Video Analytics and WebRTC

Major Project

Submitted in partial fulfillment of the requirements for the degree of

Master of Technology

in

Electronics & Communication Engineering
(Embedded Systems)

By

RONAK DOSHI
(12MECE38)

Electronics & Communication Engineering Branch

Electrical Engineering Department

Institute of Technology

Nirma University

Ahmedabad-382 481

May 2014

Video Analytics and WebRTC

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology
in
Electronics & Communication Engineering
(Embedded Systems)

By

RONAK DOSHI
(12MECE38)

Under the guidance of

External Project Guide:

Mr. Sreekanth Rao
Sr.Tech Lead,APS
Aricent Groups,
Bangalore.

Internal Project Guide:

Dr. Yogesh N. Trivedi
Associate Professor, EC Dept,
Institute of Technology,
Nirma University, Ahmedabad.

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481

May 2014

Declaration

This is to certify that

a. The thesis comprises my original work towards the degree of Master of Technology in Embedded Systems at Nirma University and has not been submitted elsewhere for a degree.

b. Due acknowledgment has been made in the text to all other material used.

- RONAK DOSHI

Certificate

This is to certify that the Major Project entitled **"Video Analytics and WebRTC"** submitted by **Ronak Doshi (12MECE38)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in Embedded System, Electronics and Communication Engineering (Embedded Systems) of Nirma University of Science and Technology, Ahmedabad is the record of work carried out by him under my supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Date: Place: Ahmedabad

Dr. Y.N. Trivedi
Guide

Dr. N.P. Gajjar
Program Coordinator

Dr. P.N.Tekwani
Head of EE Dept.

Dr. K. Kotecha
Director, IT

Acknowledgements

I would like to express my gratitude and sincere thanks to **Dr. P.N.Tekwani**, Head of Electrical Engineering Department, and **Dr. N.P.Gajjar**, PG Coordinator of M.Tech Embedded Systems program for allowing me to undertake this thesis work and for his guidelines during the review process.

I take this opportunity to express my profound gratitude and deep regards to **Dr. Yogesh N. Trivedi**, Professor ,Nirma University, for his exemplary guidance, monitoring and constant encouragement throughout the course of this thesis. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I also take this opportunity to express a deep sense of gratitude to Company Mentor **Mr. Sreekanth Rao**,Sr.Tech Lead, Aricent Group. for his cordial support, constant supervision as well as for providing valuable information regarding the project and guidance, which helped me in completing this task through various stages.

I am obliged to staff members of Multimedia Department, Motorola Solutions. for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Lastly, I thank almighty, my parents, brother, sisters and friends for their constant encouragement without which this assignment would not be possible.

- Ronak Doshi

12MECE38

Abstract

Video Analytics which is also known as Video Content Analysis(VCA) is the capability of Automatically analyzing Video and to detect events not based on a single image. It will analyze video streams run time to detect events related to particular type of condition.It will be helpful for applications such as Intrusion Detection,Crowd Management,Baggage Detection.

WebRTC stands for Web Real-Time Communication. Aim of the Project is to "Stream Real time Streaming from Surveillance cameras to Browser"which will support camera list, multiple streaming and PTZ(pann/till/zoom). WebRTC contains main three components WebRTC Client,WebRTC gateway,Media Server.
Client(Browser) supports multiple real-time streaming of surveillance cameras with support of PTZ. **Media Server** is a combination of RTSP Server and Streaming Server. To stream RTP packets from Media Server to Client it requires Signaling mechanism to complete. WebRTC Client has its own specification such as it requires WebSocket ,Ice-Stun,DTLS-SRTP Signaling, and can Play only Secured RTP packets and support VP8 codec only. Media Server does not support websocket protocol, it supports rtsp signaling and supports H264 codec and stream RTP packets only. To request Streaming from Media Server and to support PLAY,PAUSE and STOP actions from client RTSP signaling mechanism is required to follow. **WebRTC gateway** is added as a mediator between WebRTC Client and Media Server. In WebRTC Gateway supports WebSocket Signaling,RTSP Signaling,Ice Mechanism for NAT Traversal. it supports DTLS to export keying material to encrypt RTP and RTCP packets. After following Signaling mechanism, Gateway will receive RTP and RTCP packets from Media Server,encrypt it as SRTP and SRTCP packets and route it to WebRTC Client.

Abbreviation Notation and Nomenclature

WebRTC	Web Real time Communication
SDP	Session Description Protocol
RTP	Real-time Transport Protocol
RTCP	Real-time Control Protocol
SRTP	Secured RTP
SRTCP	Secured RTCP
DTLS	Datagram Transport Layer Security
RTSP	Real-time Streaming Protocol
ICE	interactive connectivity establishment
STUN	Session Traversal Utilities for NAT
VCA	Video Content Analysis
NVR	Network Video Recorder
PTZ	Pan/Tilt/Zoom
XML	Extensible Markup Language
RIC	Real-Time Intelligence Console
RTVI	Real Time Video Intelligence
HTTP	Hypertext Transfer Protocol
VGA	Video Graphics Array
QVGA	Quarter Video Graphics Array
FPS	Frames Per Second
LBPH	Local Binary Pattern Histogram
PCA	Principle Component Analysis
API	Application Programming Interface
FOV	Field Of View

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Video Analytics is enabling a rapidly growing number of embedded video products such as smart cameras and intelligent digital video recorders (DVRs) with automated capabilities that just a few years ago would have required human monitoring. Broadly, video analytics is the extraction of meaningful and relevant information from digital video. As opposed to video compression, which attempts to exploit the redundancy in digital video for the purpose of reducing size, analytics is concerned with understanding the content of video. Video Analytics builds upon research in computervision, pattern analysis and machine intelligence, and spans several industry segments including surveillance, retail and transportation. It is also called video content analysis (VCA) or intelligent video.

A **facial recognition** system is a computer application for identifying and verifying a person from digital image. or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a facial database.

WebRTC (Web Real-Time Communication) WebRTC is a set of standards from WC3 that will enable real-time communication (RTC) on the web between browsers. Chrome, Firefox, and Opera browsers natively support it. Using WebRTC, you can make peer-to-peer calls, video chats, share screens, and exchange files.

From Public safety point of view in WebRTC streaming from surveillance cameras will come,which will enable all devices which has browser to view streamings of surveil-

lance cameras,and it is plugin free. No need to install any software. Any machine which has browser and which is webrtc enabled can login to specific page and can see atreaming from multiple cameras.
WebRTC is an open source project of google, which provides support for video chat,voice chat,data exchange between browsers(between Peers).
To stream from surveillance camera same concept of webrtc is taken but instead of peer to peer communication it is Server-Client Communication.

In WebRTC main three components were introduced

- WEBRTC Client
- WEBRTC Gateway
- Media Server

RTSP Signaling is used for controlling streaming media servers,with requests sent from WebRTC Gateway.
ICE-STUN protocol for NAT traversal,so that browser from around globe can play any streaming from Media Server.
SRTP and SRTCP protocol to protect RTP packet routing from Media Server to Client. and routing of SRTCP packet from Client to Media Server as Receive Report. Client will send SRTCP Packets only which will be decrypted by Gateway and simple RTCP packet is routed to the Media Server.

1.1 Background

People use Skype type of application for Audio/Video communication, for that they need to install skype application in their machine,
WebRTC provides a mechanism by which people can communicate audio,video between Browser only.This is an open source project developed by Google.

Streaming from surveillance cameras to stream in browser with secured communication is the goal of Project.
To view streams from surveillance cameras special client software is required. We need to install them and plugin is required to play those videos.
Browser based communication was introduced so that in any machine which has browser and which supports webrtc can play streaming from surveillance cameras through media servers.
Multiple Streaming and Multiple Client Support is given. There are many different brands and technology platforms for **Video Analytics**, but they all work on the same basic principles, using pattern recognition and other Algorithms technology to provide two critical capabilities:

- Recognize unusual activities as they happen and notify the security system in real-time.
- Convert video files into a data asset that can be searched, managed and analyzed to improve security and business performance.

Todays Video Analytics software is easy to use and offers growing functionality. For example, it can be programmed to look for specifically defined anomalies. It can even be programmed to give special attention to specific elements in a video framesuch as a computer, door, or filing cabinet. Furthermore, Video Analytics can be integrated with other security and information systems to create new possibilities for using and managing video data.

1.2 Objective of Study

In the past decade, video surveillance cameras have become ubiquitous. Most companies of any size wouldnt think of running a facility without cameras in place. For large operations, video surveillance is a significant investment involving technology, guard stations, video storage and maintenance.

All of this attention to video is justified, because security cameras help protect people as well as property, both physical and intellectual.
After capturing whole video after analysis on recorded video is also tedious task.
One solution for this is a rapidly growing technology called Video Analytics.
Video Analytics as the emerging technology where computer vision is used to filter and manage real time CCTV video for security and intelligent traffic monitoring. Simply put, Video Analytics is an automated approach to managing and analyzing video, without the cost or man-hours previously required.
WeBRTC introduces mechanism of browser based communication to media server. Real-time streaming on browser and client it self communicating with media server through one Gateway which act as a mediator between Client and Media Server which supports routing of data from media server to client and from client to Media Server.

1.3 Scope of Work

1.3.1 Video Analytic

Presented System below does following jobs, In **Video Analytics** First we will configure analytic application for which we want to get events.
After Configuration of Camera in Network Video Recorder is done,whenever some object will come in area of interest an event will be generated. When object moves away from **area of interest(AOI)** it will also generate stop event.
In Video Analytic Project one general frame work will be created which will integrate these events and provide information in more useful manner which includes in form of Video and Snapshots. In Real-Time only it will give triggers and integration of this events on Client console window.
This client window will be with client monitoring lots of cameras. So with this method all analysis and suspicious activity will be analyzed run time and will be sent to client in form of Alarm Events. So client do not have to monitor all the cameras all the

time. Whenever Suspicious activity will happen in area of interest it will pop-up as events.

1.3.2 WEBRTC

WebRTC Client will send request and will send offer SDP(Session Description Protocol) to WebRTC Gateway which will have information of Media Capability and supporting Codec Information.Gateway will Request Media Server as RTSP requests.In response it will get SDP information of Media Server. Gateway will add ICE and media information of Server and replies as Answere SDP to Client.Client-Server Ice negotiation will happen.RTP packets Server will sent to Gateway and Gateway will do SRTP of each packet and will route it to Client.Client will unprotect SRTP packets and decode it and play it.

1.4 Thesis Organization

The rest of the thesis is organized as follows.

Chapter 2, Literature Survey, describes components that was studied for project and Protocols reffered. **Chapter 3**, Video Analytic Triggers, describes what is video analytic,factors need to consider for video analytic application and different types of triggers in application.Video Analytic Architecture, describes Server Based and Edge Based Approach for Video Analytic. **Chapter 4**, Video Analytic Work flow, describes whole working from some prerequisite configuration to getting triiger and posting it to Client. **Chapter 5**, Software and Hardware Tools, describes components used for WebRTC, such as NVR,Media Gateway, WebRTC Client. In **chapter ??**, WebRTC Gateway,Explanation of all signaling mechanism used such as RTSP,WebSocket,ICE,DTLS and explains media part. **Chapter ??**, WebRTC Workflow, describes basic block-diagram and explains each part,brief explationation on SRTP and SRTCP is given **Chapter 7**, Experiments and Results, describes analysis that is done for Receiving RTP and RTCP packets and DTLS-SRTP encryption

and applying crypto policy. Finally, in **chapter 8** concluding remarks and scope for future work is presented.

Chapter 2

Literature Survey

2.1 Video Analytics

2.1.1 Network Video Recorder

A network video recorder (NVR) is a software program that records video in a digital format to a disk drive, USB flash drive, SD memory card or other mass storage device. NiceVisions open platform SDK helps you use NiceVision solutions with third-party applications, or conversely, integrates your system into a NiceVision environment. With rich functionality and powerful building modules, SDK simplifies the integration of all dedicated products, creating an intelligent and comprehensive security environment that is tailored to your needs. NiceVision SDK Components: The NiceVision SDK includes the following Application Programmers Interface (API) layers:

- Player API: The NiceVision Media Player packaged as Windows ActiveX Control.
- Core API: Controls NiceVision DVRs and NVRs with a low level API.
- Database API: Retrieves site information from the NiceVision database in XML format.

```
"offer" "---" "v=0
o=Mozilla-SIPUA-28.0 10806 0 IN IP4 0.0.0.0
s=SIP Call
t=0 0
a=ice-ufrag:87f81a7f
a=ice-pwd:cc5e45e181287bc0ca04ca512eb8270b
a=fingerprint:sha-256 E7:8A:5B:6B:D3:6B:A8:2A:CE:2F:DA:32:B5:E7:57:96:85:61:3A:C5:90:9C:DC:F6:27:15:0A:83:D9:10:BA:55
m=audio 57621 RTP/SAVPF 109 0 8 101
c=IN IP4 192.168.43.209
a=rtpmap:109 opus/48000/2
a=ptime:20
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=sendrecv
a=setup:actpass
a=candidate:0 1 UDP 2128609535 192.168.43.209 57621 typ host
a=candidate:0 2 UDP 2128609534 192.168.43.209 57622 typ host
a=rtcp-mux
m=video 57623 RTP/SAVPF 120
c=IN IP4 192.168.43.209
a=rtpmap:120 VP8/90000
a=sendrecv
a=rtcp-fb:120 nack
a=rtcp-fb:120 nack pli
a=rtcp-fb:120 ccm fir
a=setup:actpass
a=candidate:0 1 UDP 2128609535 192.168.43.209 57623 typ host
a=candidate:0 2 UDP 2128609534 192.168.43.209 57624 typ host
a=rtcp-mux
```

Figure 2.1: Offer SDP

NiceVision SDK Advantages: Simply “drag and drop ”the NiceVision Player into your application

Encapsulates powerful functionality for easy integration Standard methodology across all NiceVision components

Use the same code for all NiceVision DVRs, NVRs, and decoders Rich functionality in a single package

All API modules are packaged into one kit Enhanced software compatibility Backward compatibility SDK supports at least two NiceVision NVR and DVR versions back

Forward compatibility upgrade to new SDK versions without code modifications to third-party systems

Communication Between Standard WebRTC Clients.

When communication between two clients happens they need to convey media details,transport addresses and session description.

SDP(Session Description Protocol) between Client is described.

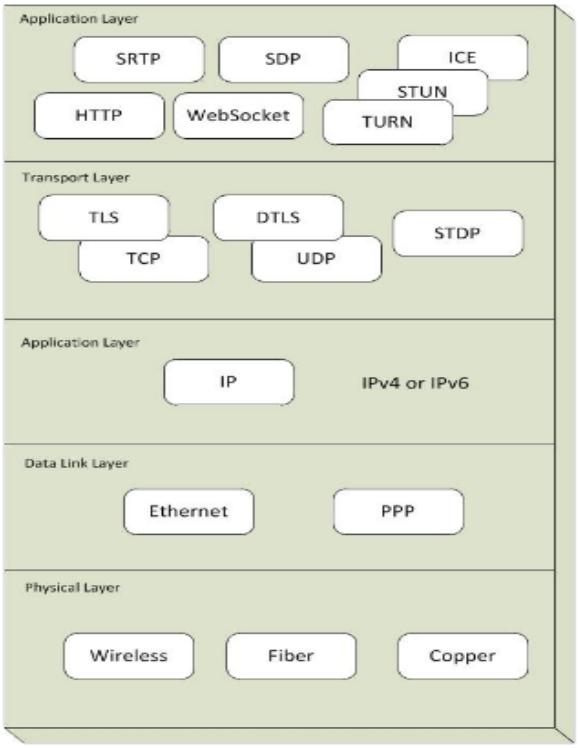

Figure 2.2: Protocol Stack

One Client will share its SDP to all other candidate who are in group. offer SDP is as shown in 2.1 This offer SDP will have mainly three Information.

1) Supporting Media Codec Information

2) Ice Candidate Information

3) Ice Username and Fingerprint

This offer will indicate capability of that client.

Other Clients who wants to do video and audio communication with that client will accept the request and reply answer SDP which indicates what media it supports and which codec group it supports.

To generate standard SDP media information needed to be collected from Media server.

To communicate with Media Server RTSP(Real Time Streaming Protocol) is used.

RTSP Singaling : RTSP is an application level protocol for control over the deliv-

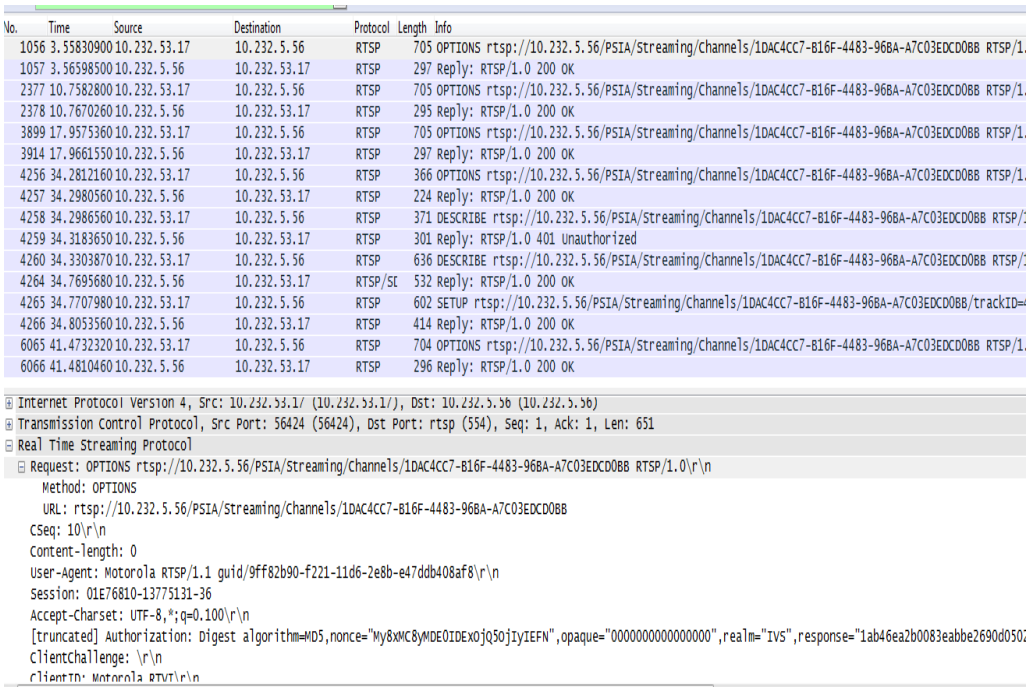

Figure 2.3: RTSP Signaling Result

ery of data with real-time properties.

RTSP-Streamer Server as a test server which will be installed in one machine.

RTSP request in standard format is sent and for each request response was received.

2.2 WebRTC

2.2.1 Protocol Stack

Normally WebRTC is using Protocol as usual except for Transport layer they will send across DTLS to provide some efficient security and performance of data.

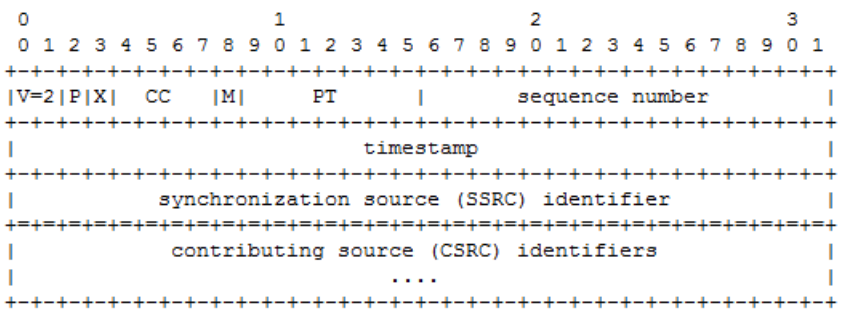

Figure 2.4: RTP Packet

2.2.2 Data Packet

RTP

RTP (Real-Time Transport Protocol) which provides end-to-end data with real-time characteristics,such as audio and video.

Information includes Payload type identification,sequence numbering,timestamping and delivery monitoring.

version (V): 2 bits

This field identifies the version of RTP. The version defined by this specification is two (2).

padding (P): 1 bit

If the padding bit is set, the packet contains one or more additional padding octets at the end which are not part of the payload.

CSRC count (CC): 4 bits

The CSRC count contains the number of CSRC identifiers that follow the fixed header.

marker (M): 1 bit

It indicates frame boundries.Each frame have multiple number of packets.To know when a frame ends marker bit is used.With help of marker bit we can count number of packets for a particular frame.

payload type (PT): 7 bits

Identifies format of RTP payload type.

specifies a default static mapping of payload type codes to payload formats.

sequence number: 16 bits

This act as a count,which increments by one for each RTP packet sent and used at receiver side to detect packet loss and to restore packet sequence.

The initial value of the sequence number is random (unpredictable) to make known-plaintext attacks on encryption more difficult.

timestamp: 32 bits

The timestamp reflects the sampling instant of the first octet in the RTP data packet.helpful for jitter and synchronization. To calculate frame rate time stamp is useful.

SSRC: 32 bits

SSRC identifies synchronization of source,within the same RTP session will have the same SSRC value.

RTCP

RTP Control Protocol

based on periodic retransmission of control packets to all participants.

RTCP functions 1) Provide feedback to the quality of data distribution. for example From the source of RTP data it will send every 2 seconds that it has transmitted 100 packets to receiver.Receiver suppose receives only 80 packets than it will send receiver report to the sender for 20%

2) CNAME It carries a persistent identifier called canonical name or CName,SSRC may change if conflicts occur then to keep track of each it is used.

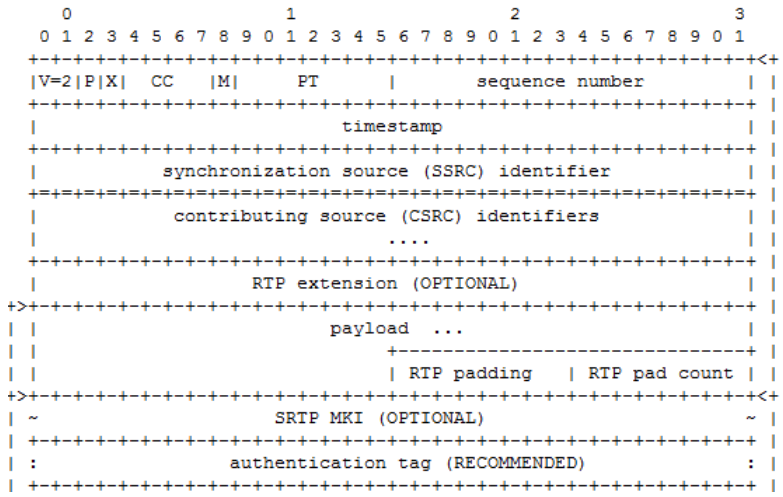

Figure 2.5: SRTP Format

RTCP packet format

SR: Sender Report

For transmission and reception statistics from active senders

RR: Receiver Report

For Reception statics,indicates Received packet information from participants who are not active senders.

SDES Source description identifier included in CNAME.

BYE indicates and of transmission.

SRTP and SRTCP

It is a profile for RTP which provides confidentiality,message authentication and Replay protection for RTP traffic.

Provides a framework for encryption and message authentication of RTP and RTCP streams.

Concept of Master and Salt Key is used. “Master Key ”provides keying material for confidentiality and integrity protection for SRTP and SRTCP. “Salt Key ”used to protect against pre-computation and time-memory tradeoff attacks.

Encrypted Portion will have encryption of the RTP payload of the equivalent RTP packet.

MKI :Master Key Identifier identifies the master key from which session keys were derived that authenticate or encrypt the particular packet.

Authentication Tag is used to carry message authentication data.

Chapter 3

Architecture and Trigger Applications

Video Analytics applications include Intrusion Detection, Vehicle Detection, Unattended Baggage Detection, Overcrowding Detection, People Count, Line Control and Counter Flow Detection.

Smart video solutions and advanced applications generate insight from multimedia interactions enabling immediate detection and improved management of critical issues. Video Analytics (VA) is a set of video analysis applications that analyze video streams arriving in real-time and track objects moving through camera scenes. We can specify the automatic and manual tasks that are performed when a VA trigger is activated, as well as specify the trigger priority. We can filter the alarms displayed in the Control application according to priority. We can set up output signals to be activated by associated triggers. We can then use the output signal to activate alarms or other devices. The system triggers alarms when the objects meet conditions set by the user. There are Counter Flow, Unattended Baggage, Vehicle, Overcrowding, Line and Intrusion triggers.

The capabilities of VA include ignoring the following outdoor events in the generation of alerts:

Rain and heavy rain Wind Snow Momentary abrupt lighting change Sunrise and sunset (continuous lighting change) Heavy, fast-moving clouds

We can specify the following perspective settings for each of the Video Analytics triggers:

Not effective ratio: The percentage of the overall area of all objects in the scene, relative to the ROI (non-ignored) area of the scene, for which the object tracking algorithm is not effective (most of the scene differs from the reference image).

Shadow Filter: The shadow filter eliminates objects that are created due to local illumination changes caused by cloud shadows.

Camera Stabilizer: The camera stabilizer handles small camera movements (allowing the algorithm to cope with offsets of up to 10 pixels of the current image relative to the background model).

Stationary Filter: The stationary motion filter prevents periodic motion from being interpreted as an object. This motion includes waves, tree leaves, flags, and so on.

Snow Filter: The snow filter considers snow-related parameters.

3.1 Video Analytic Architecture

Video surveillance systems typically include the following main components:

- Video cameras
- Network infrastructure
- Network Video Recorder
- Storage
- Video Analytics

Video Analytics can be implemented in three different configurations, which correlate to the evolution of the Video Analytics and surveillance technologies:

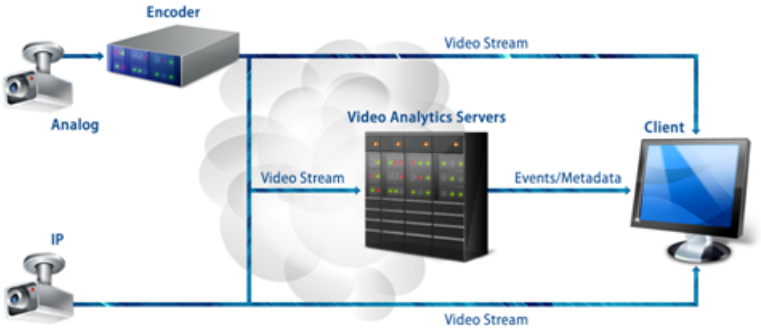

Figure 3.1: Server Based

3.1.1 Server Based Implementation

1.Server Based Implementation In this approach, the Video Analytics is implemented through a dedicated server that pulls the video, analyzes it, and issues the alerts or analysis results. This approach is independent of the video cameras, and therefore, is applicable to most types of surveillance systems. The main disadvantages to this approach are: The Video Analytics server requires the video to be transmitted to such server, and therefore causes an increase in network traffic load; The video quality being analyzed by the Video Analytics server is usually degraded due to compression and transmission effects, and therefore, the Video Analytics performance may be compromised; The Video Analytics server is limited by its processing power, and can typically handle no more than 16 cameras, with only limited Video Analytics functions, which makes it unattractive to large scale surveillance installations which deploy dozens or hundreds of cameras requiring a variety of Video Analytics functionalities.[15]

3.1.2 Edge Based Implementation

2.Edge Based Implementation In this approach, the Video Analytics is implemented through an IP video camera or video encoder, which must have sufficient processing power to run the Video Analytics functionality. On the surface, this ap-

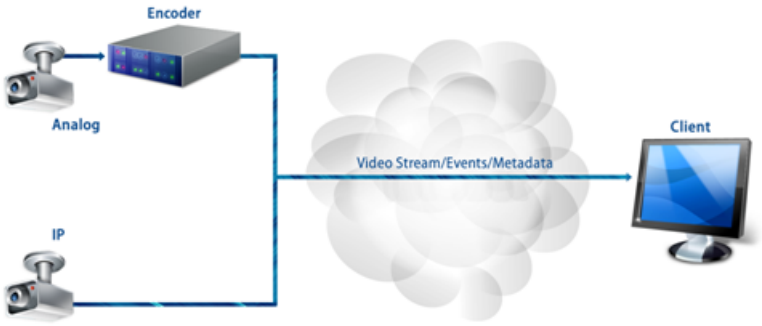

Figure 3.2: Edge Based

proach seems ideal, however it does not perform satisfactorily in many cases as it imposes limitations on the overall surveillance system design and performance. Most edge devices still lack sufficient processing power for high-end Video Analytics requirements, and therefore such implementation compromises on either the range of functions or performance quality of the Video Analytics, or both. In addition, most surveillance installations include different types of cameras, and not all cameras are suitable for edge based implementation nor do all cameras support it to the same quality.[15] At any given time task is in one of the five states as shown in the Figure.

3.2 Trigger Applications

Video Analytic Triggers:[14]

Unattended Baggage Detection Trigger The Unattended Baggage Detection application is designed to identify suspicious, unattended baggage in a given region of interest. It automatically triggers a visual and audio alarm of the suspicious object in question, pinpointing its exact location onscreen. Unattended Baggage Detection application triggers an alert when a bag is left unattended in the Activity zone (after a specified time). Objects moving in the Ignore zone are not tracked.

Vehicle Detection Trigger The Vehicle Detection application is designed to

identify suspicious vehicles illegal parked in perimeter security zones, stopped in an unauthorized zone or parked alongside a major traffic artery for a given period of time. It automatically triggers a visual alarm of the suspicious vehicle in question, pinpointing the exact location of a potential problem onscreen. The Vehicle Detection zone is defined by the user as an Activity zone. Objects moving in the Ignore zone are not tracked. To teach the system the perspective of the camera in the channel and the normal sizes of the tracked vehicles, the user draws lines on the image. Depending on the learning time the user sets, the system learns the normal state of the locale viewed through the channel and starts tracking moving vehicles. If the Recorder detects an unauthorized parking vehicle in the Activity zone, the system triggers an alert.

Counter Flow Video Analytics The Counter Flow application is designed to instantly identify a person or object moving in the wrong direction in a crowded security-critical location, and immediately alert security officers to the possible threat. It automatically triggers a visual alarm of the suspicious person or object in question, pinpointing the exact location on screen and immediately alerting security officers to the possible security threat. Officers can then resolve the incident as quickly as possible and proactively prevent a further escalation of the situation. To teach the system the normal flow of movement, the user creates an area of interest and forbidden flow directions and angles to detect movement and trigger alarms. If the Recorder detects a movement against a normal direction in the Activity zone, the system triggers an alert.

Line Control Trigger The Line Control application is designed to help regulating queues. Usually, when people stand in line, there is a specified queuing area. When the queue of people grows so that the queuing area is full, and the queue starts "overflowing" to nearby areas, interfering with the normal peoples flow of the site, the Line Control application generates an alert, allowing the sites authorities to manage the "overflowing" queue efficiently and within moments of the events occurrence. Line Control tracks the presence of people in a user-defined Interest area between the Queuing and Alerting zones and triggers a line control alert when the number of

people matches the criteria the user sets. Objects moving in the Ignore zone are not tracked. To teach the system the perspective of the camera in the channel, the user draws lines on the image. Depending on the learning time the user sets, the system learns the normal state of the locale viewed through the channel and starts tracking people in the video scene. If the Recorder detects a line crowding in the Alerting area, the system triggers an alert.

Intrusion Detection Trigger The Intrusion Detection application is designed for perimeter protection purposes. It can track objects at outdoor scenes, identify intruders or suspicious objects crossing into secured/restricted areas, within moments of the events occurrence. Intrusion Detection tracks a person or an object entering a user-defined Interest zone or passing from From to To zones. When this occurs, the Recorder triggers an intrusion alert. People or objects moving in the Ignore zone are not tracked. To teach the system the perspective of the camera in the channel and the normal sizes of people, the user draws lines on the image. Depending on the learning time the user sets, the system learns the normal state of the locale viewed through the channel. Then, if a person or object enters the user defined Interest area in front of the camera, the Recorder notes the change and triggers an alert.

Overcrowding Trigger The Overcrowding application is designed to prevent overcrowding in a defined Interest zone. Overcrowding tracks the presence of people in a user-defined Interest zone and alerts when the occupancy percentage in the zone matches the criteria set by the user. If the Recorder detects overcrowding in the Interest zone, the system triggers an alert.

Chapter 4

Video Analytics

4.1 Video Analytics Configuration

4.1.1 Network Video Recorder(NVR)Configuration

In NVR Select Trigger Application From Available Triggers

In Network Video Recorder (NVR) select an I.P. camera that has already been added.

For that camera select video analytic application that we want to execute.

In Setup Click on Video Analytic to configure camera.[12] Enable Video Analysis option and click on Setup. In Field of View we can set

1) Field of View : a. Ignore Area: Area for which we do not want to apply analytic algorithm. b. Learning Time: This is the time the channel learns the scene in front of the camera and uses it as basis for deciding if a person is a normal part of the scene or to trigger an alert.

2) Area of Interest : a. Based on Analytic Application Areas like From, To, Interest. Interest area is Activity zone.

3) Alert Properties : a. To Decide Maximum and Minimum object size which it should detect when object enters in activity zone.

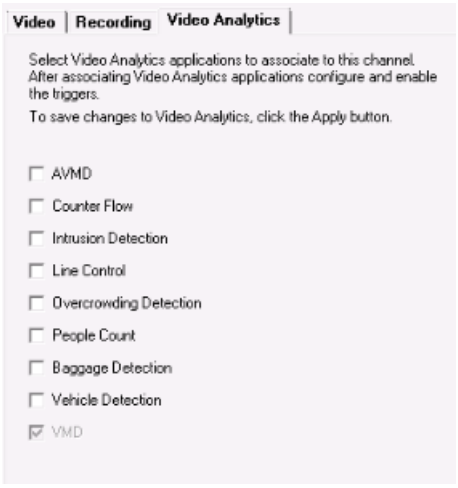

Figure 4.1: Trigger Lists

Figure 4.2: Area Of Interest

4.1.2 Video Analytic Application configuration

Video Analytic Application configuration Input File Location : File Location Of Predefined XML.

Output File Location : File Location where Trigger Details and Video Link will be attached.

NVR IP: Set Particular Network Video Recorder IP from which Trigger Data will

```
[InputFile]
input_file_location = C:\\VideoAnalytic\\Input.xml
[Link]
RicServerIp = http://10.232.15.102:8080
ChannelId = D1180B2E-392A-4775-B7D7-834A1D35C1D6
[OutputFile]
Output_file_location = C:\\VideoAnalytic\\Output.xml
[HttpPost]
RicServerIp = http://10.232.15.102:8081
[RecorderIp]
NvrIp = 10.232.15.128
[LocalIp]
MachineIp = 10.232.53.152
```

Figure 4.3: Application Configuration

come.

Event Server Server: Set Event server IP to which Http Posting will send xml file.

Camera id: Id of camera of which Recorded Video Link will be generated.

4.1.3 Trigger Detection

- Try To Connect To the NICE NVR (Network Video Recorder)
- If able to connect then proceed else terminate with error message.
- Registers the application so that it can receive alarm messages.
- If able to register then proceed else terminate with alarm message.
- start fetching alarm messages while connection exists
- close the stream connection
- disconnect from the recorder

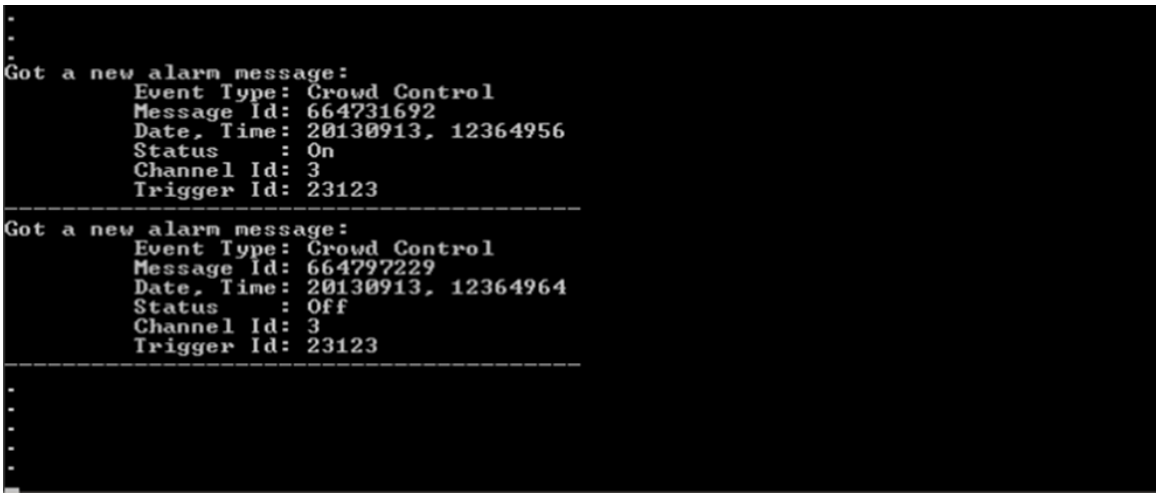

Figure 4.4: Crowd Control

4.1.4 Video Analytic Algorithm

Video Analytic Application configuration First from available list of supported trigger application select one for which basic configuration from Network Video Recorder will be done.

Then Configure Area of Interest for which analytic algorithm required to be applied. Analytic API will continuously monitor the AOI and will keep on checking any Object is there or not. If object will come then Start Event(Giving Object presence) and will save time,date,type of trigger information and will save in xml file.

When Object will move away from Area Of Interest(AOI) it will generate stop trigger.In that trigger also stop time,date,trigger id,channel id data we will save in xml file.

From Stop and Start time and channel id, Video Link will be generated which will be sent to Event Server and that link we can play with help of Streaming Server which will have time stamp of each and record it.The Video links will not be over written when new event comes, we will save it and padd other events to analyze video events after words if required.

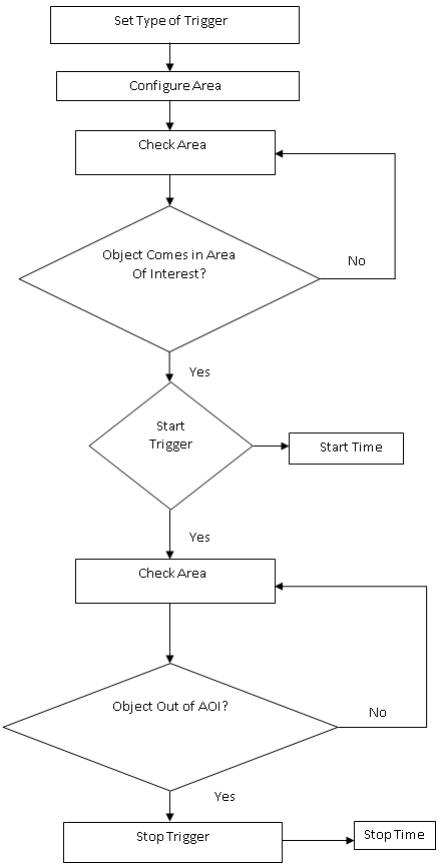

Figure 4.5: Video Analytic Algorithm

4.1.5 Event Posting

Get Trigger Details Of Both Start Trigger and Stop Trigger.

a. In Trigger Convert Date Time into Total Number Of Seconds From Standard Date Time i.e Total Seconds For Start Time = (Date Time Start.Event) - (01/01/0001 12:00:00 AM) Same thing For Stop Trigger.

b. Calculate Time Difference Time Difference = (Start Trigger).Total Seconds - (Stop Trigger).Total Second

c. Take Channel id of Particular Camera from which streams to play.

Figure 4.6: Block Diagram

d. Create Video Link from Channel Id and Total Seconds.

e. Post Events by Posting XML file to Event Server.

f. From Event Server it will be directed to Client.

4.2 Facial Recognition

4.2.1 Face Detection using Haar Cascades

4.2.2 Basics

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, **Rapid Object Detection using a Boosted Cascade of Simple Features** in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, haar features shown in below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.[7]

Now all possible sizes and locations of each kernel is used to calculate plenty of features. (Just imagine how much computation it needs? Even a 24x24 window results over 160000 features). For each feature calculation, find sum of pixels under white and black rectangles. To solve this,the integral images concept will be used. It

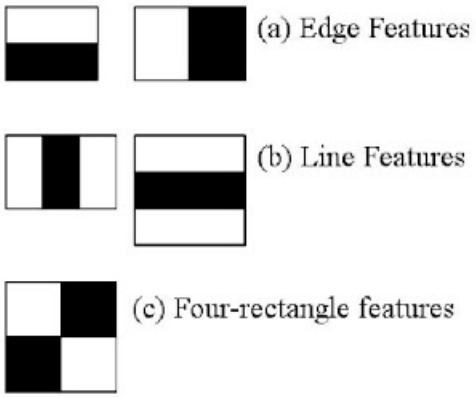

Figure 4.7: Haar Features

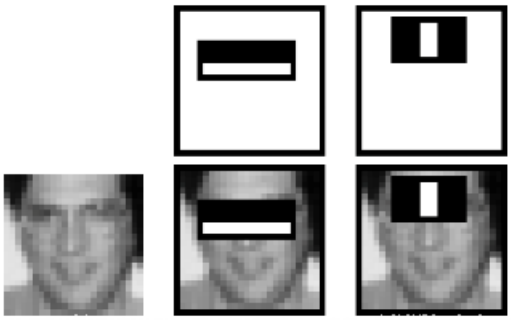

Figure 4.8: Features on Image

simplifies calculation of sum of pixels, how large may be the number of pixels, to an operation involving just four pixels.

But among all these features calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant. So how select the best features out of 160000+ features? It is achieved by **Adaboost**.[7]

For this,apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative.

But obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that best classifies the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then again same process is done. New error rates are calculated. Also new weights. The process is continued until required accuracy or error rate is achieved or required number of features are found).

Final classifier is a weighted sum of these weak classifiers. It is called weak because it alone cant classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95 percent accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features. That is a big gain).

In an image, most of the image region is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot. Dont process it again. Instead focus on region where there can be a face. This way, we can find more time to check a possible face region.

For this they introduced the concept of **Cascade of Classifiers**. Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one. (Normally first few stages will contain very less number of features). If a window fails the first stage, discard it. We dont consider remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region.

Authors detector had 6000+ features with 38 stages with 1, 10, 25, 25 and 50 features in first five stages. (Two features in the above image is actually obtained as the best two features from Adaboost).

4.2.3 Haar-cascade Detection in OpenCV

OpenCV comes with a trainer as well as detector. If you want to train your own classifier for any object like car, planes etc. you can use OpenCV to create one. Its full details are given here: Cascade Classifier Training.[11]

OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in opencv/data/haarcascades/ folder.

First load the required XML classifiers. Then load our input image (or video) in grayscale mode

import numpy as np import cv2 $face_cascade = cv2.CascadeClassifier('haarcascade_frontalf$
$eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')$ $img = cv2.imread('sachin.jpg')$
$gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)$
Now find the faces in the image. If faces are found, it returns the positions of detected faces as Rect(x,y,w,h). after getting these locations, create a ROI for the face and apply eye detection on this ROI .
$faces = face_cascade.detectMultiScale(gray, 1.3, 5)$
for (x,y,w,h) in faces:
$img = cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)$ $roi_gray = gray[y :$
$y + h, x : x + w]$
$roi_color = img[y : y + h, x : x + w]$
$eyes = eye_cascade.detectMultiScale(roi_gray)$
for (ex,ey,ew,eh) in eyes:
$cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)$ $cv2.imshow('img', img)$
$cv2.waitKey(0)$
$cv2.destroyAllWindows()$

4.2.4 Image Preprocessing

Applying face recognition directly on a normal photo image, will give probably less than 10 percent accuracy!

It is extremely important to apply various image pre-processing techniques to standardize the images required for a face recognition system.Most face recognition algorithms are extremely sensitive to lighting conditions, so that if it was trained to recognize a person when they are in a dark room, it probably wont recognize them in a bright room, etc. This problem is referred to as **"lumination dependent"**, and there are also many other issues, such as the face should also be in a very consistent position within the images (such as the eyes being in the same pixel coordinates), consistent size, rotation angle, hair and makeup, emotion (smiling, angry, etc), position of lights (to the left or above, etc). This is why it is so important to use a good image preprocessing filters before applying face recognition. For that preprocessing like removing the pixels around the face that aren't used, such as with an elliptical mask to only show the inner face region, not the hair and image background, since they change more than the face does. For simplicity, the face recognition system given below is Eigenfaces using greyscale images. convert color images to greyscale (also called 'grayscale'), and then easily apply Histogram Equalization as a very simple method of automatically standardizing the brightness and contrast of your facial images. For better results, you could use color face recognition (ideally with color histogram fitting in HSV or another color space instead of RGB), or apply more processing stages such as edge enhancement, contour detection, motion detection, etc.

Given Below is example of Preprocessing stage:[21]

- Either convert image to grayscale or use existing Grayscale Image Mat gray; $cvtColor(original, gray, CV_BGR2GRAY)$;

- Resize the image to be a consistent size, even if the aspect ratio changes.

Figure 4.9: Preprocessing Stages

```
1  E:\facerecogniton\amit/amit (1).jpg;0
2  E:\facerecogniton\amit/amit (2).jpg;0
3  E:\facerecogniton\amit/amit (3).jpg;0
4  E:\facerecogniton\amit/amit (4).jpg;0
5  E:\facerecogniton\kalpana/kalpana (1).jpg;1
6  E:\facerecogniton\kalpana/kalpana (2).jpg;1
7  E:\facerecogniton\kalpana/kalpana (3).jpg;1
8  E:\facerecogniton\kalpana/kalpana (4).jpg;1
9  E:\facerecogniton\Kaushik/kaushik (1).jpg;2
10 E:\facerecogniton\Kaushik/kaushik (2).jpg;2
11 E:\facerecogniton\Kaushik/kaushik (3).jpg;2
12 E:\facerecogniton\Kaushik/kaushik (4).jpg;2
```

Figure 4.10: Image DataBase

```
Mat face_resized;
cv::resize(face, face_resized, Size(im_width,
 im_height), 1.0, 1.0, INTER_CUBIC);
```

- Give the image a standard brightness and contrast.
  $cvEqualizeHist(imageProcessed, imageProcessed)$;

4.2.5 DataBase

Create DataBase

To create DataBase for face recognition ,create a text file that lists the image files and which person each image represents. For example Create Text file 4image3person.txt and content of that text file will look like below. In this text file after semicolon represents label that we will give for each person.As shown For Amit Label is 0, for Kalpana label is 1 and for kaushik label is 2. this will be helpful in recognition while applying prediction.

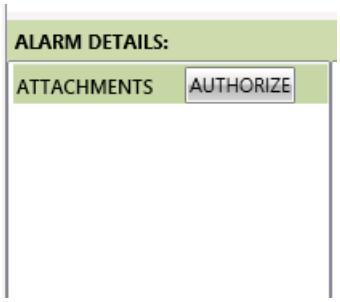

Figure 4.11: Authorize

Read DataBase

$vector < Mat > images$;

$vector < int > labels$;

$read_csv(fn_csv, images, labels)$;

fn_csv is the text file name in which location of each image and labels are given. From this in images vector all images data will be written and labels will have data 0,1,2. . . .

Train Database by applying FaceRecognition Algorithm Create a FaceRecognizer and train it on the given images:

$Ptr < FaceRecognizer > model = createFisherFaceRecognizer()$;

$model- > train(images, labels)$;

4.3 Start Recognition

In Client one button to authorize person will be there. When user press this button http request will be sent to start face recognition process. In Http Request String = "Start Face Recognition" will be sent

4.3.1 Server Listenning

One Server will be there which will wait for some request to come on one port. When request comes it will capture string and compare it with predefined string. When

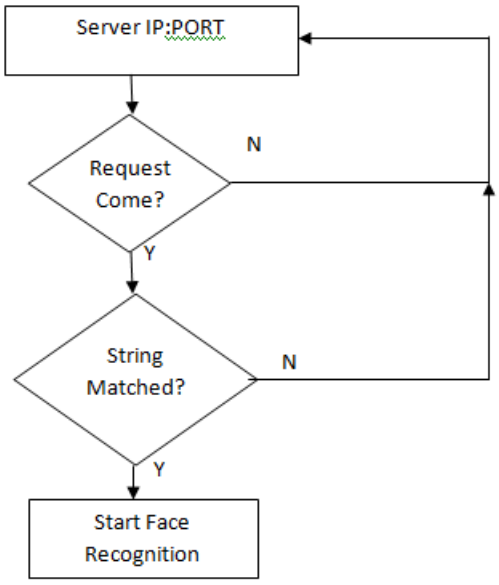

Figure 4.12: Server Algorithm

Matches it will do one asynchronous call to Face Recognizer to Start Face recognition.

4.3.2 Face Recognition

When Request comes to Server it will capture video for example 20seconds. From that video clip it will capture each frame and will compare it with the database images by applying face recognition algorithm. This will happen in two steps.

- **Face Detection** From Each frame first it will apply face detection algorithm, which will detect number of faces in each frame.
- **Face Recognize** After detection of each face from each frame it will apply face recognition algorithm which will compare eigenvector and eigenvalues of each face with database and will apply prediction.

Chapter 5

Hardware and Software Tools

1) Surveillance Cameras
2) Network Video Recorder(NVR)
3) Media Server
4) WebRtc Gateway
5) WebRtc Client

5.1 Surveillance Cameras

Surveillance cameras such as Sony RX550 and AXIS cameras.Camera supports PTZ(Pann-Till-Zoom).

5.2 Network Video Recorder

A network video recorder (NVR) is a software program that records video in a digital format to a disk drive, USB flash drive, SD memory card or other mass storage device.

5.3 Media Server

Streaming Server includes two main part

- **RTSP Server:**
  It Responds to RTSP Requests such as OPEN,PLAY,PAUSE,STOP,DESCRIBE,OPTIONS Requests.
- **RTP Server:** It Streams RTP and RTCP Packets.

5.4 WebRTC Gateway

WebRTC Gateway is intermediate between Media Server and Webrtc Client. Gateway Does Handshake Signaling with Client and Server and then route RTP and RTCP Packets From Media Server To Client. Signaling includes WebSocket,Ice,Stun,SDP,DTLS-SRTP Signaling. Data includes RTP and RTCP Packet Routing. RTP packets From Streaming Server to WebRTC Client. RTCP Packets From Streaming Server to WebRTC Client and From WebRTC Client to Streaming Server.

5.5 WebRTC Client

WebRTC Client is Browser.Support for only Firefox and Nightly Browser is Given as of now. WebRTC CLient From UserInterface prespective includes Camera List, Video Play Section,PTZ Section(Pan/Till/Zoom). WebRTC Client Supports 16up cameras playing together. WebRTC Client has it's own Requirements.

- Supports WebSocket Signaling
- Supports VP8 Codec
- Can Play only SRTP Packets
- DTLS-SRTP Handshaking is required

Chapter 6

WebRTC

6.1 WebRTC Gateway

- RTSP Signaling
- WebSocket Signaling
- ICE-STUN Signaling
- DTLS-SRTP Signaling

6.1.1 RTSP Signaling

RTSP(Real time Streaming Protocol) establishes and controls streams of media such as audio and video. RTSP acts as a "network remote control" for multimedia servers. Each presentation and media stream may be identified by an RTSP URL.

RTSP URL :

rtsp url = rtsp://host:port/absolutepath

host = ¡A legal Internet host domain name of IP address"

port = *DIGIT

Example : rtsp://media.example.com:554/twister/audiotrack

The Real Time Streaming Protocol (RTSP) is a network control protocol designed for use in entertainment and communications systems to control streaming media servers. The protocol is used for establishing and controlling media sessions between end points. Clients of media servers issue VCR-style commands, such as play and pause, to facilitate real-time control of playback of media files from the server.

The transmission of streaming data itself is not a task of the RTSP protocol. Most RTSP servers use the Real-time Transport Protocol (RTP) in conjunction with Real-time Control Protocol (RTCP) for media stream delivery, however some vendors implement proprietary transport protocols.

While similar in some ways to HTTP, RTSP defines control sequences useful in controlling multimedia playback. While HTTP is stateless, RTSP has state; an identifier is used when needed to track concurrent sessions. Like HTTP, RTSP uses TCP to maintain an end-to-end connection and, while most RTSP control messages are sent by the client to the server, some commands travel in the other direction (i.e. from server to client).

- **OPTIONS :**

  An OPTIONS request returns the request types the server will accept.

  C->S: OPTIONS rtsp://10.232.5.56/PSIA/Streaming/Channels/1DAC4CC7-B16F-4483-96BA-A7C03EDCD0BB
  CSeq: 1
  Session: 01E76810-13775131-36
  Accept-Charset : UTF-8,q=0.100 ClientID: Motorola_ RTVI
  Require: implicit-play
  Proxy-Require: gzipped-messages

  S->C: RTSP/1.0 200 OK CSeq: 1 Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE

- **DESCRIBE :** A DESCRIBE request includes an RTSP URL (rtsp://...), and the type of reply data that can be handled. The default port for the RTSP

protocol is 554 for both UDP and TCP transports. This reply includes the presentation description, typically in Session Description Protocol (SDP) format. Among other things, the presentation description lists the media streams controlled with the aggregate URL. In the typical case, there is one media stream each for audio and video

C->S: DESCRIBE rtsp://example.com/media.mp4 RTSP/1.0 CSeq: 2

S->C: RTSP/1.0 200 OK
CSeq: 3
Content-Base: rtsp://10.232.5.56/PSIA/Streaming/Channels/1DAC4CC7-B16F-4483-96BA-A7C03EDCD0BB/
Content-type: application/sdp
Server: MOTO Streaming Server Version 0.1
Session Description Protocol
Session Description Protocol Version (v): 0
Owner/Creator, Session Id (o): 12167 2372476 55 IN IP4 10.232.5.56
Session Name (s): RTSP Session
Connection Information (c): IN IP4 10.232.5.56
Time Description, active time (t): 0 0
Session Attribute (a): range:clock=20140310T061937.020Z-
Media Description, name and address (m): video 0 RTP/AVP 120
Media Attribute (a): rtpmap:120 VP8/90000
Media Attribute (a): control:trackID=4
Media Attribute (a): cliprect:0,0,480,640
Media Attribute (a): framesize:120 640-480
Media Attribute (a): framerate:16

- **SETUP:** A SETUP request specifies how a single media stream must be transported. This must be done before a PLAY request is sent. The request con-

tains the media stream URL and a transport specifier. This specifier typically includes a local port for receiving RTP data (audio or video), and another for RTCP data (meta information). The server reply usually confirms the chosen parameters, and fills in the missing parts, such as the server's chosen ports. Each media stream must be configured using SETUP before an aggregate play request may be sent.

C->S: SETUP rtsp://example.com/media.mp4/streamid=0 RTSP/1.0 CSeq: 3

S->C: RTSP/1.0 200 OK CSeq: 3 Transport: RTP/AVP;unicast;client_port=8000-8001;server_port=9000-9001 Session: 12345678

- **PLAY:** A PLAY request will cause one or all media streams to be played. Play requests can be stacked by sending multiple PLAY requests. The URL may be the aggregate URL (to play all media streams), or a single media stream URL (to play only that stream). A range can be specified. If no range is specified, the stream is played from the beginning and plays to the end, or, if the stream is paused, it is resumed at the point it was paused.

  C->S: PLAY rtsp://example.com/media.mp4 RTSP/1.0
  CSeq: 4
  Range: npt=5-20
  Session: 12345678

  S->C: RTSP/1.0 200 OK CSeq: 4
  Session: 12345678
  RTP-Info: url=rtsp://example.com/media.mp4/streamid=0;seq=9810092;rtptime=345001

- **PAUSE:** A PAUSE request temporarily halts one or all media streams, so it can later be resumed with a PLAY request. The request contains an aggregate or media stream URL. A range parameter on a PAUSE request specifies when

to pause. When the range parameter is omitted, the pause occurs immediately and indefinitely.

C->S: PAUSE rtsp://example.com/media.mp4 RTSP/1.0 CSeq: 5
Session: 12345678

S->C: RTSP/1.0 200 OK CSeq: 5
Session: 12345678

- **TEARDOWN:** A TEARDOWN request is used to terminate the session. It stops all media streams and frees all session related data on the server.

C->S: TEARDOWN rtsp://example.com/media.mp4 RTSP/1.0
CSeq: 8
Session: 12345678

S->C: RTSP/1.0 200 OK
CSeq: 8

6.1.2 WebSocket Signaling

WebSocket is a protocol providing full-duplex communications channels over a single TCP connection. WebSocket is designed to be implemented in web browsers and web servers, but it can be used by any client or server application. The WebSocket Protocol is an independent TCP-based protocol. Its only relationship to HTTP is that its handshake is interpreted by HTTP servers as an Upgrade request.[1]

A simpler solution would be to use a single TCP connection for traffic in both directions. This is what the WebSocket Protocol provides.

6.1.3 Ice-Stun Signaling

In order to establish a peer-to-peer connection, by definition, the peers must be able to route packets to each other. A trivial statement on the surface, but hard to achieve in practice due to the numerous layers of firewalls and NAT devices between most peers.

Consider the trivial case, where both peers are located on the same internal network, and there are no firewalls or NATs between them. To establish the connection, each peer can simply query its operating system for its IP address (or multiple, if there are multiple network interfaces), append the provided IP and port tuples to the generated SDP strings, and forward it to the other peer. Once the SDP exchange is complete, both peers can initiate a direct peer-to-peer connection.

[3] The basic idea behind ICE is as follows: each agent has a variety of candidate TRANSPORT ADDRESSES (combination of IP address and port for a particular transport protocol) it could use to communicate with the other agent. These might include:

- A transport address on a directly attached network interface.
- A translated transport address on the public side of a NAT ("server reflexive").
- A transport address allocated from a TURN server ("relayed address").

Potentially any of the candidate should be able to communicate with other candidate,but practically many combinations will not work if both are behind the Network address translators(NAT). The purpose of ICE is to discover which pairs of addresses will work.

Gathering Candidate Addresses

A CANDIDATE is a transport address – a combination of IP and Port. two agents want to communicate. If both are on the same private network they can communicate with what is known as HOST candidates. Agents uses STUN or TURN to obtain

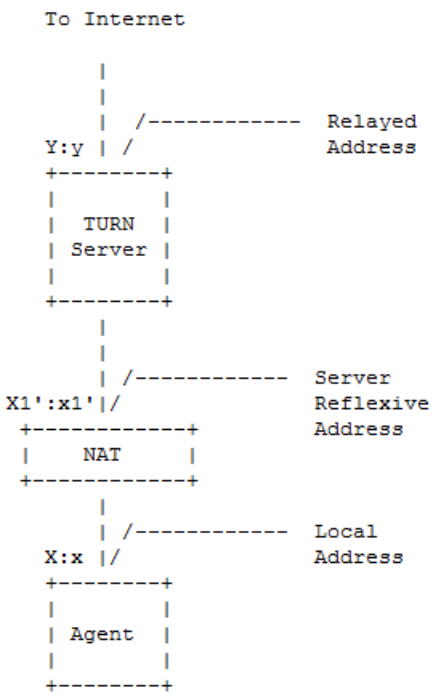

Figure 6.1: Candidate Relationships [5]

additional candidates. translated addresses on the public side of a NAT and addresses on TURN servers.

Connectivity Check

Once agent has gathered all its candidates, it orders them in highest to lowest priority and sends them to other agent. candidates are carried in attributes in SDP offer. when other agent receives the offer it does the same process and responds with its own list.At the end each agent has a complete list of both its candidates and peer's candidates.

To see which pair will work each agent performs a series of checks. Basic principle of connectivity check is 1) Sort candidate pairs in priority order 2) Send checks on each candidate pair in priority order. 3) Acknowledge checks received from the other agent.

STUN Binding request is used for the connectivity check,the STUN Binding re-

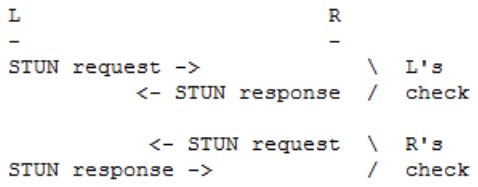

Figure 6.2: Connectivity Check [3]

sponse will contain the agent's translated transport address on the public side of any NATs between the agent and its peer.

Sorting Candidates

To produce faster results, candidates are sorted in a specified order,this list is called CHECKLIST. this follows two general principles:

- Each agent gives its candidates a numeric priority,which is sent along with a candidate to peer.
- Local and remote priorities are combined to have same ordering for the candidate pairs.

ICE checks in each direction will not succeed until both sides have sent a check through their respective NATs.[3]

Concluding ICE

Ice checks are performed in a specific sequence,High Priority candidate pairs are checked first, followed by low priority candidates.[3] ICE assigns one of the agent a role of controlling agent and the other controlled agent.the controlling agent gets to nominate which candidate pairs will get used for media amongst the ones that are valid. It can do this in two ways

1) REGULAR NOMINATION In this the controlling agents cheecks one valid candidate pair for each media stream, then it pecks amongst those that are valid and send a second STUN request on its nominated candidate pair, but this time a flag set

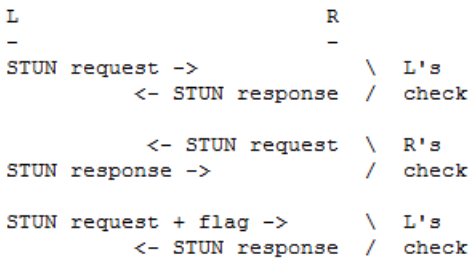

Figure 6.3: Regular Nomination

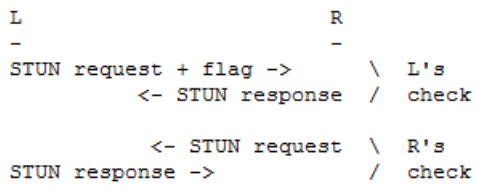

Figure 6.4: Aggressive Nomination

to tell peer that this pair has been nominated.

2) AGGRESSIVE NOMINATION controlling agent puts the flag in every STUN request it sends.once the first check succeeds, ICE processing is complete for that media stream and the controlling agent doesn't have to send a second STUN request. The selected pair will be the highest-priority valid pair whose check succeeded. Aggressive nomination is faster than regular nomination,but gives less flexibility.

6.1.4 DTLS-SRTP Signaling

DTLS(Datagram Transport Layer Security) extension to establish keys for Secure RTP(SRTP) and Secure RTCP(SRTCP).

Secure RTP Profile(SRTP) provides confidentiality,message authentication and replay protection to RTP data and RTCP traffic.SRTP does not provide key management functionality,but instead depends on external key management to exchange secret master keys,and to negotiate the algorithms and parameters for use with those keys.

Datagram Transport Layer Security (DTLS) is a channel security protocol that

offers integrated key management,parameter negotiation, and secure data transfer.[1]

Key Points for DTLS-SRTP

- Application Data is Protected using SRTP
- DTLS Handshake is used to establish keying material,algorithms ,and parameter for SRTP,
- DTLS extension is used to negotiate SRTP algorithm

DTLS-SRTP operation : DTLS-SRTP is defined for point to point media sessions,which includes exactly two participants. Each DTLS-SRTP session contains one DTLS Session and either two SRTP context if media flows in both direction or one context if media flows in single direction.

For each RTP or RTCP flow the peers do a DTLS handshake on the same source and destination port pair to establish a DTLS association. Between the two points which side is DTLS client and which side is DTLS Server must be decided by some mechanism.Keying material from that handshake is fed into the SRTP stack. once that part is done RTP packets are secured SRTP using that keying material.

The SRTP keys used to protect packets originated by the client are distinct from the SRTP keys used to protect packets originated by the server.[4]

DTLS Extensions for SRTP Key Establishment

"The use_ srtp Extension"

In order to negotiate the use of SRTP data protection, clients includes an extension of type "use_ srtp" in the DTLS extended client hello. Server that receives extended "use_ srtp" extension can agree to use SRTP by including an extension of type "use_ srtp", with the chosen protection profile in the extended server Hello.[1]

SRTP Protection Profiles:

DTLS-SRTP Protection profiles defines the parameters and options that are in effect for the SRTP processing.

SRTPProtectionProfile SRTP_ AES128_ CM_ HMAC_ SHA1_ 80 = 0x00, 0x01;

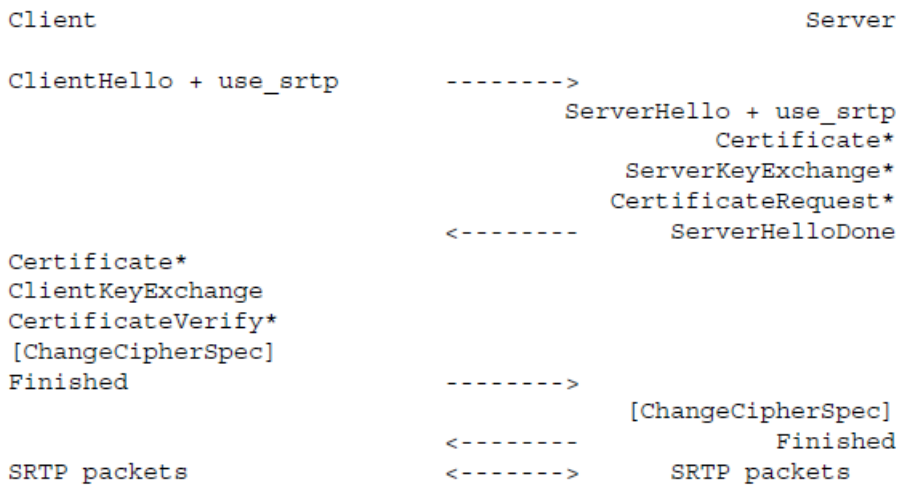

Figure 6.5: Preprocessing Stages

SRTPProtectionProfile SRTP_ AES128_ CM_ HMAC_ SHA1_ 32 = 0x00, 0x02;

SRTPProtectionProfile SRTP_ NULL_ HMAC_ SHA1_ 80 = 0x00, 0x05;

SRTPProtectionProfile SRTP_ NULL_ HMAC_ SHA1_ 32 = 0x00, 0x06;

SRTP_ AES128_ CM_ HMAC_ SHA1_ 80

cipher: AES_ 128_ CM

cipher_ key_ length: 128

cipher_ salt_ length: 112

maximum_ lifetime: 2$\hat{}$31

auth_ function: HMAC-SHA1

auth_ key_ length: 160

auth_ tag_ length: 80

SRTP_ AES128_ CM_ HMAC_ SHA1_ 32

cipher: AES_ 128_ CM

cipher_ key_ length: 128

cipher_ salt_ length: 112

maximum_ lifetime: 2$\hat{}$31

auth_ function: HMAC-SHA1

auth_ key_ length: 160

auth_ tag_ length: 32

RTCP auth_ tag_ length: 80

SRTP_ NULL_ HMAC_ SHA1_ 80

cipher: NULL

cipher_ key_ length: 0

cipher_ salt_ length: 0

maximum_ lifetime: 2$\hat{3}$1

auth_ function: HMAC-SHA1

auth_ key_ length: 160

auth_ tag_ length: 80

SRTP_ NULL_ HMAC_ SHA1_ 32

cipher: NULL

cipher_ key_ length: 0

cipher_ salt_ length: 0

maximum_ lifetime: 2$\hat{3}$1

auth_ function: HMAC-SHA1

auth_ key_ length: 160

auth_ tag_ length: 32

RTCP auth_ tag_ length: 80

Exporting Keying Material:

When SRTP mode is in effect, different keys are used for ordinary DTLS record protection and SRTP packet protection. These keys are generated using a TLS exporter [RFC5705] to generate

2 * (SRTPSecurityParams.master_ key_ len + SRTPSecurityParams.master_ salt_ len) bytes of data

which are assigned as shown below.

client_ write_ SRTP_ master_ key[SRTPSecurityParams.master_ key_ len];

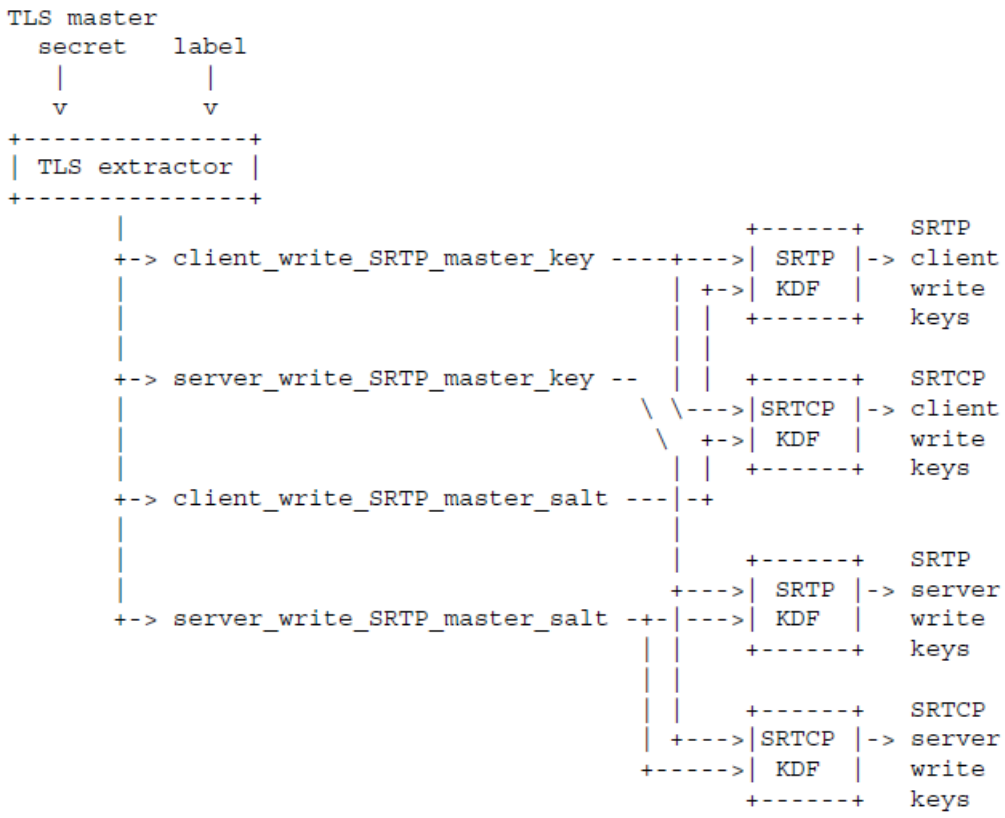

Figure 6.6: DTLS Key Extraction

server_ write_ SRTP_ master_ key[SRTPSecurityParams.master_ key_ len];

client_ write_ SRTP_ master_ salt[SRTPSecurityParams.master_ salt_ len];

server_ write_ SRTP_ master_ salt[SRTPSecurityParams.master_ salt_ len];

6.2 WebRTC Working

6.2.1 Block Diagram

- **Surveillance camera :** Streaming from Multiple Surveillance cameras will come to Network Video Recorder.Cameras will be IP cameras.

- **Network Video Recorder :** In Network Video Recorder add different cam-

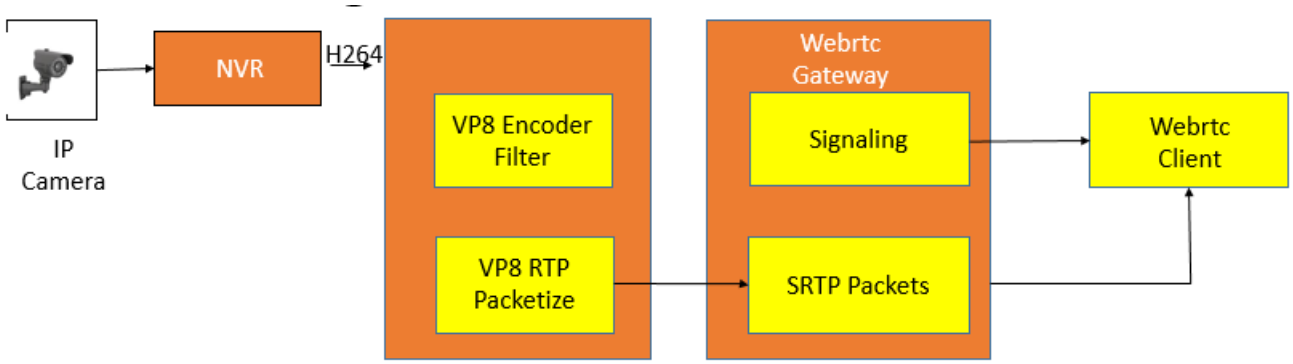

Figure 6.7: webrtcblock

eras.Do codec configuration and set resolution.

Streams from IP cameras will come to NVR and from that to Media Server,frames will be of h264 format.

- **Media Server:** Media Server is divided in two parts. **RTSP Server :** Give Response to RTSP Requests coming from different clients.

  RTP Server : Stream RTP Packets in H264 format.

  Support is added for vp8 format because webrtc Client Supports VP8 codec only.

- **WebRTC Gateway:** Gateway is divided in two parts.

  1) Signaling

  2) Media

  Signaling : In this support is added for RTSP,Ice-Stun,DTLS-SRTP signaling.

  RTSP Signaling for requesting particular channel(camera) to play,pause,teardown.

  ICE-STUN signaling for Network Address Translator(NAT) traversal,Client may or may not be in the same network as Server. DTLS-SRTP is used to encrypt RTP and RTCP Packets,this is required because WebRTC Client supports only SRTP and SRTCP packets only.To Secure Packets DTLS negotiation will happen to export keying material and to import key into SRTP stack.

  Media : For each packet coming from RTP Server it will apply SRTP protect

algorithm to Protect it and Unprotect it.

6.2.2 Client-Gateway Handshake

WebRTC Client will have Channel List(Camera List) to Play or Stop particular camera. When clicked play it will send a Websocket Message in Json Format to Gateway. Request Message consist of UserId, RtspUrl,Username and Password.

- UserId : As an Identifier from which client request came.
- RtspUrl : Url of the camera which client wants to play
- Username and Password : To Protect from anonymous user.

Gateway receives this message and decodes it and send a response if correct data received. Response message consist of Participation request and UserId of Gateway.

- Participation request : boolean data type- true if request is correct.
- UserId : Id of gateway for further communication.

Client on Receiving Participation as True will gather its Media Information,Transport Information and Session Description and send this as Session Description Protocol(SDP).
SDP for Example is shown below.
o=Mozilla-SIPUA-28.0 18985 0 IN IP4 0.0.0.0
s=SIP Call
t=0 0
a=ice-ufrag:5c30f425
a=ice-pwd:20e9d214ba7ae8665757903b33cff0b9
a=fingerprint:sha-256 E2:75:C8:9E:78:79:28:5B:49:F6:69:91:45:0E:A0:52:B9:45:81:36:FE:AA:68:2
m=audio 50978 RTP/SAVPF 109 0 8 101

c=IN IP4 192.168.2.109
a=rtpmap:109 opus/48000/2
a=ptime:20
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=sendrecv
a=setup:actpass
a=candidate:0 1 UDP 2128609535 192.168.2.109 50978 typ host
a=candidate:0 2 UDP 2128609534 192.168.2.109 50979 typ host
a=rtcp-mux
m=video 50980 RTP/SAVPF 120
c=IN IP4 192.168.2.109
a=rtpmap:120 VP8/90000
a=sendrecv
a=rtcp-fb:120 nack
a=rtcp-fb:120 nack pli
a=candidate:0 1 UDP 2128609535 192.168.2.109 50980 typ host
a=candidate:0 2 UDP 2128609534 192.168.2.109 50981 typ host

Sdp contains ICE information,Fingerprint, media information,Transport Information.[2]

- ICE Info : Ice Username , Ice Password and Candidate information for each media type
- Media Info : Audio and Video , Port on which media will be received, and supporting codec information. audio or video attribute such as sendrecv , send or recvonly.
- Transport Info : Protocol such as UDP or TCP , IP and Port and type of

candidate such as host,reflexcive.

Client may reside in same private network as Server or may be behind NAT. For Connectivity-Check STUN negotiation happens which will decide which transport address will be used for media streaming.
WebRTC Client support only VP8 Codec and Secured RTP(SRTP) packets.
For that DTLS-SRTP negotiation is required to export Keying material to protect RTP packets received from Media Server. In DTLS-SRTP signaling client server will exchange certificate and Keys to protect media. For each type of media separate handshake is required and media from client will use different key from the one that server will use.With Keys, Protection Profiles are decided which each side will include in SRTP Stack.

Gateway will receive RTP packets from RTP Server of Media Server and will convert it into Secured RTP(SRTP) packets and route it to Client which will unprotect it and will play it. How RTP Packets will come from Media Server to Gateway is explained in the next section.

6.2.3 Gateway-Media Server

RTSP Signaling Happens between Gateway and RTSP Server of Media Server.
This Signaling consists of series of Requests and Responses which are shown as below.

As shown in figure OPTIONS as a request is sent to RTSP Server with rtsp://url This request is sent to know functionality that particular RTSP Server supports. In Response we will get "200 OK" as an Indication that OPTION Request is successfully processed. with 200OK it will give functionality it supports. for example : DESCRIBE,GET_ PARAMETER,OPTIONS,PAUSE,PLAY,SETUP,TEARDOWN.

- DESCRIBE : To Know Media Information and codec information
- PLAY,PAUSE,STOP : To start streaming of data , to temporary pause packets to come or to stop streaming to come.

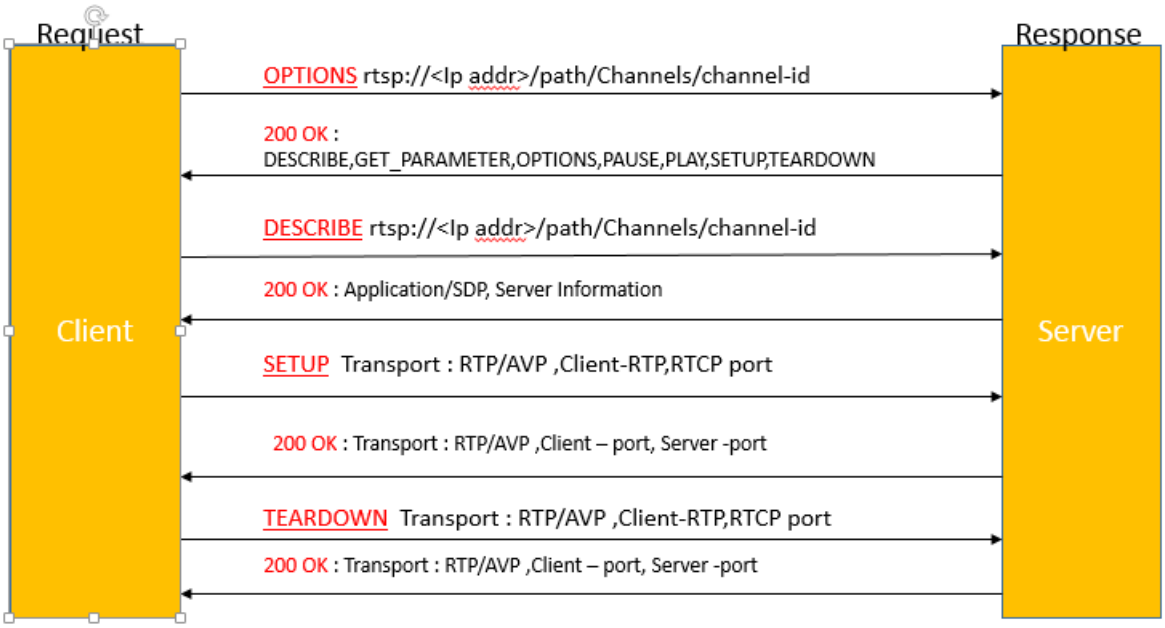

Figure 6.8: RTSP Signaling

- SETUP : Before Play Request it is required,and to inform server Port information on which it can receive rtp and rtcp packets and Transport Protocol information.

- TEARDOWN : To Break connection.Indication of Release Connection.

6.2.4 Relay SRTP and SRTCP Packets

After Setup and Play RTSP requests RTP and RTCP packets it will receive from Media Server. In DTLS handshake both agents will exchange Certificate and will use Public Key used in SDP exchange to unlock Certificate and to exctract private key which will be used for Securing RTP and RTCP Packets.

With help of standard SRTP library RTP and RTCP packet protection and unprotection is done. SSL algorithm is used for that.

An RTP session is defined by a pair of destination transport addresses, that is, a network address plus a pair of UDP ports for RTP and RTCP.

RTCP,the RTP control protocol, is used to coordinate between the participants in an

RTP session, e.g. to provide feedback from receivers to senders.
An SRTP session is similarly defined; it is just an RTP session for which the SRTP profile is being used. An SRTP session consists of the traffic sent to the SRTP or SRTCP destination transport addresses.
Each participant in a session is identified by a synchronization source (SSRC) identifier. Some participants may not send any SRTP traffic; they are called receivers, even though they send out SRTCP traffic, such as receiver reports.
RTP allows multiple sources to send RTP and RTCP traffic during the same session. The synchronization source identifier (SSRC) is used to distinguish these sources. In libSRTP, we call the SRTP and SRTCP traffic from a particular source a stream.
Each stream has its own SSRC, sequence number, rollover counter, and other data. A particular choice of options, cryptographic mechanisms, and keys is called a policy. Each stream within a session can have a distinct policy applied to it. A session policy is a collection of stream policies.A single policy can be used for all of the streams in a given session, though the case in which a single key is shared across multiple streams requires care. When key sharing is used, the SSRC values that identify the streams must be distinct. This requirement can be enforced by using the convention that each SRTP and SRTCP key is used for encryption by only a single sender.
In other words, the key is shared only across streams that originate from a particular device. libSRTP supports this enforcement by detecting the case in which a key is used for both inbound and outbound data.

LibSRTP Implementation

To Protect RTP and RTCP Packets srtp_ protect() function and to Unprotect srtp_ unprotect function is used. A session is created using the function srtp_ create(), the policy to be implemented and ssrc value for particular stream, key to protect stream are passed as part of policy as structure. srtp_ policy structure contains two crypto_ policy structure,one for RTP and other for RTCP. crypto_ policy_ rtp_ default() is used if want to set default policy,same thing applies for rtcp.

Chapter 7

Experiments and Results

In this chapter result analysis is discussed. For WebRTC Project Data Received on Client including DTLS handshake and algorithm used for Protection is shown. For which condition Client was able to unprotect SRTP and SRTCP is discussed. Through this experimental results able to debug actual flow of project and error received which is needed to debug.

7.1 Results

Two analysis is done one for Signaling and other for Media section.

- Signaling analysis Ice Candidate information received on client side.
  trickling candidate candidate:0 1 UDP 2128609535 10.8.115.201 65503 typ host
  trickling candidate candidate:0 2 UDP 2128609534 10.8.115.201 65504 typ host
  trickling candidate candidate:0 1 UDP 2128609535 10.8.115.201 65505 typ host
  trickling candidate candidate:0 2 UDP 2128609534 10.8.115.201 65506 typ host
  trickling candidate candidate:0 1 UDP 2128609535 10.8.115.201 65507 typ host
  trickling candidate candidate:0 2 UDP 2128609534 10.8.115.201 65508 typ host

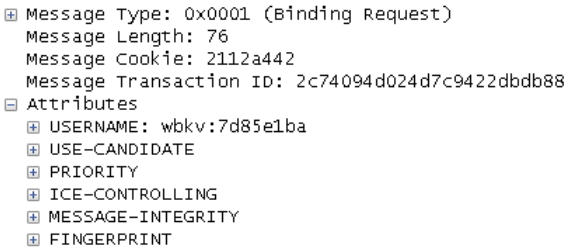

Figure 7.1: stun Request

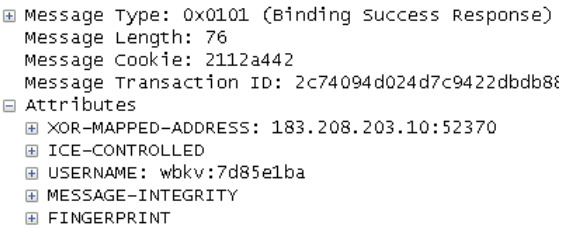

Figure 7.2: stun Response

- **STUN Signaling** As shown in figure above is stun signaling required to get Public Ip is shown.

  Stun Request is sent to "Stun Server "which will reply with XOR mapped address which is public IP in response.

- RTCP Packet Error **Couldn't write RTCP packet; SRTCP not set up**

  Unless RTCP packet as sender report it will not receive it will not to able to send SRTCP packet as Receiver report.

  able to debug that SRTCP packet it is not able to decrypt,so Protect key or algorithm is not set properly.

- RTP packet error **Error in unprotecting an SRTP packet of len 1086**

  To unprotect SRTP packet DTLS handshake will give key to decrypt it and protect algorithm is decided to apply on.

  If it is unable to Unprotect it mismatch in algorithm between Client and Gateway or Key is worong.

- RTP packet success **Successfully unprotected an SRTP packet of len 655**

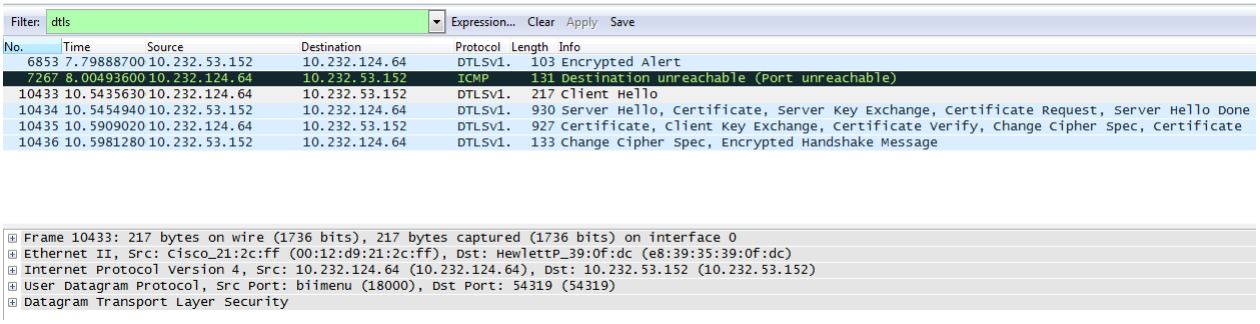

Figure 7.3: DTLS-Handshake

It has successfully unprotected SRTP packet and able to decode RTP packet,meaning DTLS key exchange is successful and algorithm used to protect and unprotect is correct.

- Protection Method **Setting SRTP cipher suite SRTP_ AES128_ CM_ HMAC_ SHA1_ 32** In client logs we can see what protection crypto policy is used for RTP when we set crypto policy.
- **codec info** VIDEO CODING: Codec: VP8, Payload type 100, Height 288, Width 352, Bitrate 100,Framerate 30

DTLS-Handshake As shown in fig 7.3 is wireshark result which come across network.

In DTLS handshke will happen between client-server. Gateway will act as a client and WebRTC client will act as a server.

gateway initiates process by calling client method and sends “Client Hello ”.

WebRTC Client will reply with “Server Hello”,and exchange of Certificate and Key exchange will happen with “Server Hello Done ”

RTP Packets As shown in figure 7.4 is wireshark result for RTP packets which flows from Media Server to Gateway.

It has many useful information which is useful for debugging.

Payload type will specify RTP format used for that Session.

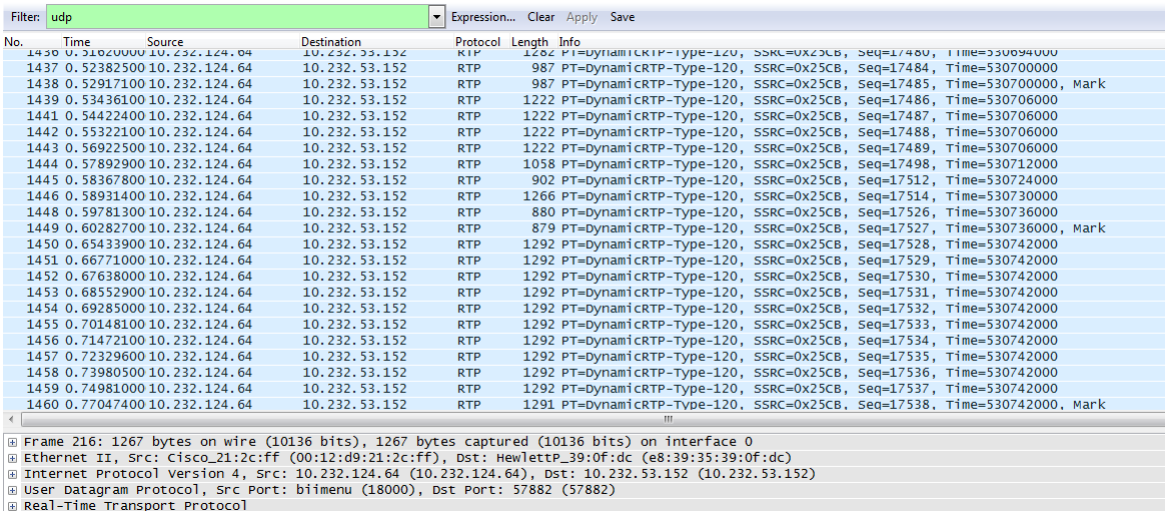

Figure 7.4: RTP packet

SSRC specifies from which source Packets are coming.

All packets coming from same source for a particular session must have same SSRC value.

Sequence will indicate sequence number,helpful for debugging packet loss and which packets are lost.

Mark indicates end of a frame.

Chapter 8

Conclusion and Future Scope

8.1 Conclusion and Future Scope

8.1.1 WebRTC

In WebRTC from client perspective

1) Camera List(channel list) will come which will have channel id and username and password to authenticate from data base,password is sha1 protected.

2) Multiple Client Support is added,single gateway will support multiple requests coming from different WebRTC Clients.

3) PTZ controll is added, existing pann/till/zoom support is given for each camera.

4) Multiple Streaming Support, in single client it can play multiple videos.

In Webrtc Gateway Ice support was added to support NAT(Network Address Translator) traversal with help of stun Server.

For Secured Communication and to protect replaying of RTP packets Secured RTP feature was added. To encrypt RTP and RTCP Packets,Private Key is required which was extracted with help of Datagramlayer security(DTLS) with concept of Master Key and Salt Key exported when DTLS handshake happens between client and Server.

In Media Server support for VP8 codec was added because webrtc client supports

vp8 codec only which is specifically for web related media streams.

8.1.2 Video Analytic

The current state of the art and direction of research in computer vision aimed at automating the analysis of images is presented. This includes low level identification of objects within the field of view of cameras, following those objects over time and between cameras, and the interpretation of those objects appearance and movements with respect to models of behaviour .

By Using Video Analytics Triggers there are good possible ways for Posting the data through XML files and HTTP posting to all the clients and the proposed system can record different types of triggers like Intrusion Detection,Baggage Detection and Crowd Control.

For Future Upgradation,the system can get the triggers like Vehicle Detection,People Count,Tracking of an individuals.The smart camera devices for applications like facial expression recognition must be embedded. However, more intelligent inference to identify people and objects and fully parse scenes is likely to require scalable data-centric systems that can be more efficiently scaled in a data center.

Functionalities can be extended in future by applying Face Recognition on those videos captured, and using Video Analytics the system can predict by analyzing the captured image this software will conclude whether the user is Authorized User or Intruder. Using Combination of Video Analytics and Face Recognition Project, some of the features can also be developed and deployed, and are listed as follows:

- In order to prevent the frauds of ATM, it is recommended to prepare the database of all ATM customers with the banks & deployment of high resolution camera and face recognition software at all ATMs. So, whenever user will enter in ATM his photograph will be taken to permit the access after it is being matched with stored photo from the database.

- Passport and visa verification can also be done using the proposed technology.

- Driving license verification can also be exercised using Video Analytics and face recognition technology.

Appendix A

Appendix A

This section describes the basic methods used for Face Recognition.

A.1 FaceRecognizer

class FaceRecognizer : public Algorithm

All face recognition models in OpenCV are derived from the abstract base class-FaceRecognizer, which provides a unified access to all face recongition algorithms in OpenCV. class FaceRecognizer : public Algorithm public:

- ! virtual destructor virtual FaceRecognizer()
- Trains a FaceRecognizer. virtual void train(InputArray src, InputArray labels) = 0;
- Updates a FaceRecognizer. virtual void update(InputArrayOfArrays src, InputArray labels);
- Gets a prediction from a FaceRecognizer. virtual int predict(InputArray src) const = 0;

Predicts the label and confidence for a given sample. virtual void predict(InputArray src, int & label, double & confidence) const = 0;

- Serializes this object to a given filename. virtual void save(const string & filename) const;

- Deserializes this object from a given filename. virtual void load(const string & filename);

- Serializes this object to a given cv::FileStorage. virtual void save(FileStorage & fs) const = 0;

  Deserializes this object from a given cv::FileStorage. virtual void load(const FileStorage & fs) = 0;

A.2 Setting the Thresholds

You can apply a threshold on the prediction, to tell, whether a face belongs to the training dataset or if it is unknown. Setting a threshold for the Eigen faces method, when creating the model

Let's say we want to keep 10 Eigenfaces and have a threshold value of 10.0 int numcomponents = 10; double threshold = 10.0; Then if you want to have a cv::FaceRecognizer with a confidence threshold, create the concrete implementation with the appropiate parameters:

$Ptr < FaceRecognizer > model == createEigenFaceRecognizer(num\_components, threshold)$

The following line reads the threshold from the Eigenfaces model:

double current_threshold = $model- > getDouble("threshold")$;

And this line sets the threshold to 0.0:

$model- > set("threshold", 0.0)$;

A.3 Getting the name of a FaceRecognizer

Create a FaceRecognizer:
$Ptr < FaceRecognizer > model = createEigenFaceRecognizer()$; And here's how to get its name: $std :: stringname = model- > name()$;

A.4 FaceRecognizer::train

Trains a FaceRecognizer with given data and associated labels. void FaceRecognizer::train(InputArrayOfArrays src, InputArray labels) = 0

Parameters:

src The training images, that means the faces you want to learn. The data has to be given as a $vector < Mat >$.

labels The labels corresponding to the images have to be given either as a $vector < int >$ The images are read with imread() and pushed into a $std :: vector < Mat >$ The labels of each image are stored within a $std :: vector < int >$ the label as the subject (the person) this image belongs to, so same subjects (persons) should have the same label. Create a new Fisherfaces model and retain all available Fisherfaces, this is the most common usage of this specific FaceRecognizer:

$Ptr < FaceRecognizer > model = createFisherFaceRecognizer()$; This is the common interface to train all of the available cv::FaceRecognizer implementations: $model- > train(images, labels)$;

A.5 FaceRecognizer::update

Updates a FaceRecognizer with given data and associated labels

- **src** The training images, that means the faces you want to learn. The data has to be given as a $vector < Mat >$.

- **labels** The labels corresponding to the images have to be given either as a $vector < int >$

Create a new LBPH model (it can be updated) and use the default parameters, this is the most common usage of this specific FaceRecognizer:

$Ptr < FaceRecognizer > model = createLBPHFaceRecognizer()$; This is the common interface to train all of the available cv::FaceRecognizer implementations:

$model- > train(images, labels)$; Some containers to hold new image: $vector < Mat >$ newImages; $vector < int >$ newLabels; You should add some images to the containers:

Now updating the model is as easy as calling: $model- > update(newImages, newLabels)$; This will preserve the old model data and extend the existing model with the new features extracted from newImages!

A.6 FaceRecognizer::predict

C++: int FaceRecognizer::predict(InputArray src) const = 0 C++: void FaceRecognizer::predict(InputArray src, int& label, double& confidence)const = 0 Predicts a label and associated confidence (e.g. distance) for a given input image.

Parameters:

- **src** Sample image to get a prediction from.
- **label** The predicted label for the given image.
- **confidence** Associated confidence (e.g. distance) for the predicted label.

The suffix const means that prediction does not affect the internal model state, so the method can be safely called from within different threads. The following example shows how to get a prediction from a trained model: using namespace cv; Do your initialization here (create the cv::FaceRecognizer model) ... ... Read in a sample image: Mat img = imread("person1.jpg", CV_ LOAD_ IMAGE_ GRAYSCALE); And get a prediction from the cv::FaceRecognizer: int predicted = model-¿predict(img);

A.7 FaceRecognizer::save

C++: void FaceRecognizer::save(const string & filename) const Saves this model to a given filename, either as XML or YAML. Parameters: filename The filename to store this FaceRecognizer to (either XML/YAML). C++: void FaceRecognizer::save(FileStorage & fs) const Saves this model to a given FileStorage.

A.8 FaceRecognizer::load

Loads a FaceRecognizer and its model state. C++: void FaceRecognizer::load(const string & filename) C++: void FaceRecognizer::load(const FileStorage & fs) = 0 Loads a persisted model and state from a given XML or YAML file . Every FaceRecognizer has to overwrite FaceRecognizer::load(FileStorage & fs) to enable loading the model state.FaceRecognizer::load(FileStorage & fs) in turn gets called byFaceRecognizer::load(const string & filename), to ease saving a model.

Appendix B

Appendix B

In this section Standard APIs used for project is explained.

B.1 NICE VISION API

B.1.1 VID_CONNECT

The **VID_ CONNECT** API function to receive alarms from Recorder.

VID_ Connect (& Handle, 10, //seconds IpAddress, & RetAddress);

Input Parameters

Table I: Input Parameter

Type	Name	Description
LONGINT	TimeoutInSeconds	The timeout, in seconds, used to send the commands to the server.
CHAR*	Address	The TCP/IP address of the server

Output Parameters

Table II: Output Parameter

Type	Name	Description
LONGINT	RetStatus	A value of Vid_ Ok is returned on success.
CHAR*	Handle	A communication handle, created and closed by the client.

B.1.2 SYS_AlarmConnect

The SYS_AlarmConnect API function is used to register the application and VID_StreamFetch to receive alarm message.

SYS_AlarmConnect (Handle, MessageId, NULL, NULL, & RetStatus, & MaxDataSize, NULL, NULL);

Table III: Input Parameter

Type	Name	Description
COMM_HANDLE	Handle	The communication handle.
DWORD	MessageId	The message ID from which we want to start receiving messages. 0: Start receiving new messages (usually sent on first connect) n: Start receiving from message ID = n. When reconnecting, use the last received message ID
void	*Reserved2...1	Reserved for future use.

Table IV: Output Parameter

Type	Name	Description
SYS_ STATUS*	*ret_ stat	Pointer indicating the status of the command. A value of Sys_ Ok is returned on success..
LONGINT	MaxDataSize	Maximum data size needed for one message body.
void	*Out_ Reserved2...1	Reserved for future use.

B.1.3 VID_StreamFetch

API is used to receive the alarm messages.

void VIDAPI VID_StreamFetch (COMM_HANDLE Handle,

void ResBuf,

LONGINT ResultLen,

LONGINT LongTimeOut,

VID_STATUS RetStatus,

LONGINT ReceivedSize);

Table V: Input Parameter

Type	Name	Description
COMM_HANDLE*	Handle	A communication handle, created and closed by the client.
LONGINT	ResultLen	The length of the result buffer containing the requested information.
LONGINT	LongTimeOut	A Boolean value. If **True**, timeout is set to **NC_LONG_TIMEOUT**.

B.1.4 VID_DisConnect

Unallocates a handle for a specific server address. void **VIDAPI VID_DisConnect**(/* Inputs: */ COMM_HANDLE Handle, /* Outputs: */ LONGINT RetStatus);

Table VI: Output Parameter

Type	Name	Description
void	ResBuf	The buffer of the result.
VID_STATUS	RetStatus	A value of **Vid_Ok** is returned on each retrieved. When all record has been read a value of **NC_CONNECTION_CLOSE** is returned. To stop fetching alarms before reading all the records or in case of failure, execute **VID_StreamClose**.
LONGINT	ReceivedSize	The size of the received buffer. **NC_LONG_TIMEOUT**.

Table VII: Input Parameter

Type	Name	Description
COMM_HANDLE*	Handle	A communication handle, created and closed by the client. Closes the handle obtained by **VID_Connect**.
LONGINT	ResultLen	The length of the result buffer containing the requested information.

Table VIII: Output Parameter

Type	Name	Description
LONGINT	RetStatus	A value of **Vid_Ok** is returned on success. **VID_Connect**.

References

[1] DTLS extension To Establish Keys for SRTP http://tools.ietf.org/html/rfc5764

[2] Session Description Protocol(SDP) http://tools.ietf.org/html/rfc4566.html

[3] Interactive Connectivity Establishment https://tools.ietf.org/html/rfc5245

[4] WebRTC book http://chimera.labs.oreilly.com/books/1230000000545/ch18.html

[5] STUN for NAT http://tools.ietf.org/html/rfc5389

[6] WebRTC http://www.webrtc.org/

[7] Paul Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE CVPR, 2001.

[8] Rainer Lienhart and Jochen Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.

[9] Online source URL: http://msdn.microsoft.com/en-us/library/debx8sh9(v=vs.110).aspx

[10] Real Time Intelligence Console(RIC) URL: http://www.motorolasolutions.com/US-EN/Business+Product+and+Services/Software+and+Applications/Public+Sector+Applications/Public+Safety+Applications/Real-time_Intelligence_Console

[11] R. T. Collins, R. Gross, and J. Shi. Silhouette-based human identification from body shape and gait. In Proc. of Fifth IEEE Conf. on Automatic Face and Gesture Recognition, pages 366371, 2002.

[12] Real Time Video Intelligence(RTVI) URL: http://www.motorolasolutions.com/US-EN/Business+Product+and+Services/Software+and+Applications/Public+Sector+Applications/Government+Video+Applications/Real-Time+Video+Intelligence

[13] Online source :Fiddler Tool URL: http://msdn.microsoft.com/enus/library/windows/desktop/ff966510(v=vs.85).aspx

[14] Online source : NICE NVR applications URL: http://www.nice.com/video/analytics

[15] Online source : Trigger Architecture URL:http://www.agentvi.com/20-technology-58-Video_ Analytics_ Architectures

[16] Online source URL:http://docs.opencv.org/modules/contrib /doc/facerec/facerec_tutorial.html/introduction

[17] Book: Adaptive image contrast enhancement using generalizations of histogram equalization

[18] C. Papageorgiou, T. Evgeniou, and T. Poggio. A trainable pedestrian detection system. In Proc. of IEEE Int. Conf. on Intelligent Vehicles, pages 241246, Germany, October 1998.

[19] Online source URL: http://docs.opencv.org/doc/tutorials/objdetect /cascade_classifier/cascade_classifier.html

[20] Online source URL: http://docs.opencv.org/modules/objdetect/doc /cascade_classification.html

[21] Online source URL:http://www.shervinemami.info/faceRecognition.html