storing where analogue cameras transferring visual data with coaxial cables to the monitors and recording hardware. First generation technologies were working good in some situations but it has own problems like deficiencies of the analogue techniques. This let the researches canalize to digital techniques.

Second generation of video surveillance systems increase the surveillance efficiency by combining computer vision technology to automate primitive tasks of surveillance. While gathering and transferring of visual data improved by digital cameras and computer network communications, computer technology prospects the automation of all surveillance tasks with robust detection and tracking algorithms. This demand lead to third generation of video surveillance systems.

Third generation of video surveillance systems improved the accuracy of information processed by automated wide-area surveillance systems equipped with high definition normal and infrared camera sensors.

Advancements in hardware and software lead to new problems like distribution of information, design methodology, moving and multi-sensor platforms. And current research areas for video surveillance systems include data fusion, probabilistic reasoning framework and multi-camera surveillance techniques. With the increasing demand for security, applications of video surveillance systems

become substantial security implementations in the following areas (Valera et al., 2006):

- Transport Applications to process vehicle and human traffic in airports, maritime environments, railways, underground, motorways,
- Public places: stadiums, banks, supermarkets, malls, parking lots
- Sports: Football matches and others
- Industrial process for quality control purposes
- Military applications

2.1.1. Video Surveillance System Components

A video surveillance system's components can be classified under two substructure: hardware and software. Hardware is all the physical layer components used in the video surveillance system, from cameras, cables, capture cards to computers. There are various combinations of these equipments can be used to form the system and software substructure builds on according to these parts. Software substructure of the system is logical layer components of the video surveillance system consisted from techniques, algorithms and methods.

2.1.1.1. Hardware

Hardware components of the video surveillance systems can be defined under four main function: acquisition, transmission, processing and storage. CCD, CMOS and/or Infrared sensors can acquire images in various resolution and color mode for acquisition function. Acquisition requires image data formation to convert gathered data to image file or stream format. Acquired image can be transferred on analogue coaxial cables or on ethernet and fiber cables (and on wireless networks) digitally, based on the image data. After transferring image data, image can be processed by a FPGA or CPU based computing environment with software instructions. Tapes, hard disk drive and flash memory cards can be used to save all the processed and raw data

for storage functions. Intelligent network cameras may have all hardware components all in one place to distribute work load efficiently (Rinner et al., 2007). Inside these cameras there can be high resolution CCD/CMOS/Infrared sensors, FPGA or CPU based SoC and Gigabit ethernet port with custom image processing SDK for programming purposes.

2.1.1.2. Software

Software components of video surveillance systems can be defined under five main function: object detection, object recognition, video tracking, behavior and activities analysis, database and communication. Software components can be designed on CPU and FPGA based digital hardware. Object detection is the first step of video and image processing of video surveillance systems. Object detection can be applied to detect moving objects with background segmentation algorithms or detect preferred objects with object detection algorithms for chosen objects like faces (Lienhart et al., 2002), pedestrians (Dalal et al., 2005), etc. Object recognition is a video and image process for recognition of predefined objects from memory to extract knowledge for utilizing model-based techniques (Valera et al., 2006). Face recognition algorithms are most used object recognition algorithms in video surveillance systems to extract information for human movement analysis.

Video tracking is the video processing for locating a moving objects from a video stream or sequences of images. It is second step for intelligent video surveillance systems to extract movement data for behavioral analysis. Tracking can be applied with blob tracking, kernel-based tracking methods like Mean shift (Comaniciu et al., 1999) or filtering and data association methods like Kalman filter (Welsh et al., 1995) and Particle filter.

Behavior and activities analysis is the last step of visual data analysis of the video surveillance systems which interprets raw tracking and recognition data to high level pre-defined behaviors and actions.

Database and communication is the underlay frame software part of the video surveillance systems. Managing information traffic, storing and retrieving data

managed by database systems with database and network programming. A typical implementation of Makris et al. (2004) uses PosgreSQL for surveillance database with an intelligent camera network.

2.2. Video and Image Processing

Basic software parts of the video surveillance systems, defined as in section 2.1.1.2 has various combinations of methods and algorithms, video surveillance systems must arrange the optimum methods for the case will be implemented. In this thesis, an indoor scoped intelligent video surveillance system is designed and implemented on existing real-time passive video surveillance systems. To improve of video surveillance systems, usually non-discrete and merged technologies of applications practiced, thus reveals coherence problems like implementing face recognition and human detection algorithms to existing running video surveillance systems. There are several image processing algorithms evaluated in this thesis for these issues.

2.2.1. Background Segmentation Algorithms

Background segmentation is the first step of image processing for generic computer vision application. As a pre-processing task, it is vital to accomplish highly reliable results to move next image processing tasks. Background segmentation works by computing the difference with the present frame and the background model to sign the corresponding pixels as foreground or background. Background models vary by the algorithm itself. It can be simple previous frame difference or a model which has a changing memory of previous frames to improve the result. Every background segmentation algorithm, has its own advantages and disadvantages, some algorithms are simple for computation which makes them faster, some algorithms are more complicated and time consuming but with a better results according to simpler ones. Basic methods of background segmentation algorithms are the elementary algorithms based on frame differencing. They generally use

simple frame difference. Some of them has a simple memory based on mean or median calculations. Basic algorithms can be faster especially when calculating only with integer values. On the other hand, basic methods have poor results according to other algorithms with precision/recall comparison (Benezeth et al. 2010).

Statistical methods using one Gaussian or k Gaussian mixture model are based on probability density function of Gaussian distribution:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (2.1)$$

Stauffer and Grimson's method with Gaussian mixture model is most common algorithm based on GMM on background segmentation. This method creates a background model by modeling a mixture of Gaussian distribution for each pixel (Stauffer et al., 1999).

Eigenbackground method uses eigenspace to model the background which describes the range of appearances like lighting variations over the day, weather variations and etc(Oliver et al., 2000).

Fuzzy based methods of background segmentation algorithms use fuzzy integral computations with probabilistic approach. Choquet and Sugeno integrals can be used to aggregate different features of frame, like color space and texture similarity measures. (Baf et al. 2008a).

Early video surveillance systems used background segmentation algorithms to find movement in video feeds to trigger recording phase of the system, with this approach recorded data takes lesser space and removes the motionless video for the human and computer analysis. And lesser recorded data eases high hardware power demanding image processing algorithms work load.

2.2.2. Histograms of Oriented Gradients

Histogram of oriented gradients (HOG) descriptor method is based on local histograms of image gradient orientations evaluation after a well normalization

process. This method assumes local objects' appearance and shape can be found without knowing the precise corresponding gradient or edge positions of them on the image by the distribution of local intensity gradients or edge directions (Dalal et al., 2005).

This method applies six computational steps when evaluating an image for whether it does or not contains the object trained to find, first step of the method is to use a image-wide global gamma and color (if color channels exist) normalization to eliminate unwanted light conditions on the scene. This step is an optional step and varies to the image format and normalization methods.

After global normalization of image, image computed with 1-D or 2-D masks with an option using Gaussian smoothing, with this computations on image, gradients of the image will be usable for next step. As Dalal and his colleagues defined in their work (Dalal et al., 2005), simple 1-D [-1,0,1] masks with parameter $\sigma=0$ value turned out to be the best computation result for the work.

After gradient computations, spatial and orientation binning will be applied to the image. This step is the first step of local calculations will produce the cells. Cells are the local spatial regions that reserve the orientation bins accumulated by the weighted votes of the pixels. Weighted votes of pixels calculated with the orientation of the gradient element centered on it. Cells can be formed in rectangular (R-HOG) or radial (C-HOG) formation. For reducing aliasing between pixels, weighted vote values interpolated bilinear in orientation and position between the neighboring bins (Dalal et al., 2005).

After spatial and orientation binning step, normalization computations takes locally by grouping cells into larger blocks and normalizing the blocks one of the L1-norm(2.2), L1-sqrt(2.3), L2-norm(2.4) and L2-Hys (L2-norm followed by clipping with $max_v=0.2$ and renormalizing) block normalization schemes to ensure that low contrast regions are stretched (Dalal et al. 2005):

$$v \rightarrow v/(\|v\|_1 + \epsilon) \tag{2.2.}$$

$$v \rightarrow \sqrt{v/(\|v\|_1 + \epsilon)} \tag{2.3.}$$

$$v \rightarrow v/\sqrt{\|v\|_2^2 + \epsilon^2} \quad (2.4.)$$

Overlapping blocks while normalization increase the consistency of the image without the loss of local variations. After normalization step, with a designed detection window size (e.g. 64x128) scanning and collecting HOG descriptor blocks for the next step, the classifier.

A trained linear SVM used to classify the collected HOG descriptor blocks data in two class, does contains and does not. If the classified HOG descriptor blocks is contain the trained object it will give the boundaries of the object with its region. Classifier can be changed with a Gaussian kernel SVM which will increase the performance with the cost of more computation time.

2.2.3. FAST: Features from Accelerated Segment Test

FAST: Features from Accelerated Segment Test (Rosten et al., 2006) method is a corner detection algorithm, which proposed for a better computational performance than Harris detector (Harris et al., 1988) and SIFT detector (Lowe, 2004) methods.

This binary corner detection method decrease the complexity of corner detection to form a real-time operable corner detector algorithm with three main stages of computation.

In first phase, the segment test criterion operates pretest to confirm if the selected corner candidate pixel *p* can be considered as a corner by comparing the intensity difference of the circle of sixteen pixels with the I_P, pixel intensity of selected corner candidate pixel *p* using a threshold *t*.

First proposed version of this test applied and passed if one of the $I_n < I_P - t$ (darker intensity) and $I_n > I_P + t$ (lighter intensity) conditions is true for any subset of *n* contiguous pixels of sixteen pixels. A high speed test can be applied and passed if one of the $I_n < I_P - t$ (darker intensity) and $I_n > I_P + t$ (lighter intensity) conditions is true for at least three values of $n \in \{1,5,9,13\}$.

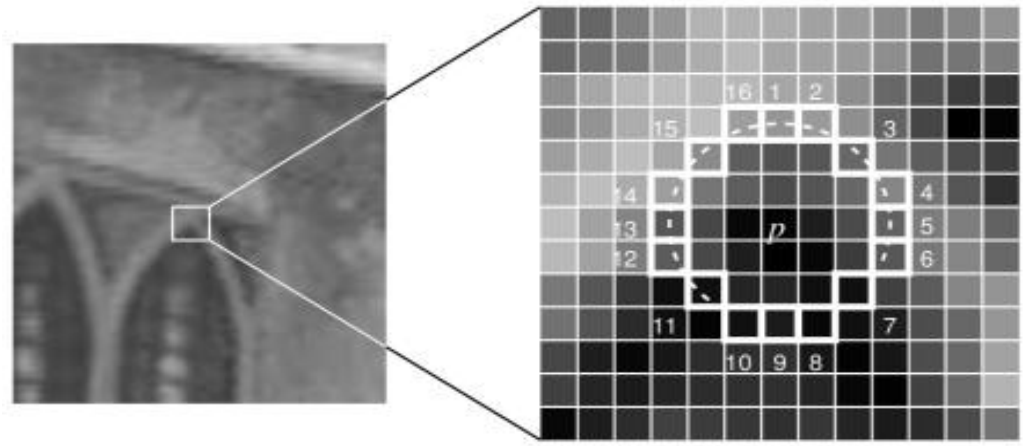


Figure 2.1. Circle of sixteen pixels around candidate corner *p* (Rosten et al., 2006)

This detector works with high performance in itself, but on the other side it has its own weaknesses(Rosten et al., 2006):

- For $n<12$ values the high-speed test can generalize poorly
- The choice and ordering of the fast test pixels contains implicit assumptions about the distribution of feature appearance.
- Test results are not reserved for the previous 4 pixels.
- Multiple features are detected adjacent to one another.

To overcome the first three issues mentioned above, algorithm uses a machine learning approach which has two stages. First, corners are detected from a set of images using the segment test criterion for *n* with a convenient threshold to build a corner detector for the given *n* with a slow algorithm which applied to all sixteen pixels of the circle.

For each pixel on the circle with corresponding location $x \in \{1..16\}$ can have one of three states(Rosten et al., 2006):

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \le I_p - t \quad (darker) \\ s, & I_p - t < I_{p\to x} < I_p + t \quad (similar) \\ b, & I_p + t \le I_{p\to x} \quad (brighter) \end{cases} \tag{2.5}$$

After computing every x values, depending on the results, feature vector of circle of sixteen pixels, P, subdivided into 3 subsets named P_d, P_s and P_b representing the darker, similar and brighter subsets of the vector P. In second stage of the method, first defines a Boolean variable, K_p, which will be true if p is a corner and false otherwise. After that, using ID3(Quinland, 1986) decision tree classifier algorithm query each subset using entropy of K_p to calculate if the candidate pixel is a corner. The entropy of K for the set P is(Rosten et al., 2006):

$$H(P) = (c + \bar{c})log_2(c + \bar{c}) - clog_2 c - \bar{c}log_2\bar{c} \tag{2.6}$$

where $c = \left|\{p|K_p\ is\ true\}\right|$ corners and $\bar{c} = \left|\{p|K_p\ is\ false\}\right|$ non-corners counts. This computations recursively applied to all the subsets until its entropy equals to 0 and the decision tree for fast detection for other images are ready to use.

Detecting multiple features adjacent to each other is the last weakness of the method. To overcome this problem, non maximal suppression method is used by computing a score function, V, by addition of the absolute difference between the pixels in the contiguous arc and the pixel p. Modified for performance version of the score function V (Rosten et al., 2006):

$$V = max\left(\sum_{x\,\in\, S_{bright}}\left|I_{p\to x} - I_p\right| - t, \sum_{x\in S_{dark}}\left|I_p - I_{p\to x}\right| - t\right) \tag{2.7}$$

with:

$$S_{bright} = \{x|I_{p\to x} \geq I_p + t\} \tag{2.8}$$

$$S_{dark} = \{x|I_{p\to x} \leq I_p - t\} \tag{2.9}$$

2.2.4. BRIEF: Binary Robust Independent Elementary Features

BRIEF is a binary descriptor algorithm proposed by Calonder and his colleagues (Calonder et al., 2010) for the growing need of faster computing and matching with a memory efficient binary descriptor method.

BRIEF method applies a test with image on randomly chosen patch p with size $S \times S$ as (Calonder et al., 2010):

$$\tau(p; x, y) := \begin{cases} 1 & if\ p(x) < p(y) \\ 0 & otherwise \end{cases} \tag{2.10}$$

where $p(x)$ is the pixel intensity of p at $x=(u,v)^T$. Before the test applies it needs a pre-smoothing for the patch selected on the image to overcome sensitivity to noise in the patch. A Gaussian filter applied for smoothing operation. And with this binary test BRIEF descriptors created with a set of n_d -dimensional bit string (Calonder et al., 2010):

$$f_{n_d}(p) := \sum_{1 \le i \le n_d} 2^{i-1} \tau(p; x_i, y_i) \tag{2.11}$$

where n_d = 128, 256 and 512 for the bit string length. For the spatial arrangement of the binary test there can be five methods to determine x_i and y_i values (Calonder et al., 2010):

- The (x_i, y_i) locations are chosen on the patch p uniformly and values can be closer to the patch border (Fig. 2.2 - G I).
- The (x_i, y_i) locations are randomly chosen with a Gaussian distribution, this gives more centric locations (Fig. 2.2 - G II).
- The (x_i, y_i) locations are generated with two steps: first location x_i is chosen with a Gaussian distribution around the origin of patch p and location y_i is chosen with another Gaussian distribution around the location x_i. This gives more local vectors (Fig. 2.2 - G III).

- The (x_i, y_i) locations are randomly generated from discrete location of a coarse polar grid (Fig. 2.2 - G IV).
- For each i, x_i=(0,0), origin of the patch p, and y_i takes all values on a coarse polar grid containing n_d points (Fig. 2.2 - G V).

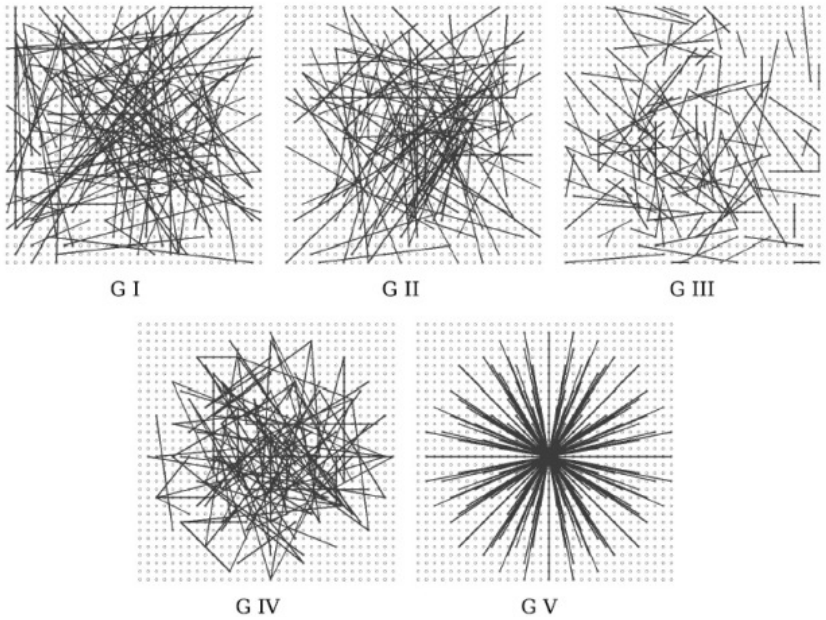


Figure 2.2. Generated vector samples with five methods (Calonder et al., 2010)

For binary descriptors measurement, BRIEF uses Hamming distance, which compare bit string with a XOR and add all values of the result.

2.2.5. ORB: Oriented FAST and Rotated BRIEF

Oriented FAST and Rotated BRIEF (ORB) method is a fast binary descriptor based on BRIEF (Calonder et al., 2010) and FAST (Rosten et al., 2006) with more rotation invariant, noise resistant and higher computing performance according to BRIEF and FAST method.

This method has improvements on:

- Faster and more accurate orientation components than the FAST method.
- More efficient computations for oriented BRIEF features.

- Analysis for variance and correlation of oriented BRIEF features.
- A learning method for de-correlating BRIEF features under rotational invariance (improves the nearest-neighbor applications' performance)

ORB method starts with detecting FAST points using a circular radius of 9 (FAST-9) and because FAST method does not order points with a cornerness measure, ORB employ a Harris corner measure (Harris et al., 1988) to order number N key points by lowering threshold to get more than N key points and pick the top N key points. And lack of orientation components in FAST features, ORB implements a simple but effective corner orientation method called intensity centroid (Rosin, 1999). Intensity centroid method assumes that a corner's intensity equals to the offset from its center point and this vector can be used to refer a orientation with defining the moments of a patch as (Rosin, 1999):

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{2.12}$$

and with moments to find the centroid (Rosin, 1999):

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \tag{2.13}$$

After that constructing a vector from corner's center, *O*, to the centroid, $\overrightarrow{OC}$ can be done and the orientation of the patch defined as (Rublee et al., 2011):

$$\theta = atan2(m_{01}, m_{10}) \tag{2.14}$$

where atan2 is the quadrant-aware version of arc tan, used to find the direction of the vector.

ORB method use BRIEF descriptors, but because of BRIEF descriptors poor performance with orientation, ORB steer BRIEF descriptors according to the orientation of key points by defining the $2 \times n$ matrix S (Rublee et al., 2011):

$$S = \begin{pmatrix} x_1, \dots, x_n \\ y_1, \dots, y_n \end{pmatrix} \tag{2.15}$$

which contains the coordinates of the pixels, then using the orientation of the patch, θ, finds its rotation matrix and rotates the S to get steered version S_θ.

$$S_\theta = R_\theta S \tag{2.16}$$

And the steered BRIEF operator becomes (Rublee et al., 2011):

$$g_n(p, \theta) \coloneqq f_n(p) | (x_i, y_i) \epsilon S_\theta \tag{2.17}$$

Steered BRIEF has considerably lower variance value, to recover low variance, ORB has a learning method for choosing a good subset of binary tests. For settings of learning algorithm, training set of 300k key points drawn from PASCAL 2006 dataset (Everingham, 2006). And the algorithm applied as(Rublee et al., 2011):

1) Run each test against all training patches.
2) Form the vector T by ordering the tests according to their distance with a mean of 0.5.
3) Apply a greedy search

    a) Put the first test to the result vector R and remove from vector T.
    b) Get the next test from vector T and compare it with all tests in vector R. If its absolute correlation is greater than a threshold, discard it; else add it to vector R.
    c) Repeat the previous step until there are 256 tests in vector R. If all tests gathered from vector T and vector R has less tests than 256, raise the threshold value and repeat previous step again.

After this algorithm completed, there are 256 relatively uncorrelated tests with high variance.

2.2.6. Haar Feature-based Cascade Classifier

Haar feature-based cascade classifier is a object detector method which is proposed by Viola et al. (2001) and improved by Lienhart et al. (2002). This method can be trained for a particular object. In this thesis, it is used for face detection already implemented on OpenCV API.

Method has two phases, in first phase method must be trained for the particular object. Classifier uses Haar-like features(Figure 2.5) with positive examples of the desired object to detect and negative examples with arbitrary images all in same size.

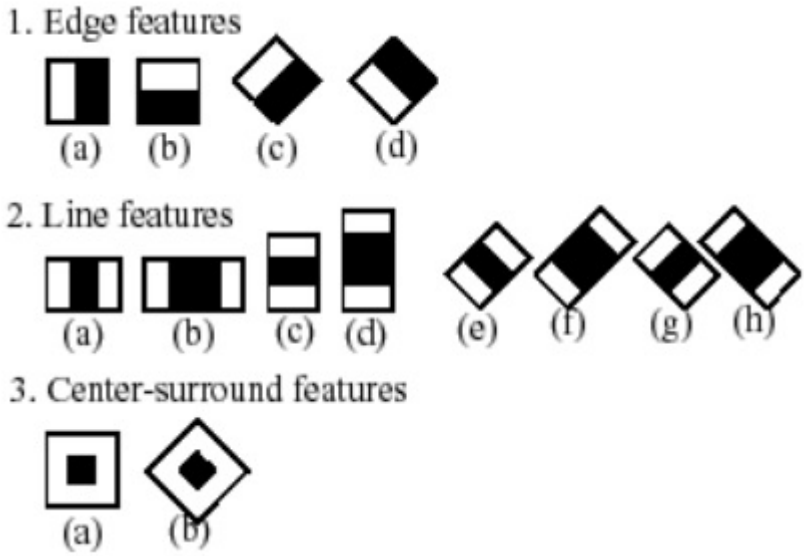


Figure 2.3 Haar-like features. (Lienhart et al., 2002)

After training classifier, method can be used in an image or region of interest of an image. When classifier window is searching for a matching pattern, 1 returns for a positive pattern and 0 for negative one. Classifier can be used on different sizes to match the correct size of the object on the image.

2.2.7. Eigenfaces

Eigenfaces method used to identify faces based on Principal Component Analysis which proposed independently by Pearson(1901) and Hotelling(1933). To reduce the information represented by a 100x100 image can be overwhelming to figure out the corresponding image is the one that is look for a particular pattern. In PCA, this high-dimensional dataset can be reduced to a unique less-dimensional information for a comparison.

Let $X = \{ x_1, x_2, \ldots, x_n \}$ be a random vector with observation $x_i \in R^d$. Compute the mean μ and covariance matrix S:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T \tag{2.18}$$

Compute the eigenvalues λ_i and eigenvectors v_i of S and order the eigenvectors descending by their eigenvalue. And form k principal components from the eigenvectors which has the k largest eigenvalues. And the k principal components of the vector x are as follows:

$$y = W^T(x - \mu) \quad where\ W = (v_1, v_2, \ldots, v_k) \tag{2.19}$$

And the reconstruction from the PCA basis as follows:

$$x = Wy + \mu \quad where\ W = (v_1, v_2, \ldots, v_k) \tag{2.20}$$

And eigenfaces method can perform face recognition by projecting all training samples and the query image into the PCA dataset and find the nearest neighbors of the projected query image in the training samples.

3. MATERIAL AND METHOD

3.1. Material

In this thesis, to create an intelligent video surveillance system, from coding to compiling, Qt Creator with Visual C++ 2010 C++ Compiler used to compile applications; for image processing OpenCV image and machine learning library used and to raise the image processing part to a running practical program Qt Framework used especially with its concurrency and GUI modules. For database needs in this thesis, SQLite module of Qt is used to create databases of the system.

3.1.1. Visual C++ 2010 C++ Compiler

Visual C++ 2010 C++ compiler is the commercial C++ compiler of the Microsoft Corporation. Compiler Collection can be acquired with free and paid versions with different support and equipment included. Beside the lack of operating systems variety it has x86 and x86-64 compiling capabilities for Intel CPU hardware. In this thesis, to compile the test programs and the Pedestrian Watcher application Visual C++ 2010 C++ Compiler used with debug and release options with 32 bit application profile.

3.1.2. Qt Framework

Qt Framework (http://qt-project.org/) is an open-source, cross-platform application and UI framework for developers using C++ programming language. It has supports for Windows, Linux and MacOS desktop operating systems, embedded, mobile and real-time operating systems. Framework has several modules coherently works together. In this thesis, Core, Threading, Database and GUI modules used to design a distributed intelligent video surveillance system. For concurrency needs between the OpenCV image and machine learning library Thread module used extensively.

3.1.3. OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library built to provide a common infrastructure for computer vision applications. OpenCV has more than 2500 optimized algorithms, for computer vision and machine learning, and in this thesis, used by all the image processing tasks directly and with OpenCV based libraries. For image acquisition, OpenCV used to extract images from video files and used to read images from disk. In this thesis, HOG algorithm, implemented in OpenCV, used to evaluate the pedestrian detection performance. And for image comparison, FAST, ORB and FREAK algorithms implemented in OpenCV used. And for face detection Haar feature-based cascade classifier and Eigenfaces implemented in OpenCV used.

3.1.4. SQLite

SQLite (http://www.sqlite.org//) is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed SQL database engine in the world. The source code for SQLite is in the public domain. It is the most popular mobile database API because of the long time stable and small footprint at runtime, in open source and commercial communities. In this thesis, SQLite database implementation of Qt Framework used to manage the aggregated data for every step of the image analysis. Records of face and pedestrian coordinates saved with SQLite API.

3.1.5. Dataset

In this thesis, for the image processing testing and evaluations there are 52 video files made with 5 different people from two different point of view in 1920*1088 frame resolution and 20 fps frame rate. All of the people have at least two entering and leaving videos for both views. scenes with 3 image resolution sets.

Table 3.1. Video dataset list of positioned indoor of the entrance.

Video Name	Pedestrian ID	Frames	Action	Position
inP1C1	1	116	Entering	Inside
inP1C2	1	120	Entering	Inside
inP1C3	1	113	Entering	Inside
inP1G1	1	178	Leaving	Inside
inP1G2	1	150	Leaving	Inside
inP1G3	1	166	Leaving	Inside
inP2C1	2	122	Entering	Inside
inP2C2	2	151	Entering	Inside
inP2G1	2	168	Leaving	Inside
inP2G2	2	147	Leaving	Inside
inP3C1	3	135	Entering	Inside
inP3C2	3	123	Entering	Inside
inP3C3	3	164	Entering	Inside
inP3G1	3	169	Leaving	Inside
inP3G2	3	167	Leaving	Inside
inP3G3	3	157	Leaving	Inside
inP4C1	4	144	Entering	Inside
inP4C2	4	133	Entering	Inside
inP4C3	4	115	Entering	Inside
inP4G1	4	160	Leaving	Inside
inP4G2	4	156	Leaving	Inside
inP4G3	4	166	Leaving	Inside
inP5C1	5	114	Entering	Inside
inP5C2	5	124	Entering	Inside
inP5G1	5	154	Leaving	Inside
inP5G2	5	174	Leaving	Inside

Table 3.1. shows the list of the video files recorded inside the entrance of the test scene. And Table 3.2. shows the list of the video files recorded outside the entrance of the test scene. These videos recorded at night time without the sunlight conditions with lighting of fluorescent lamps implemented homogeneous on the building ceiling. This gives a clear view of pedestrians and their faces without the lateral shades. Video files recorded aims to mimic a passive video surveillance system that records surveillance video with a motion triggering method.

Table 3.2. Video dataset list of positioned outside of the entrance.

Video Name	Pedestrian ID	Frames	Action	Position
outP1C1	1	156	Entering	Outside
outP1C2	1	107	Entering	Outside
outP1C3	1	137	Entering	Outside
outP1G1	1	164	Leaving	Outside
outP1G2	1	160	Leaving	Outside
outP1G3	1	148	Leaving	Outside
outP2C1	2	134	Entering	Outside
outP2C2	2	163	Entering	Outside
outP2G1	2	172	Leaving	Outside
outP2G2	2	163	Leaving	Outside
outP3C1	3	156	Entering	Outside
outP3C2	3	117	Entering	Outside
outP3C3	3	110	Entering	Outside
outP3G1	3	178	Leaving	Outside
outP3G2	3	166	Leaving	Outside
outP3G3	3	160	Leaving	Outside
outP4C1	4	135	Entering	Outside
outP4C2	4	118	Entering	Outside
outP4C3	4	112	Entering	Outside
outP4G1	4	136	Leaving	Outside
outP4G2	4	161	Leaving	Outside
outP4G3	4	151	Leaving	Outside
outP5C1	5	116	Entering	Outside
outP5C2	5	133	Entering	Outside
outP5G1	5	162	Leaving	Outside
outP5G2	5	201	Leaving	Outside

3.2. Method

The purpose of this study is to design and develop an intelligent video surveillance system over a traditional video surveillance system to count the pedestrians and tag them to extract the info for every individual's entering and leaving time. For this purpose, first HOG pedestrian detector algorithm evaluated for pedestrian detection, for pedestrian image comparing FAST and ORB feature extraction methods used and for face detection and recognition Haar Cascade Classifier and Eigenfaces used.

3.2.1. Histograms of Oriented Gradients

Histogram of oriented gradients (HOG) descriptor method is based on local histograms of image gradient orientations evaluation after a well normalization process. This method assumes local objects' appearance and shape can be found without knowing the precise corresponding gradient or edge positions of them on the image by the distribution of local intensity gradients or edge directions. OpenCV library has an implemented HOG algorithm to use on its own image format. Algorithm's function has six input and one input/output parameters. First input is the image data to searching for pedestrians, second input/output is the vector list of the detected pedestrians, second input is the threshold for the distance between features and SVM classifying plane, fourth input is the window stride, fifth input is a parameter for CPU interface compatibility, sixth input is the coefficient of the detection windows increase and the seventh parameter is the coefficient to regulate the similarity threshold. In this study default parameter values used for third (0), fourth(8x8), fifth (32x32), sixth (1.05) and seventh (2) parameters.

3.2.2. FAST Features from Accelerated Segment Test

FAST: Features from Accelerated Segment Test is a corner detector method which implements faster algorithm than SIFT and Harris corner detectors feasible for

real-time operations. OpenCV library has an implemented FAST algorithm to use on its own image format. Algorithm's function has three input and one input/output parameters. First input is the image data to searching for corners, second input/output parameter is the vector list of key points, third input is the threshold value for FAST algorithm and the last input parameter is the Boolean value for option to use non-maximal suppression method. In this study, FAST used with 10 for threshold value and true for non-maximal suppression method.

3.2.3. ORB Oriented FAST and Rotated BRIEF

Oriented FAST and Rotated BRIEF (ORB) method is a fast binary descriptor based on BRIEF with more rotation invariant, noise resistant and higher computing performance according to BRIEF and FAST method. OpenCV library has an implemented ORB algorithm to use on its own image format and vector of key points to find binary descriptors. Algorithm's function has two input and one input/output parameters. First input is the image data to searching for corners, second input parameter is the vector list of key points and third input/output is the descriptors for the results.

3.2.4. Haar Feature-based Cascade Classifier

Haar feature-based cascade classifier is a object detector method which can be trained for a particular object. OpenCV library has trained, ready to use face and eye detector configurations. In this study, method used with the five different pre-trained front face configuration files:

- haarcascade_frontalface_alt.xml
- haarcascade_frontalface_alt_tree.xml
- haarcascade_frontalface_alt2.xml
- haarcascade_frontalface_default.xml
- lbpcascade_frontalface.xml

and scale factor float parameter with value 1.1, minimum neighbors integer parameter with value 3 and minimum size of search frame with values 30x30 pixels. For faster processing, image can be resized to a smaller image, but this could reduce the face detection performance.

3.2.5. Eigenfaces

Eigenfaces method used to identify faces based on Principal Component Analysis by reduce the dimension of the information ease to find particular pattern for comparison of two datasets like 2 dimensional images. Code of method implemented in OpenCV library with FaceRecognizer class.

4. PEDESTRIAN WATCHER SOFTWARE

This study aims to design and develop an intelligent video surveillance system software, supplementary to a passive video surveillance system. Passive video surveillance systems record all video feeds with movement detection triggering or plain full time recording. Pedestrian Watcher software proposed for a supplemental system to passive video surveillance systems by analyzing pre-recorded video feeds for pedestrians to count and tag them as a gateway keeper.

4.1. The Hardware Implementation of Pedestrian Watcher

Pedestrian Watcher needs special hardware implementation to record pedestrian's face and whole body clearly for reliable pedestrian analyze. Implementing to desired entrance, there must be two camera system placed opposing views to get coming and going pedestrians clearly at the same time. Cameras must be located between 170 cm and 200 cm height from the ground to have a parallel view of the pedestrian faces and for face detection at least 60x60 pixels of clear view, a high definition of 720p or 1080p recording resolution. And cameras recording frame per second performance must be at least 20 frames per second to compare sequential frames easily to find the pedestrian image groups.

For this study a simple implementation of camera system consisting two 1080p high definition cameras located on both sides of an entrance doorway with a 170 centimeters height from the ground and 3 meters far away from the doorway. And presumes that the camera system recording with a motion triggering method to record passing pedestrians. Video recording performed with a 20 frame per second frame rate.

4.2. The Software Architecture of Pedestrian Watcher

Pedestrian Watcher is a graphical user interface based application for pedestrian detection and counting for general purpose building surveillance system, it

acts as a supplementary system for existing passive video surveillance systems. The passive video surveillance system records with motion triggering method to storage media and Pedestrian Watcher loads these video feeds and analyzes the video feeds with three main steps: extracting pedestrians and their faces, grouping founded pedestrians and training face recognizer dataset from these groups of face images. After these steps accomplished Pedestrian Watcher tries to predict the pedestrians coming and leaving moments and count them.

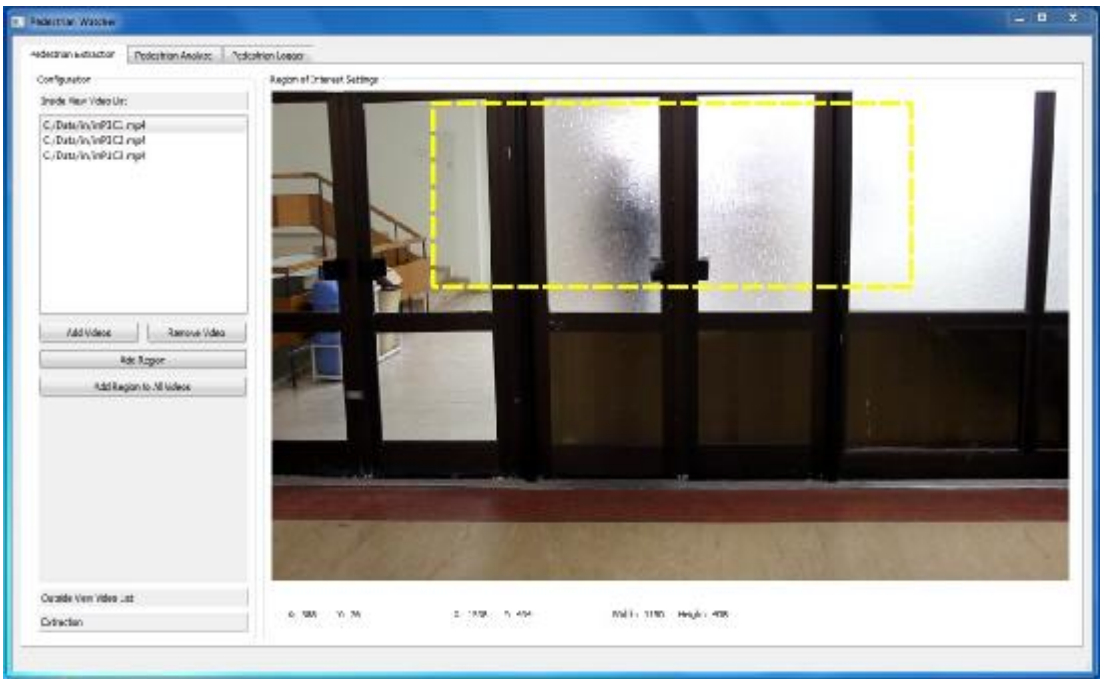


Figure 4.1. Video enlisting in Pedestrian Watcher

The first step of Pedestrian Watcher is to extract pedestrians and their faces from enlisted inside and outside video feeds. With this extraction process Pedestrian Watcher will have pedestrian and face data to analyze. As seen in the Figure 4.2 Pedestrian Watcher interface is fairly simple for video feed enlisting.

User can add videos with Add Video button and then define processing area for extraction of the video files on the image by pressing mouse with moving around and submitting with Add Region for the chosen video feed or with Add Region to All Videos for whole inside or outside video feeds, if no processing area is defined with interacting on image, default processing area will be the whole size of the video

feed's frame width and height. This will extend the processing time particularly on high definition video feeds with 1080p resolution.

After choosing the video feed files and setting their processing areas for both inside and outside video feeds, user can define the method to extract the pedestrians on the video feeds. HOG pedestrian detector and pseudo pedestrian detection based on face detector can be chosen for the method.

After choosing the extraction method, user must choose a predefined trained face xml file for face detection. Then user must set the database and data folders location with Set Database button. After these settings user can save the settings with Save Settings button and start extraction process with submitting the Start Extraction button.



Figure 4.2. Settings for pedestrian extraction in Pedestrian Watcher.

When extraction process finished it will inform the user that processing is done. User now can proceed to the next step for pedestrian grouping.

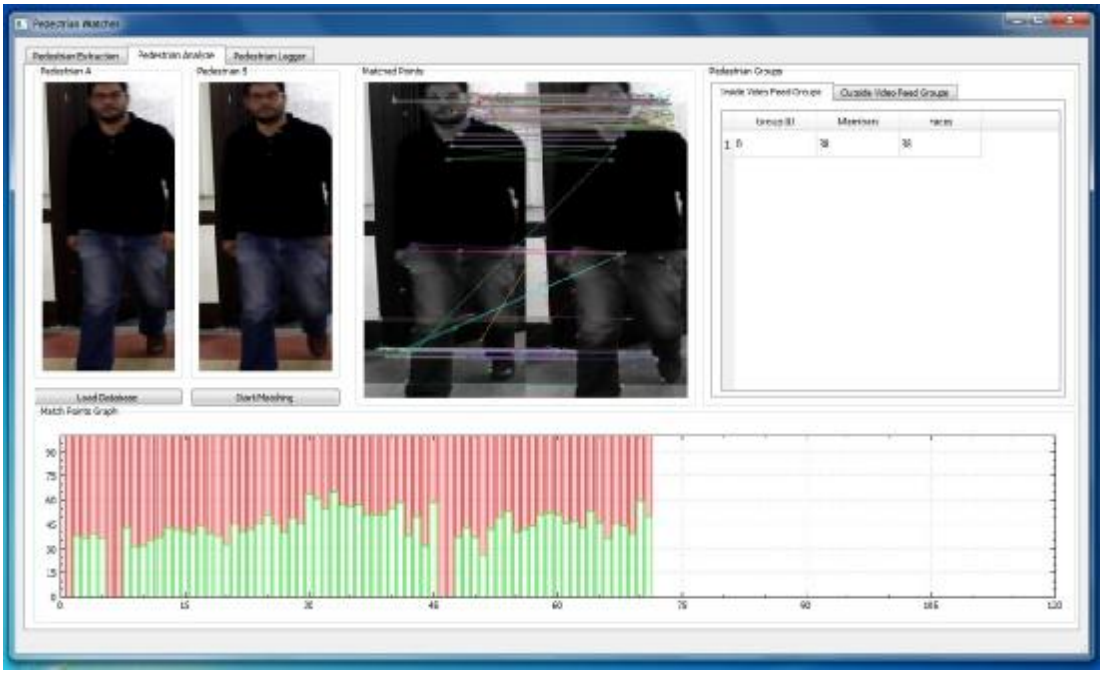


Figure 4.3. Running pedestrian matching in Pedestrian Watcher.

The second step of the Pedestrian Watcher is to perform a comparison between the consecutive pedestrian images to find every individual in the extracted pedestrian images. To start comparison, user must load the defined and saved database in the first step of the Pedestrian Watcher with Load Database button. Then user can start the matching process by submitting Start Matching button. When process started, it will show the matching pair of images and comparison result on Pedestrian A, Pedestrian B and Matched Points images. If matching of two images does not produce a result which means at least one of the images has no detected key points for matching will be passed without a Matched Points image shown.

Every result of matching pairs will be projected to the Match Point Graph with after calculation of good matching points to all matching points ratio normalized to 100 percentage. A good matching result can be considered with a 25% value and above it. While process is running it will find the pedestrian image groups and post it to the Inside Video Feed Groups and Outside Video Feed Groups tables with group ID, amount of group images and faces.

When processing finished, user can pass to the final step to pedestrian recognition based on face recognizing process.

The last step of Pedestrian Watcher is to train a face recognizer model and evaluate inside and outside video feeds with trained face recognizer.

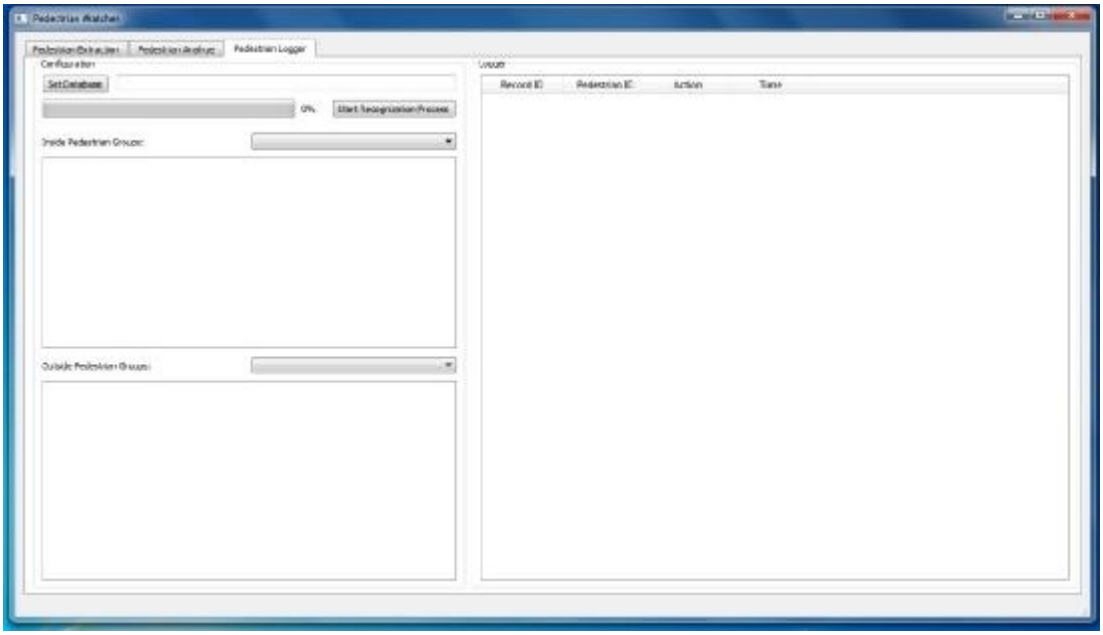


Figure 4.4. Pedestrian recognition and logging interface of Pedestrian Watcher.

Face recognizer training and classifying process configuration is straightforward, user must choose the database which processed with the second step with Set Database button. After choosing the desired database, user can start process by submitting Start Recognition Process button.

After processing is done, Pedestrian Watcher will dispatch the results on Logger and enlist the inside and outside video feeds' groups with their entering and leaving times.

4.2.1. Database Design of Pedestrian Watcher

Pedestrian Watcher records all extracted information to a SQLite database named *main.db* for all three steps of the software. First step creates three tables for each inside and outside video feeds: *vidin, vidout, pedin, pedout, facein, faceout*. *vidin* and *vidout* tables contains the video feed files names and corresponding code for that file names. This will simplify the tracking of the pedestrian and face images

origin of video. *pedin* and *pedout* tables contains the pedestrian name based on the time appeared on the video, pedestrian frame's first pixel x and y values, pedestrian frame's width and height and video feed file code. *facein* and *faceout* tables contain, the face name based on the time appeared on the video, which is same name of the pedestrian of that face, face frame's first pixel x and y values, face frame's width and height, video feed file code and group ID of that face. And in the first step, while program extracting pedestrians and their faces, it will insert the pedestrian and face record to the database.

In the second step, while program matching pedestrians it will request the pedestrians' data from the database and if a new group detected by matching process, it will update the corresponding face images' group ID value.

In the third step, while program training face recognizer, it will request the face groups with group ID larger than -1 and feed the face recognizer with the face images and their corresponding group ID value.

4.3. Image Processing in Pedestrian Watcher

Image processing in Pedestrian Watcher implemented with Histogram of Oriented Gradients pedestrian detector, Haar feature-based cascade classifier for face detection and pseudo pedestrian detection, corner detection algorithm FAST and binary descriptor algorithm ORB for matching pedestrians and Eigenfaces face recognition algorithm for pedestrian recognition.

HOG pedestrian detector is a suitable pedestrian detector for most of the condition, however HOG pedestrian detector performed poorly with some of the video feed samples for this study. Because of this unexpected situation, in this study, a simple but effective way used for pedestrian detection based on face detection, this pseudo pedestrian detection applies by after detecting a face on the frame, assumes the human body ratio can be used to calculate the pedestrian body by learning detected face size with multiplying face height with 8 and face width with 3 after that subtracting face width from face first pixel's x coordinate to set the face in the middle of the pedestrian frame. If the calculated pedestrian frame overflow on whole image,

it will be trimmed to fit in the image. This way performs with a sufficient viewport to apply pedestrian matching for finding consecutive pedestrian images.

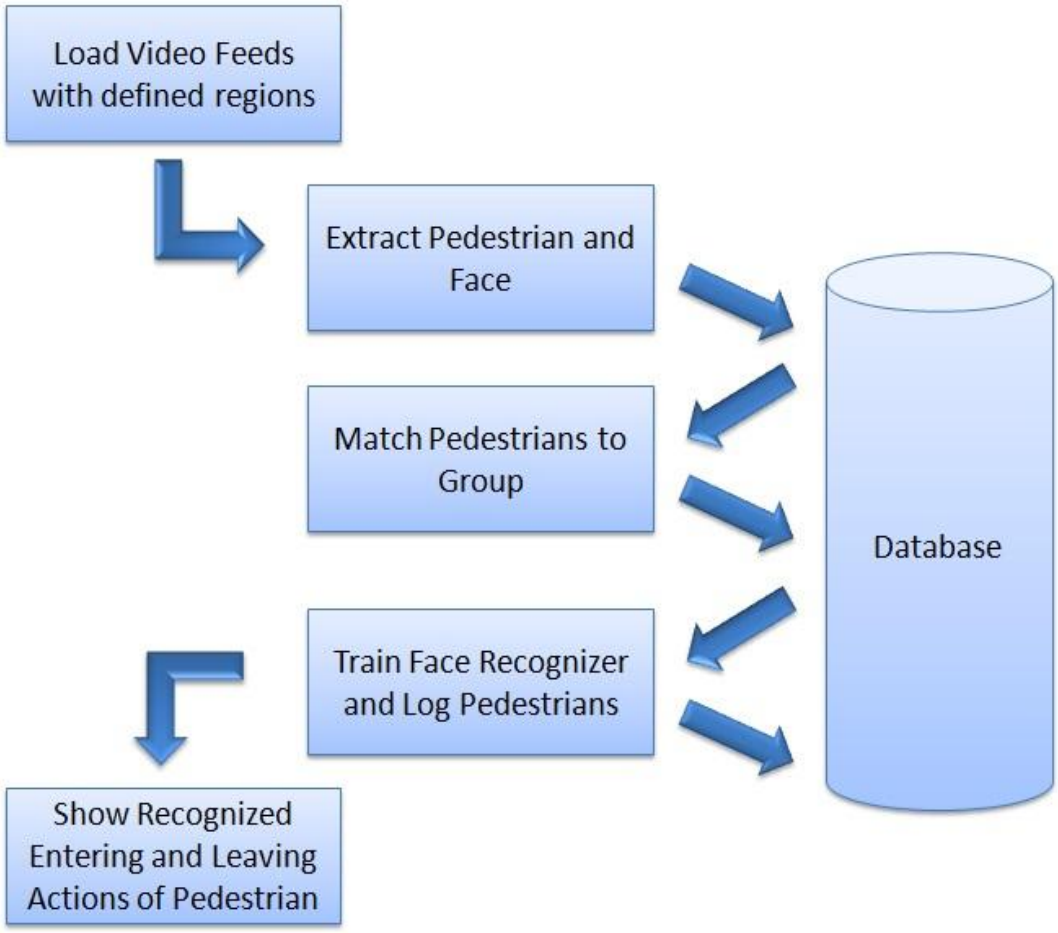


Figure 4.5. Pedestrian Watcher flow diagram.

First image processing phase, applies face detection for all video feeds' frames, when a face detected, Pedestrian Watcher calculates the pedestrian body with the method described and then both face and pedestrian images will be saved on storage media and recorded to the database. This processing phase is the most time consuming because evaluating face detections on high resolution images.

Second image processing phase applies pedestrian matching with FAST corner detection aided ORB binary descriptor method. Process begins with gathering pedestrian images in order for each inside and outside dataset. Gathered two images will be processed with FAST method in OpenCV and then results will computed

with the ORB method and after produced binary descriptors for two consecutive image in the dataset, images will be matched with brute force matcher class of OpenCV, BFMatcher.

After matching process, matched results will be trimmed with a threshold value calculated by:

$$threshold = \frac{(maxDist - minDist) * TH}{100} \tag{4.1}$$

where ***maxDist*** is the maximum distance between the matching two featured point of the pedestrian images, ***minDist*** is the minimum distance between the matching two featured point of the pedestrian images and ***TH*** is the percentage of normalization which can be adjusted to find optimum threshold value, in this study, ***TH*** value given 40. This threshold will eliminate most of the poor matched points in the matched point list. After this thresholding, if there is no good matches it passes to draw matched points image. On the other hand if the results has good matches, there will be a calculation to produce a percentage of good matches to the all matches where total counts ***TC*** is 100 and good counts ***GC*** is:

$$GC = \frac{Good\ Matches * 100}{All\ Matches} \tag{4.2}$$

$$\begin{cases} if\ All\ Matches < 200 & GC = 0 \\ if\ GC < 25 & GC = 0 \\ if\ Good\ Matches = 0 & GC = 0 \end{cases} \tag{4.3}$$

where negative conditions shown in Equation 4.3. If negative conditions are false for this pair of matching images result, it will pass GC as defined in Equation 4.2. After this normalization, according to this study results, if the GC value above 25, this pair of matching images will be considered as images of the same pedestrian and its face name will be added to the face list to form the pedestrian face dataset and will be passed to the next matching. On the other side, after matching calculations, negative conditions will trigger the end of the face dataset because the matching images are

not belong to the same pedestrian and when this happens face dataset will be updated with the corresponding group ID on face databases of both inside and outside video feeds. After processing all consecutive pedestrian images on inside and outside video feeds this matching phase will be completed and ready to process the third step.

Third image processing phase applies face recognition with Eigenfaces implemented with the Face Recognizer class in OpenCV. After second step, database will provide the pedestrian groups labeled on the face tables, first of all face recognition for inside video files evaluated to find all unique pedestrians by running face dataset training for each group of faces.

To find same pedestrian which appears in the inside video feeds, a loop starts for each group ID, when running inside the loop every group ID, except the one that has the loop identifier, i^{th} group ID, will be trained for face recognition, after training, i^{th} group ID's face dataset will be predicted on this trained model to find if any reappearance of the i^{th} pedestrian exist with another group ID. Prediction will assume that there should be a chosen group ID even though the predicted group ID's pedestrian never reappeared, so this problem will be overcome by using the predictions' confidence values which is a double variable indicates the confidence of the prediction, in this study, value 1500 will be sufficient to eliminate non-related predictions of the i^{th} group ID. If the predictions has a mean under this 1500 threshold value for prediction group ID set, this prediction group ID and i^{th} group ID suppose to be same pedestrian. Then matched pairs will be reserved in an array of pairs. To decrease the group ID amount to the actual unique number of the pedestrians, the matched pairs will be used to change group ID of the reappearing pedestrians to the group ID of first appearance made with the pedestrians. Pairs consist two group ID that is presumed labels the same pedestrian, for more than two appearance there will be different group of pairs point out the same pedestrian, to overcome this problem, pairs will be searched and same pedestrian's different group ID's collected to one array.

After finishing uniting all group IDs of each pedestrian, database will be updated with new group IDs and then re-training the face recognizer with updated

group IDs will be the last step of training face recognizer. From this point the face recognization is ready to predict outside video feeds.

When group IDs are unique for each pedestrian, system can report entering times of the pedestrian with querying ***facein*** database for the pedestrian group ID for the ***vidin*** table code values, after query, with a loop system can report the entering times of the pedestrian to the graphical user interface. And after processing inside video feeds, face recognizer can process the outside video feeds to detect pedestrians from their faces and report leaving times of the pedestrian with querying ***faceout*** database for the pedestrian group ID for the ***vidout*** table code values, after query, with a loop, system can report to the graphical user interface.

5. RESEARCH AND DISCUSSION

In this section, benchmark results are presented which evaluated on datasets. The benchmarks evaluated under Microsoft 7 Home Premium operating system with Intel i7 Q740 1.73 GHz CPU and 6 GB RAM computer hardware. In the first benchmark, pedestrian and face extracting evaluated for runtime and accuracy performance. In the second benchmark, pedestrian matching evaluated for grouping pedestrians. And In the third benchmark pedestrian recognizing evaluated for recognizing same pedestrian that passes multiple times.

5.1. Pedestrian and Face Extraction Benchmarks

In this benchmark, pedestrian and face extracting evaluated with the five different pre-trained front face configuration files:

- haarcascade_frontalface_alt.xml
- haarcascade_frontalface_alt_tree.xml
- haarcascade_frontalface_alt2.xml
- haarcascade_frontalface_default.xml
- lbpcascade_frontalface.xml

and benchmark results fetched from the time consumed for video dataset file total time and per frame time in milliseconds for runtime performance and total faces founded in video dataset file and the amount of true positive and false positive for accuracy performance.

For a brief conclusion for this benchmark, haarcascade_frontalface_alt.xml configuration file is give best performance by far with low false positive values. Beside that most of the time, xml configuration files results give close amount of true positive values. And the video dataset files that has the leaving action happening gives low and all false positive values for face detection because pedestrians' faces

turned back to the cameras. All benchmarks evaluated with a 1300*400 defined region to decrease the processing time and false detections from out of region.

Table 5.1. Extraction results with haarcascade_frontalface_default.xml for indoor video feeds.

Video Name	Frames	Total Time (milliseconds)	Frame Time (milliseconds)	Total Face	TP	FP
inP1C1	116	66610	574	45	44	1
inP1C2	120	68502	570	43	41	2
inP1C3	113	63536	562	42	40	2
inP1G1	178	136774	768	2	0	2
inP1G2	150	115632	770	3	0	3
inP1G3	166	125359	755	7	0	7
inP2C1	122	67378	552	49	46	3
inP2C2	151	86955	575	65	58	7
inP2G1	168	128275	763	6	0	6
inP2G2	147	109589	745	18	0	18
inP3C1	135	78901	584	48	37	11
inP3C2	123	69821	567	43	40	3
inP3C3	164	95603	582	59	58	1
inP3G1	169	127753	755	10	0	10
inP3G2	167	126943	760	8	0	8
inP3G3	157	115551	735	4	0	4
inP4C1	144	81757	567	59	57	2
inP4C2	133	77449	582	52	50	2
inP4C3	115	59469	517	53	50	3
inP4G1	160	122117	763	2	0	2
inP4G2	156	119985	769	1	0	1
inP4G3	166	123931	746	9	0	9
inP5C1	114	62234	545	47	46	1
inP5C2	124	71504	576	41	40	1
inP5G1	154	110961	720	11	0	11
inP5G2	174	126570	727	6	0	6

Table 5.2. Extraction results with haarcascade_frontalface_default.xml for outside video feeds.

Video Name	Frames	Total Time (milliseconds)	Frame Time (milliseconds)	Total Face	TP	FP
outP1C1	156	99229	636	54	32	22
outP1C2	107	63031	589	44	39	5
outP1C3	137	85227	622	54	35	19
outP1G1	164	121394	740	27	0	27
outP1G2	160	117536	734	29	0	29
outP1G3	148	111518	753	20	0	20
outP2C1	134	76240	568	66	52	14
outP2C2	163	88686	544	77	53	24
outP2G1	172	124606	724	19	0	19
outP2G2	163	120484	739	30	0	30
outP3C1	156	101096	648	52	37	15
outP3C2	117	70016	598	42	37	5
outP3C3	110	69753	634	48	34	14
outP3G1	178	129879	729	15	0	15
outP3G2	166	122080	735	20	0	20
outP3G3	160	117257	732	17	0	17
outP4C1	135	79286	587	65	46	19
outP4C2	118	61868	524	70	48	22
outP4C3	112	58877	525	74	52	22
outP4G1	136	102879	756	11	0	11
outP4G2	161	114520	711	22	0	22
outP4G3	151	111645	739	17	0	17
outP5C1	116	70658	609	42	39	3
outP5C2	133	80840	607	56	40	16
outP5G1	162	119904	740	23	0	23
outP5G2	201	141977	706	18	0	18

Table 5.3. Extraction results with haarcascade_frontalface_alt.xml for indoor video feeds.

Video Name	Frames	Total Time (milliseconds)	Frame Time (milliseconds)	Total Face	TP	FP
inP1C1	116	42613	367	43	42	1
inP1C2	120	43877	365	42	42	0
inP1C3	113	41819	370	39	39	0
inP1G1	178	72373	406	0	0	0
inP1G2	150	61822	412	0	0	0
inP1G3	166	244528	1473	1	0	1
inP2C1	122	45783	375	38	38	0
inP2C2	151	53906	356	51	50	1
inP2G1	168	69149	411	0	0	0
inP2G2	147	61703	419	0	0	0
inP3C1	135	52577	389	34	30	4
inP3C2	123	46174	375	39	39	0
inP3C3	164	59881	365	48	48	0
inP3G1	169	69998	414	1	0	1
inP3G2	167	68496	410	1	0	1
inP3G3	157	64970	413	1	0	1
inP4C1	144	50877	353	52	51	1
inP4C2	133	47921	360	46	45	1
inP4C3	115	41150	357	46	46	0
inP4G1	160	65450	409	0	0	0
inP4G2	156	64015	410	1	0	1
inP4G3	166	67675	407	4	0	4
inP5C1	114	41857	367	41	41	0
inP5C2	124	45548	367	37	36	1
inP5G1	154	63036	409	6	0	6
inP5G2	174	69533	399	1	0	1

Table 5.4. Extraction results with haarcascade_frontalface_alt.xml for outside video feeds.

Video Name	Frames	Total Time (milliseconds)	Frame Time (milliseconds)	Total Face	TP	FP
outP1C1	156	57294	367	31	31	0
outP1C2	107	39277	367	40	39	1
outP1C3	137	50036	365	35	35	0
outP1G1	164	65584	399	3	0	3
outP1G2	160	63969	399	1	0	1
outP1G3	148	59992	405	0	0	0
outP2C1	134	46689	348	52	52	0
outP2C2	163	57535	352	53	52	1
outP2G1	172	68582	398	2	0	2
outP2G2	163	65027	398	0	0	0
outP3C1	156	57941	371	37	37	0
outP3C2	117	44108	376	36	36	0
outP3C3	110	42001	381	33	33	0
outP3G1	178	71493	401	0	0	0
outP3G2	166	67014	403	1	0	1
outP3G3	160	65366	408	1	0	1
outP4C1	135	48702	360	42	41	1
outP4C2	118	41321	350	48	48	0
outP4C3	112	39687	354	51	50	1
outP4G1	136	55925	411	3	0	3
outP4G2	161	64703	401	0	0	0
outP4G3	151	60872	403	1	0	1
outP5C1	116	42587	367	38	38	0
outP5C2	133	48310	363	39	39	0
outP5G1	162	65274	402	2	0	2
outP5G2	201	79658	396	3	0	3

Table 5.5. Extraction results with haarcascade_frontalface_alt2.xml for indoor video feeds.

Video Name	Frames	Total Time (milliseconds)	Frame Time (milliseconds)	Total Face	TP	FP
inP1C1	116	49922	430	52	47	5
inP1C2	120	52700	439	51	45	6
inP1C3	113	50668	448	45	40	5
inP1G1	178	99538	559	11	0	11
inP1G2	150	83603	557	16	0	16
inP1G3	166	90694	546	21	0	21
inP2C1	122	52615	431	55	48	7
inP2C2	151	63412	419	65	53	12
inP2G1	168	91410	544	22	0	22
inP2G2	147	80578	548	20	0	20
inP3C1	135	61889	458	52	31	21
inP3C2	123	54847	445	47	35	12
inP3C3	164	70901	432	63	49	14
inP3G1	169	94542	559	14	0	14
inP3G2	167	90860	544	16	0	16
inP3G3	157	82159	523	8	0	8
inP4C1	144	56790	394	63	55	8
inP4C2	133	53902	405	52	50	2
inP4C3	115	42645	370	59	51	8
inP4G1	160	83105	519	8	0	8
inP4G2	156	81102	519	7	0	7
inP4G3	166	84634	509	21	0	21
inP5C1	114	46064	404	49	43	6
inP5C2	124	51739	417	52	40	12
inP5G1	154	76856	499	16	0	16
inP5G2	174	86679	498	24	0	24

Table 5.6. Extraction results with haarcascade_frontalface_alt2.xml for outside video feeds.

Video Name	Frames	Total Time (milliseconds)	Frame Time (milliseconds)	Total Face	TP	FP
outP1C1	156	68169	436	59	30	29
outP1C2	107	43326	404	46	39	7
outP1C3	137	58539	427	53	35	18
outP1G1	164	81810	498	32	0	32
outP1G2	160	80613	503	33	0	33
outP1G3	148	73520	496	25	0	25
outP2C1	134	51558	384	78	52	26
outP2C2	163	64459	395	83	53	30
outP2G1	172	85818	498	22	0	22
outP2G2	163	80649	494	32	0	32
outP3C1	156	68834	441	64	36	28
outP3C2	117	50361	430	45	35	10
outP3C3	110	47364	430	51	34	17
outP3G1	178	91107	511	23	0	23
outP3G2	166	83994	505	34	0	34
outP3G3	160	81308	508	22	0	22
outP4C1	135	54070	400	66	46	20
outP4C2	118	45793	388	64	48	16
outP4C3	112	41730	372	69	52	17
outP4G1	136	68655	504	21	0	21
outP4G2	161	80001	496	29	0	29
outP4G3	151	75059	497	30	0	30
outP5C1	116	47264	407	46	39	7
outP5C2	133	55019	413	61	40	21
outP5G1	162	79448	490	35	0	35
outP5G2	201	98711	491	20	0	20

Table 5.7. Extraction results with haarcascade_frontalface_alt_tree.xml for indoor video feeds.

Video Name	Frames	Total Time (milliseconds)	Frame Time (milliseconds)	Total Face	TP	FP
inP1C1	116	59126	509	57	46	11
inP1C2	120	59232	493	72	46	26
inP1C3	113	56712	501	59	43	16
inP1G1	178	121889	684	34	0	34
inP1G2	150	93978	626	44	0	44
inP1G3	166	106911	644	41	0	41
inP2C1	122	57348	470	67	48	19
inP2C2	151	68219	451	93	56	37
inP2G1	168	102140	607	54	0	54
inP2G2	147	89244	607	45	0	45
inP3C1	135	70421	521	68	31	37
inP3C2	123	61498	499	64	35	29
inP3C3	164	80384	490	79	53	26
inP3G1	169	105866	626	43	0	43
inP3G2	167	103913	622	43	0	43
inP3G3	157	93566	595	56	0	56
inP4C1	144	69292	481	79	54	25
inP4C2	133	65016	488	72	50	22
inP4C3	115	51480	447	72	50	22
inP4G1	160	103250	645	41	0	41
inP4G2	156	102221	655	39	0	39
inP4G3	166	101309	610	45	0	45
inP5C1	114	54416	477	67	47	20
inP5C2	124	62659	505	67	40	27
inP5G1	154	97874	635	45	0	45
inP5G2	174	109231	627	59	0	59

Table 5.8. Extraction results with haarcascade_frontalface_alt_tree.xml for outside video feeds.

Video Name	Frames	Total Time (milliseconds)	Frame Time (milliseconds)	Total Face	TP	FP
outP1C1	156	84127	539	70	32	38
outP1C2	107	49382	461	75	39	36
outP1C3	137	66422	484	87	36	51
outP1G1	164	95377	581	57	0	57
outP1G2	160	93160	582	62	0	62
outP1G3	148	85290	576	61	0	61
outP2C1	134	61339	457	81	52	29
outP2C2	163	75212	461	105	52	53
outP2G1	172	96962	563	66	0	66
outP2G2	163	92512	567	71	0	71
outP3C1	156	77486	496	88	35	53
outP3C2	117	56826	485	75	35	40
outP3C3	110	54065	491	61	33	28
outP3G1	178	102775	577	67	0	67
outP3G2	166	98699	594	71	0	71
outP3G3	160	91922	574	60	0	60
outP4C1	135	61759	457	88	45	43
outP4C2	118	53032	449	77	48	29
outP4C3	112	49242	439	75	52	23
outP4G1	136	80063	588	42	0	42
outP4G2	161	93806	582	51	0	51
outP4G3	151	91232	604	45	0	45
outP5C1	116	55038	474	61	39	22
outP5C2	133	64709	486	75	40	35
outP5G1	162	93426	576	58	0	58
outP5G2	201	115027	572	78	0	78

Table 5.9. Extraction results with lbpcascade_frontalface.xml for indoor video feeds.

Video Name	Frames	Total Time (milliseconds)	Frame Time (milliseconds)	Total Face	TP	FP
inP1C1	116	39918	344	47	47	0
inP1C2	120	41438	345	43	42	1
inP1C3	113	39436	348	42	41	1
inP1G1	178	51946	291	0	0	0
inP1G2	150	45137	300	1	0	1
inP1G3	166	48821	294	0	0	0
inP2C1	122	41171	337	43	43	0
inP2C2	151	49398	327	55	52	3
inP2G1	168	49578	295	2	0	2
inP2G2	147	44112	300	1	0	1
inP3C1	135	44744	331	43	39	4
inP3C2	123	41506	337	43	39	4
inP3C3	164	53101	323	56	51	5
inP3G1	169	50009	295	3	0	3
inP3G2	167	49264	294	3	0	3
inP3G3	157	46489	296	2	0	2
inP4C1	144	47717	331	56	55	1
inP4C2	133	44515	334	50	48	2
inP4C3	115	39994	347	52	52	0
inP4G1	160	46959	293	0	0	0
inP4G2	156	45709	293	0	0	0
inP4G3	166	48353	291	2	0	2
inP5C1	114	39007	342	47	47	0
inP5C2	124	41029	330	42	41	1
inP5G1	154	45279	294	5	0	5
inP5G2	174	49789	286	1	0	1

Table 5.10. Extraction results with lbpcascade_frontalface.xml for outside video feeds.

Video Name	Frames	Total Time (milliseconds)	Frame Time (milliseconds)	Total Face	TP	FP
outP1C1	156	46773	299	31	30	1
outP1C2	107	36826	344	42	39	3
outP1C3	137	42740	311	34	32	2
outP1G1	164	45421	276	1	0	1
outP1G2	160	44586	278	1	0	1
outP1G3	148	41943	283	2	0	2
outP2C1	134	44295	330	51	50	1
outP2C2	163	51318	314	55	51	4
outP2G1	172	47568	276	5	0	5
outP2G2	163	45768	280	5	0	5
outP3C1	156	47878	306	39	36	3
outP3C2	117	38891	332	38	36	2
outP3C3	110	37236	338	37	34	3
outP3G1	178	49918	280	13	0	13
outP3G2	166	46637	280	5	0	5
outP3G3	160	44882	280	3	0	3
outP4C1	135	43203	320	43	43	0
outP4C2	118	40526	343	49	48	1
outP4C3	112	38764	346	50	48	2
outP4G1	136	39717	292	7	0	7
outP4G2	161	45109	280	3	0	3
outP4G3	151	42513	281	3	0	3
outP5C1	116	37957	327	38	37	1
outP5C2	133	42210	317	40	40	0
outP5G1	162	45426	280	5	0	5
outP5G2	201	54180	269	2	0	2

Pedestrian and face extraction process is a heavy time consuming image processing phase, XML configure file must be chosen carefully to lower the time for processing. According to these results, haarcascade_frontalface_alt.xml and lbpcascade_frontalface.xml XML files gives good results with low false positive face detections. In this benchmark, haarcascade_frontalface_alt.xml XML file has lower true positive detections than lbpcascade_frontalface.xml XML file but with lower false positive detections. And, for a decent face recognization every pedestrian must

be grouped with at least 15 images of face detected so haarcascade_frontalface_alt.xml file can be used with least time for processing.

5.2. Pedestrian Matching Benchmarks

In this study, matching pedestrians is very important step to deduct the group of faces from the pedestrians to train a face recognizer model. This benchmark run with five different XML files on inside and outside video feeds. A good pedestrian matching will provide enough face samples to run face recognizer with an acceptable accuracy.

Table 5.11. Grouping results after pedestrian matching process for each XML file.

	Cascade Classifier XML File									
	default		Alt		alt2		alt_tree		lbp	
ID	In	Out	In	Out	In	Out	In	Out	In	Out
-1	240	489	89	36	423	619	1032	1278	245	102
0	56	30	46	30	21	30	25	31	26	34
1	39	36	41	35	46	36	25	36	46	35
2	33	45	28	40	35	45	45	44	19	42
3	48	51	39	51	46	51	28	51	33	42
4	42	38	26	37	32	38	34	39	46	36
5	46	38	39	38	29	38	21	38	19	39
6	40	35	43	35	46	35	43	35	22	35
7	39	47	41	47	45	47	42	47	21	50
8	31	39	35	39	39	39	46	39	18	39
9	37	33	21	32	21	33	39	33	31	33
10	39	29	43	33	44	29	42	34	23	28
11	44	51	38	51	39	52	39	51	38	47
12	-	51	20	48	20	51	18	51	34	40
13	-	-	23	-	23	-	23	-	18	-

As seen in Table 5.11. false positive face detections can't be grouped and remains under -1 group ID, which is the ungrouped default value for group ID. In both, inside and outside video feeds, there are 13 pedestrians entering and leaving

videos. So the grouping must find 13 groups for inside and outside video feeds. Because of unwanted interrupts inside consecutive video frame some of the groups can be divided into two groups. And on the other side, matching can connect two groups as one group which decrease the founded groups count. Matching pedestrian can be time consuming with the amount of false positive faces and Pedestrian Watcher does not classifies these faces under any group. These results point out that using haarcascade_frontalface_alt.xml XML file to detect faces is most effective for both runtime and accuracy performance.

5.3. Pedestrian Recognition Results

Pedestrian recognition results, has two sub parts, first part is the classification of incoming pedestrians from inside video feeds, for a building surveillance system, entering inside is the first action of the pedestrian, so pedestrian recognization must start from inside video feeds and after finishing classification system can pass to the second part, predict outside video feeds because all outside feeds are absolutely belongs to a pedestrian from inside video feeds.

In this study, haarcascade_frontalface_alt.xml XML file based pedestrian recognition process evaluated and as seen in the Table 5.11. there are 13 groups for inside video feeds and 12 groups for outside video feeds founded after pedestrian matching. For outside video feeds there is one pedestrian group undetected.

Table 5.12. Face Recognizer training results for inside video feeds.

Pedestrian	Actual Group ID	Predicted Group ID	Trimmed Group ID
P1	1 - 7 - 11	1 - 7 - 11	1
P2	2 -12 - 13	2 -12 - 13	2
P3	0 - 4 - 9	0 - 4	0
P4	3 - 6 - 10	3 - 6 - 10	3
P5	5 - 8	5 - 8	5
P3	-	9	9

After processing face recognition training, Pedestrian Watcher decreased to 6 groups of pedestrian and actual unique pedestrian count was 5. Group ID number 9 from pedestrian *P3* left out from its united pedestrian ID groups. After trimming the pedestrian group IDs, face recognizer will be re-trained with decreased pedestrian IDs for better prediction for outside video feeds.

Table 5.13. Face Recognizer predicts for outside video feeds

Pedestrian	Actual ID	Predicted ID	Count	Mean
P1	1	1	8	2990
P1	1	5	13	3085
P1	1	3	8	3310
P1	1	2	1	3514
P3	0	0	7	2745
P3	0	5	28	2774
P4	3	3	40	3248
P2	2	5	12	3044
P2	2	2	39	3246
P5	5	5	37	3195
P1	1	5	27	3139
P1	1	1	11	3376
P3	0	5	30	2647
P3	0	0	5	3027
P4	3	3	47	3180
P5	5	5	34	3055
P5	5	1	5	3410
P3	0	5	20	2622
P3	0	0	12	3032
P1	1	1	8	2991
P1	1	5	11	3114
P1	1	3	14	3339
P2	2	1	1	2872
P2	2	2	38	3148
P2	2	5	12	3151
P4	3	3	48	3229

Prediction of outside video feeds results showed in Table 5.13. Consecutive pedestrian results presents for one outside video feed group ID, some predictions only one result which is the true prediction of the pedestrian and the others has two or more prediction which one of them are the true prediction of pedestrian. More than one prediction results makes it difficult to identify the pedestrian identity. At this point to decrease to one prediction, count and mean of the prediction can be used to order them. Count shows the number of predicted ID and mean is the mean of all predicted ID's confidence values. A lower mean shows a better recognization and high count values indicates the pedestrian prediction is the most predicted group ID. If prediction results decreased to one prediction considering the mean of predictions, outside video feeds predicted with 8 true and 5 false predicted group ID, on the other hand if the predicted results decreased to one prediction considering the count amount of the predictions, outside video feeds predicted with 7 true and 6 false predicted group ID. Because of P3 pedestrian uniting group IDs performance was poor, P3 group ID predictions for outside video feeds also gives inaccurate results. Origin of the problem, comes from the video quality of the P3 pedestrian video feeds.

Beside this video and image quality problem, all three steps show fairly efficient performance to aid people in charge for video surveillance systems.

6. CONCLUSION

In this thesis, an intelligent video surveillance system is designed and developed. Before development of the software, face and pedestrian detection evaluated for runtime and accuracy performance to find an optimum face and pedestrian detection infrastructure for the Pedestrian Watcher software. In benchmark results, for a 1300*400 pixel resolution all configurations consume more than 500 ms with CPU computing, this processing time can be decrease by resizing the region size to half size and using GPU aided computing.

In the second phase of image processing same performance enhancement can be applied to decrease the processing time with the sacrificing some key points founded and could be matched in two pedestrian images.

Third phase of the image processing results are depend on the first phase of the image processing and video recording quality. A progressive scanning camera should eliminate the blurred images which leads to wrong and multiple prediction results for outside video feeds.

As a conclusion, in this study, after the enhancements for performance, Pedestrian Watcher software could help the people in charge of the passive video surveillance systems for easily watch entering and leaving actions of the pedestrians in the buildings.

REFERENCES

ANTON, A. and RORRES, C., 2000. Elementary Linear Algebra Applications Version, 8th Edition, John Wiley & Sons, Inc., USA, 822p.

BAGGIO, D. L., EMAMI, S., ESCRIVA, D. M., LEVGEN, K., MAHMOOD, N., SARAGIH, J. and SHILKROT, R., 2012. Mastering OpenCV with Practical Computer Vision Projects. Packt Publishing, Birmingham, 340 p.

BENEZETH, Y., JODOIN P., EMILE, B., ROSENBERGER, C. and LAURENT, H., 2010. Comparative Study of Background Subtraction Algorithms. Journal of Electronic Imaging, 19(3):033003.

BRADSKI, G. and KAEHLER, A., 2008. Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media, 1st Edition, USA, 555 p.

CALONDER, M., LEPETIT, V., STRECHA, C. and FUA, P., 2010. BRIEF Binary Robust Independent Elementary Features. European Conference on Computer Vision, 2010, pp. 778-792.

COMANICIU, D. and MEER, P., 1999. Mean Shift Analysis and Applications. Computer Vision, 1999, Volume 2, pp. 1197-1203.

DALAL, N. and TRIGGS, B., 2005. Histogram of Oriented Gradients for Human Detection. Computer Vision and Pattern Recognition, 2005, Volume 1, pp. 886-893.

DESURMONT, X., BASTIDE, A., CZYZ, J., PARISOT, C., DELAIGLE, J. and MACQ, B., 2005. A General-Purpose System for Distributed Surveillance and Communication. (Editors: VELASTIN, S. A. and REMAGNINO, P.), Intelligent Distributed Video Surveillance Systems, New cover, The Institution of Engineering and Technology, London, pp. 121-156.

EL BAF, F., BOUWMANS, T. and VACHON, B., 2008a. Foreground Detection Using the Choquet Integral. Image Analysis for Multimedia Interactive Services, 2008, pp. 187-190.

EVERINGHAM, M., 2006. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results, 2006. http://pascallin.ecs.soton.ac.uk/challenges/VOC/databases.html

EZUST, A. and EZUST, P., 2012. An Introduction to Design Patterns in C++ with Qt, 2nd Edition, Prentice Hall, USA, 768 p.

HARRIS, C. and STEPHENS, M., 1988. A Combined Corner and Edge Detector. Alvey Vision Conference, 1988, 2, pp. 147-151.

HOTELLING, H., 1933. Analysis of a Complex of Statistical Variables into Principal Components. Journal of Educational Psychology, 24, pp. 417-441 and 498-520.

LAGANIERE, R., 2011. OpenCV 2 Computer Vision Application Programming Cookbook. Packt Publishing, Birmingham, 304 p.

LIENHART, R., KURANOV, A. and PISAREVSKY, V., 2002. Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. Pattern Recognition, 2003, Volume 2781, pp. 297-304.

LOWE, D. G., 2004. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 2004, Volume 60(2), pp. 91-110.

MAKRIS, D., ELLIS, T. and BLACK, J., 2004. Bridging the Gap Between Cameras. Computer Vision and Pattern Recognition, 2004, Volume 2, pp. 205-210.

MOHAN, A., PAPAGEORGIOU, C. and POGGIO T., 2001. Example-based Object Detection in Images by Components. Pattern Analysis and Machine Intelligence, 2001, 23(4), pp. 349-361.

OLIVER, N. M., ROSARIO, B. and PENTLAND, A. P., 2000. A Bayesian Computer Vision System for Modelling Human Interactions. Pattern Analysis and Machine Intelligence, 2000, 22(8), pp. 831-843.

QUINLAN, J. R., 1986. Induction of Decision Trees. Machine Learning, 1986, Volume 1, pp. 81-106.

PEARSON, K., 1901. On Lines and Planes of Closest Fit to Systems of Points in Space. Philosophical Magazine, 2, pp. 559-572.

RINNER, B., JOVANOVIC, M. and QUARITSCH, M., 2007. Embedded Middleware on Distributed Smart Cameras. Acoustics, Speech and Signal Processing, 2007, Volume 4, pp. 1381-1384.

ROSIN. P. L., 1999. Measuring Corner Properties. Computer Vision and Image Understanding, 1999, Volume 93(2), pp. 291-307.

ROSTEN, E. and DRUMMOND, T., 2006. Machine Learning for High-speed Corner Detection. European Conference on Computer Vision, 2006, Volume Part I, pp. 430-443.

ROY-CHOWDHURY, A. K. and SONG, B., 2012. Camera Networks: The Acquision and Analysis of Videos over Wide Areas. Morgan & Claypool Publishers, USA, 134 p.

RUBLEE, E., RABAUD, V., KONOLIGE, K. and BRADSKI, G., 2011. ORB: An Efficient Alternative to SIFT or SURF. Computer Vision, 2011, pp. 2564-2571.

STAUFFER, C. and GRIMSON W.E.L., 1999. Adaptive Background Mixture Models for Real-time Tracking. Computer Vision and Pattern Recognition, 1999, Volume 2.

TURK, M. A. and PENTLAND, A., 1991. Face Recognition Using Eigenfaces. Computer Vision and Pattern Recognition, 1991, pp. 586-591.

VALERA, M. and VELASTIN, S. A., 2006. A Review of the state-of-the-art in Distributed Surveillance Systems. (Editors: VELASTIN, S. A. and REMAGNINO, P.), Intelligent Distributed Video Surveillance Systems, New cover, The Institution of Engineering and Technology, London, pp. 1-30.

_____,2008. (Editors: VELASTIN, S. A. and REMAGNINO, P.), Intelligent Distributed Video Surveillance Systems, New cover, The Institution of Engineering and Technology, London, 304 p.

VIOLA, P. and JONES, M., 2001. Rapid Object Detection Using a Boosted Cascade of Simple Features. Computer Vision and Pattern Recognition, 2001, Volume 1, pp. I-511 - I-518.

VIOLA, P. and JONES, M. and SNOW, D., 2003. Detecting Pedestrians Using Patterns of Motion and Appearance. Computer Vision, 2003, Volume 2, pp. 734-741.

WELCH, G. and BISHOP, G., 1995. An Introduction to the Kalman Filter. (Technical Report) An Introduction to the Kalman Filter, University of North Caroline at Chapel Hill, Chapel Hill, NC, USA, 1995.

CURRICULUM VITAE

He was born in Adana, Turkey in 1985. He achieved Çukurova University Faculty of Engineering and Architecture Department of Computer Engineering in 2004 and completed his undergraduate education in 2009. After working in software companies inside Adana Teknokent and Gaziantep Teknopark, he started to work in Osmaniye Korkut Ata University as a research assistant at Faculty of Engineering Department of Computer Engineering in 2010. Currently, he is still working in the same department.