RAMAIAH INSTITUTE OF TECHNOLOGY
(Autonomous Institute, Affiliated to VTU)

Project Report entitled

SMART PARKING AND SURVEILLANCE

Submitted in partial fulfillment for the award of degree
of
BACHELOR OF ENGINEERING
in
Electronics and Communication Engineering
By

Adhishree Jaiprakash	1MS13EC004
Karan R. Motwani	1MS13EC047
Nikhil Chhabria	1MS13EC069
Siddarth Asokan	1MS13EC107

Under the guidance of

Dr. K Indira	Dr. Bharadwaj Amrutur
Professor,	Chairman,
Department of E&C, RIT	RBCCPS, IISc

May 2017

RAMAIAH
Institute of Technology

Department of Electronics and Communication Engineering

CERTIFICATE

Certified that the project titled "SMART PARKING AND SURVEILLANCE" is a bonafide work carried out by ADHISHREE JAIPRAKASH (1MS13EC004), KARAN R. MOTWANI (1MS13EC047), NIKHIL CHHABRIA (1MS13EC069) and SIDDARTH ASOKAN (1MS13EC107) in partial fulfillment for the award of degree of Bachelor of Engineering in Electronics and Communication of Visvesvaraya Technological University, Belgaum, during the year 2016-17.

It is certified that all the corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the department library. The project has been approved as it satisfies the academic requirements in respect of project work prescribed for Bachelor of Engineering degree.

.....	.....	.....
Signature of Guide Dr. K Indira, Professor, Dept. of E & C, RIT	Signature of Guide Dr. Bharadwaj Amrutur, Chairman, RBCCPS, IISc	Signature of HOD Dr. S. Sethu Selvi, HOD, Dept. of E & C, RIT

External Viva:

Name of Examiners	Signature with date
1.	
2.	

Department of Electronics and Communication Engineering

DECLARATION

We hereby declare that the entire work embodied in this report has been carried out by us at Ramaiah Institute of Technology under the supervision of Dr. K Indira, Professor, Department of E & C, RIT and Dr. Bharadwaj Amrutur, Chairman, RBCCPS, IISc. This report has not been submitted in part or full for the award of any diploma or degree of this or any other university.

.....	.....
ADHISHREE JAIPRAKASH	KARAN R. MOTWANI
(1MS13EC004)	(1MS13EC047)
.....	.....
NIKHIL CHHABRIA	SIDDARTH ASOKAN
(1MS13EC069)	(1MS13EC107)

ACKNOWLEDGMENT

We would like to express our gratitude towards all the people who have contributed their precious time and effort to help us. Without them it would not have been possible for us to understand and complete the project.

We would like to thank **Professor Dr. K Indira, Department of Electronics and Communication Engineering, RIT**, our Project Supervisor for her guidance, support, motivation and encouragement through out the period this work was carried out. Her readiness for consultation at all times, her educative comments, her concern and assistance even with practical things have been invaluable.

We would also like to thank **Dr. Bharadwaj Amrutur, Chairman, RBCCPS, IISc**, **Dr. Abhay Sharma, Member of Technical Staff, RBCCPS, IISc** and **Srivatsa Bhargava Jagarlapudi, PhD Student, RBCCPS, IISc** for their invaluable guidance and support.

We would like to express our heart-felt gratitude to the faculty of Electronics and Communication Engineering Department, RIT and the faculty of RBCCPS, IISc for providing a challenging and enlightening experience.

A project of this nature could never have been attempted without our reference to and inspiration from the works of others whose details are mentioned in the References section. We acknowledge our indebtedness to all of them. Last, but not the least, our sincere thanks to all of our friends who have patiently extended all sorts of help for accomplishing this undertaking.

List of Figures

List of Acronyms

PoE	Power over Ethernet
SSH	Secure Shell
Pi	Raspberry Pi node
GST	GStreamer
V4L2	Video For Linux 2
GDP	GST Datagram Protocol
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
GMM	Gaussian Mixture Model
B.A.T.M.A.N	Better Approach To Mobile Adhoc Networking
RCNN	Region-based Convolutional Neural Network
SPP	Spatial Pyramid Pooling
ROI	Region Of Interest
FPS	Frames Per Second
IoT	Internet of Things
DHCP	Dynamic Host Configuration Protocol
NRE cost	Non-Recurring Engineering cost
ITS	Intelligent Transport Society
GPU	Graphics Processing Unit
RPS	Remote Processing Server
LAN	Local Area Network
SOC	System On a Chip

ABSTRACT

As the Internet of Things (IoT) ecosystem progresses to become all-pervasive, significant data, with the potential of solving several real world problems, has now become accessible through embedded technology. Given the increasing number of vehicles and rate of crime in major cities, surveillance data and smarter parking solutions have become a necessity. Thus, we cater to this problem statement by implementing a stable embedded wireless mesh network capable of low-cost surveillance and a video/image processing algorithm which provides parking availability detection from a remote server. Our solution has been tuned to specifically accommodate the less-than-ideal condition of Indian roads and the addition of future smart city applications.

Contents

Chapter 1

Introduction

Cities have a major impact on the economic and social development of nations. They are platforms for lifestyle, business and provide numerous services. Given a scenario that shows an urban environment with a growing demand for efficiency and resources, public administrations have to consider an evolution in the management models of cities. To do this, the use of Information and Communication Technologies (ICT) is essential. This has been translated as the Smart City concept which, with its services, is moving forward to what has now become known as the Internet of things and itself the Internet of the future.

1.1 Smart Cities : Definition and Features

1.1.1 Definition

We define a Smart City as a city which uses information and communication technologies so that its critical infrastructure as well as its components and public services provided are more interactive, efficient and so that citizens can be made more aware of them. In a broader definition, a city can be considered as smart, whenever investments in human and social capital, and in communication infrastructure, specifically encourage sustainable economic development and a higher quality of life, with judicious management of natural resources through participatory government.

1.1.2 Features

- ***Increases efficiency and quality of services:*** It makes it possible to manage resources in an optimized manner and improve the quality of the services provided.

- ***Provides information in real time:*** Enhances the awareness of citizens about the environment in which they live by providing information that flows in real time and also improves the transparency of the administration.
- ***Provides support in decision-making:*** Facilitates the identification of the needs of the city and the approach for new services to provide them with support.
- ***Reduces public spending in the long-run:*** Public spending on the provision and management of public services is reduced after the initial NRE Cost.
- ***Promotes innovation:*** Provides an ideal platform for innovating, incubating new businesses and, in general, promoting social development.

1.2 Need for Smart Parking

ITS America reports that 30 percent of all traffic congestion in urban areas is caused by drivers looking for a parking space, this wastes time, is inconvenient, and increases carbon dioxide emissions. In an attempt to alleviate traffic congestion, smart parking solutions aim to provide each driver with the information of parking availability on a specific road before he/she traverses it. Therefore, a user can access this information using a mobile or web application and then choose to enter/avoid a road based on the presence of an empty parking space. This creates a smarter environment where the percentage of congestion caused by scouring for a parking spot is reduced.

1.3 Need for Surveillance

As per National Crime Records Bureau (NCRB) and Ministry of Home Affairs, during 2010, a total of 6.7 million cognizable crimes comprising 2.2 million Indian Penal Code (IPC) crimes were reported. This number has been on a constant rise in the last 25 years. With the improving lifestyles of citizens, criminal intent is on the rise and there comes a need for increasing the level of sophistication in protecting the population. Increasing the area of a city under active surveillance behaves as an active deterrent for crimes as video surveillance provides clear evidence that can accelerate court proceedings and convictions.

Furthermore, an administrator observing the surveillance from traffic bottlenecks can dynamically control traffic signals to re-route traffic and alleviate congestion.

1.4 Existing Implementations

1.4.1 Fastprk, WorldSensing

Fastprk is a Smart City system that helps drivers find parking spaces more quickly and empowers cities to manage their parking resources more effectively. Most successfully deployed in Moscow, Fastprk also facilitates 24/7 monitoring and management of parking bays that correlates real-time occupancy and payment information. This technology translates to significant extra revenues for the city through the increase in occupancy and reduction in non-payment.
However, Fastprk uses magnetic sensors to detect the presence of vehicles. Although accurate, it requires the pre-emptive definition of distinct parking spots in all areas of application and consequent installation of a sensor at each defined bay. A central node then relays such information to the server for storage and manipulation of data. Such ideal conditions aren't available in India.

1.4.2 SFPark, San Francisco

SFPark is San Francisco's system for managing the availability of on-street parking. Taking effect in April, 2011, the program utilizes smart parking meters that change their prices according to location, time of day, and day of the week, with the goal of keeping about 15% of spaces vacant on any given block. The San Francisco Municipal Transportation Agency launched the system with congestion mitigation funding from the Federal Highway Administration in July 2010 as a fallback from a downtown cordon. It is one of several such systems in the world. The City of Calgary, Canada and the Calgary Parking Authority with their ParkPlus system have been using a similar demand based pricing model since 2008.
SFPark, similar to Fastprk, uses magnetic sensors to detect the presence of vehicles.

1.4.3 Siemens Smart City Vision

The Siemens Integrated Smart Parking Solution is a modular, infrastructure-based sensor system that goes beyond the possibilities of ground sensors. It provides information about the number, structure and legality of all cars in a parking lot.
This Smart City concept uses Radar sensors and is thus incapable of any other form of surveillance. Furthermore, any object covering the area of the car can be misinterpreted as an automobile.

1.4.4 Cisco Smart+Connected Parking

The Cisco Smart+Connected Parking solution provides citizens with real-time information about available parking. The solution gathers and delivers the data by combining the foundational Smart+Connected Wi-Fi infrastructure with IP cameras, sensors, and smartphone apps. It integrates with enforcement applications, pushing expiration notices to traffic officers for ticketing. It also provides visibility into parking analytics, including usage and vacancy periods, which helps cities with long-term planning.
The individual nodes are far more expensive thus deterring large scale implementation. The project, as a whole, is still in its infancy/vision stage.

1.5 Project Objective

In this project, we have designed a dual-band embedded mesh network which captures a real-time video feed using a camera attached to a Raspberry Pi S.O.C (System on Chip) i.e. a node, positioned at regular intervals on a road. A GStreamer pipeline provides a low latency framework for routing the stream from a node to the nearest server node via a stable wireless link and finally to the Remote Processing Server (RPS) through an optical fibre. At the RPS end, we have designed a video/image processing algorithm to accurately detect the availability of parking space in the vicinity of each embedded node. This information can now be ported via a web or mobile application to a user/driver to alleviate traffic congestion. Hence, our solution successfully addresses the problem of providing low-cost surveillance data and information about parking spot availability on a specific road.

Chapter 2

Proposed System

2.1 Proposed Solution

2.1.1 Basic Description

The proposed solution of the smart parking system consists of three major streams; networking, video streaming and video/image processing. The solution can be described as follows:

- The system will consist of Raspberry Pi sensor nodes, capable of setting up an Ad-hoc mesh network using B.A.T.M.A.N Protocol and is implemented on both 2.4 and 5 GHz networks.
- Video streaming is done using the GStreamer pipeline framework, which allows for encoding the video and transmitting it over the network.
- The video/image processing is done in real-time on MATLAB and Python+OpenCV. For the parking spot detection, the algorithm will consist of multiple stages of a decision tree to conclude on parking.

2.1.2 Block Diagram and Explanation

The implementation involves realizing a wireless ad-hoc mesh network that will enable capture and transmission of video feed which is the input to the system. The input stream is fed through a GST pipeline framework that is also capable of audio support. An IP camera connected to a Pi node captures the real world scenarios that will be communicated to the processing server through the realized network. The Pi nodes are capable of small local processing, if required by a specific application, but are majorly used for set-up and handling of network and to provide support for video transmission.

A platform-independent non-mesh client can be added to the network by a mesh bridge through an Ethernet connection to the Pi node. A Gateway Server is the medium through which the router, acting as a DHCP server, leases out IP addresses to the devices currently present in and joining the network. The feed relayed to the server is processed upon by the GPU that will evaluate results at different stages of a decision tree that is included as a part of an adaptive learning algorithm to learn Indian roads. These results are combined together to conclude upon the required information as per the initial request to the server.

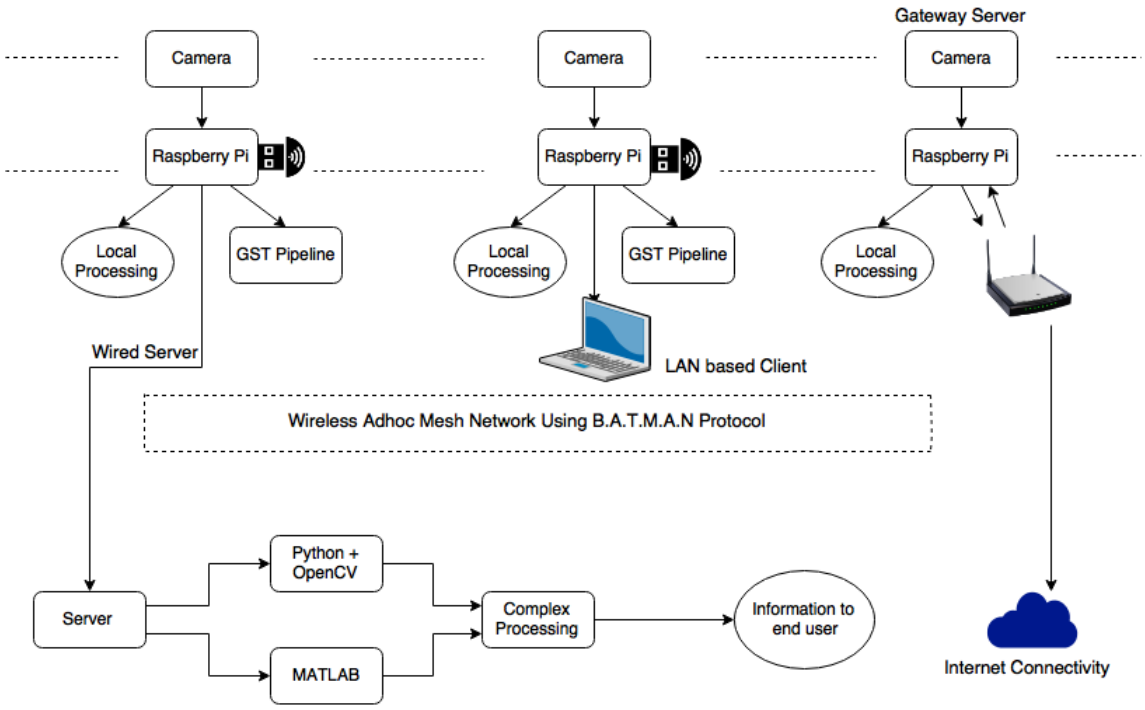


Figure 2.1: Block Diagram of the System

Advantages of Proposed Solution

- As this implementation is based on a wireless dual-band mesh network which allows for reliable video streaming and capable of surveillance applications in urban environments, it is cheaper than most existing wireless solutions such as Cisco Smart + Connected parking.

- Since the implementation is based on stable wireless mesh, it reduces the dependency on the much costlier fiber-optic connection at every node, which is hard to implement in smaller towns and cities without the base infrastructure. Simple 3G or 4G routers can be used at predetermined nodes to provide a stable internet connection to the mesh.
- The established network provides a foundation for new applications through the endless data available, such as detecting illegal parking, number plate detection, tracking objects or people over multiple camera feeds, crowd sensing and so on.

2.2 Hardware

2.2.1 Raspberry Pi 3 Model B

The Raspberry Pi is a small, single-board computer that was originally developed for computer science education and has since been popularized by digital hobbyists and makers of Internet of Things devices. Raspberry Pi, whose basic model costs $35 (approx. Rs. 2400) is about the size of a credit card, has a 64-bit quad-core ARMv8 processor and uses a Raspbian distribution of Linux for its default operating system. The Raspberry Pi computer is essentially a wireless Internet capable system-on-a-chip (SoC) with 1 GB RAM, connection ports, a Micro SD card slot, camera and display interfaces and an audio/video jack.

Advantages

- ***Affordable:*** Compared to other similar costing alternatives, the Pi (version B) offers the best specifications.
- ***Small Form Factor:*** At the size of a credit card, the Pi is very portable and consumes almost no space, regardless of application.
- ***Low Power Consumption:*** The Pi draws about five to seven watts of electricity at 5V, 1A.
- ***Built-in HDMI Capable Graphics:*** The display port on the Pi is HDMI and can handle resolutions up to 1920 by 1200.

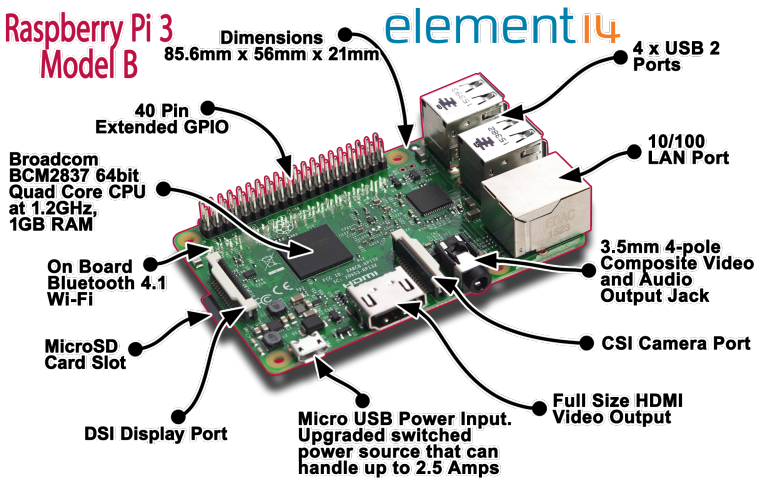


Figure 2.2: Raspberry Pi 3 Model B

There have been a number of tests that further elucidate the Raspberry Pi 3 model B's dominance over its competitors, a few have been described below:

- ***Blowfish Test***
  Blowfish is capable of strong encryption and can use key sizes up to 56 bytes (a 448 bit key). The key must be a multiple of 8 bytes (up to a maximum of 56). Thus measuring the processing capability of a device to decrypt a password in minimum time.

- ***FLAC Audio Encoding***
  FLAC stands for Free Lossless Audio Codec, an audio format similar to MP3, but lossless, meaning that audio is compressed in FLAC without any loss in quality.

- ***Performance to Cost, OpenSSL***
  Normalising the 4K performance of a device to its cost.

- ***C-Ray, Total Time***
  A simple raytracer designed to test the floating-point CPU performance. This test is multi-threaded (16 threads per core), will shoot 8 rays per pixel for anti-aliasing, and will generate a 1600 x 1200 image.

The results of these tests can be seen below and are in favour of using the Raspberry Pi 3 Model B as the nodal platform.

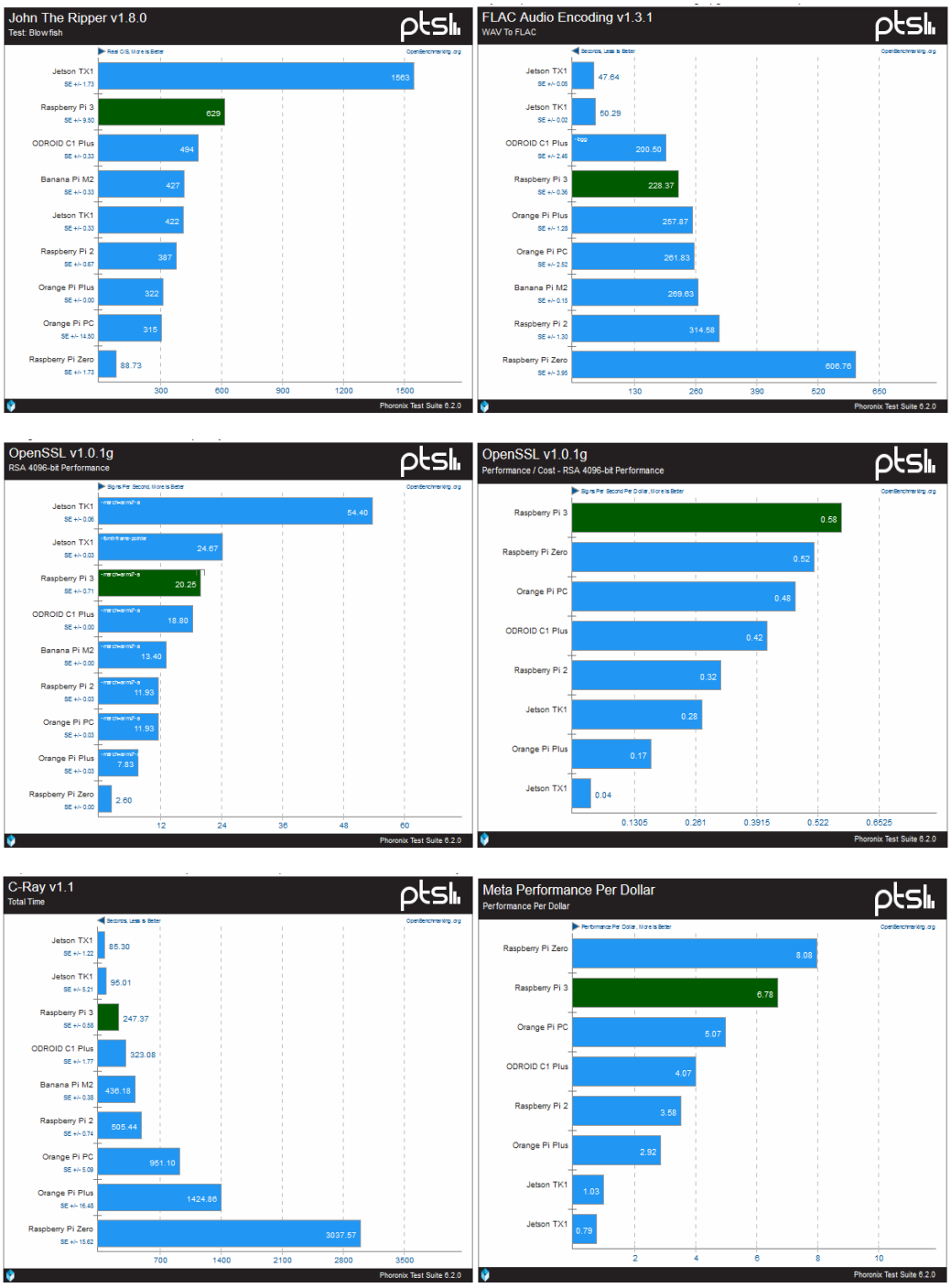


Figure 2.3: Results of different tests comparing Raspberry Pi 3 Model B to its competitors

2.2.2 IP Camera

An Internet protocol camera, or IP camera, is a type of digital video camera commonly employed for surveillance, and which, unlike analog closed circuit television (CCTV) cameras, can send and receive data via a computer network and the Internet.

Figure 2.4: IP Camera

2.2.3 USB Network adapters

For each Raspberry Pi node, we use two USB network adapters -

- TP Link TL-WN722N 2.4GHz adapter - This adapter operates on a 2.4GHz channel.
- Panda PAU09 Dual band adapter - This adapter operates in a 5GHz channel.

The on-board Wi-Fi adapter of a Pi node is incapable of providing adequate pole-to-pole range required for realization of the system. Thus, we use two external Wi-Fi adapters to provide the necessary range and data-rate.

(a) TP Link TL-WN722N

(b) Panda PAU09

Figure 2.5: USB Network Adapters

2.2.4 Router

The wireless ad-hoc mesh network consists of one or more routers, which apart from providing internet access, also act as DHCP servers i.e. on request, they provide IP addresses dynamically to mesh nodes and non-mesh clients.

Figure 2.6: Router

2.2.5 PoE Switches

A PoE switch is a network switch that has Power over Ethernet injection built-in where Power over Ethernet is a technology that lets network cables carry electrical power. PoE is ubiquitous on networked surveillance cameras, where it enables fast deployment and easy repositioning of IP cameras. Also, IP cameras are connected to PoE switches to provide required power up for the camera while also enabling data transmission.

Figure 2.7: 8-Port 10/100Mbps Desktop Switch with 4-Port PoE

2.2.6 Server (8GB+ RAM, NVIDIA GTX Graphics Card)

The server is that system that is capable of running heavy computations such as a machine learning algorithm.

2.3 Software

- Raspbian Jessie 4.4 - Open source operating system on each Raspberry pi 3 Model B node.
- Ubuntu 16.04.2 LTS - Open source operating system employed on server.
- B.A.T.M.A.N IV routing protocol using kernel modules batman-adv 2016.5 and batctl 2016.3, on each Raspberry Pi 3 Model B node
- Media Streaming using GStreamer Pipelining Framework
- Network testing tool - iperf3
- Programming tools - Shell, Python2.7 + OpenCV3.1, MATLAB R2016b

Chapter 3

Networking

3.1 Why Wireless Ad Hoc Mesh Networking?

A wireless ad hoc network (WANET) is a decentralized type of wireless network. The network is ad hoc because it does not rely on a pre-existing infrastructure, such as routers in wired networks or access points in managed (infrastructure) wireless networks. Instead, each node participates in routing by forwarding data for other nodes, so the determination of which nodes forward data is made dynamically on the basis of network connectivity. In addition to the classic routing, ad hoc networks can use flooding for forwarding data. Wireless mobile ad hoc networks are self-configuring, dynamic networks in which nodes are free to move. Wireless networks lack the complexities of infrastructure setup and administration, enabling devices to create and join networks "on the fly"; anywhere, anytime.

- Separation from central network administration.
- Self-configuring nodes are also routers.
- Mobility allows ad hoc networks created on the fly in any situation where there are multiple wireless devices.
- Scalability incorporates the addition of more nodes.
- Lower getting-started costs due to decentralized administration.
- The nodes in ad hoc network need not rely on any hardware and software. So, it can be connected and communicated quickly.

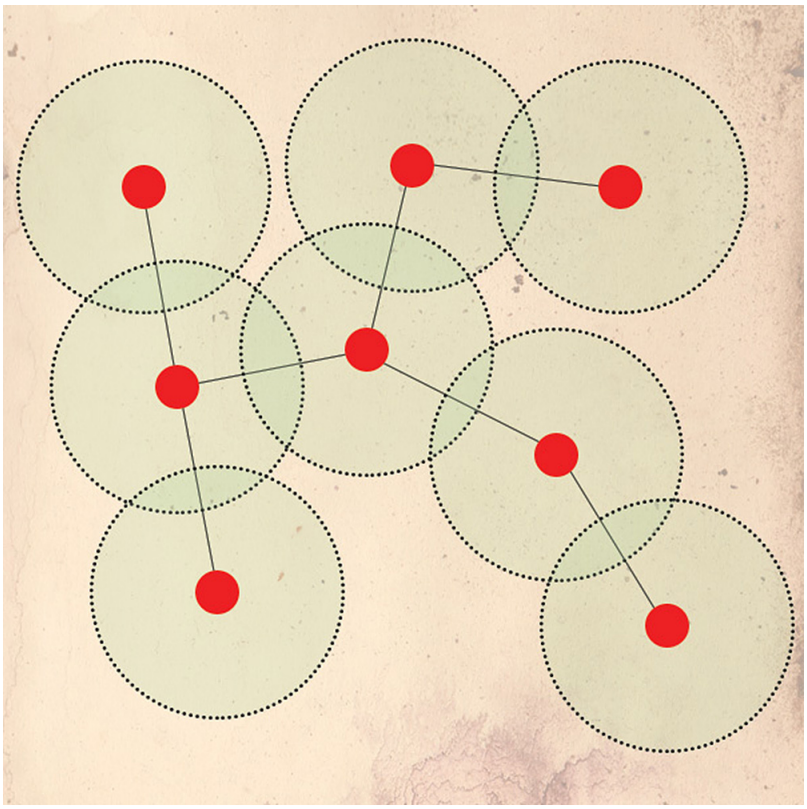
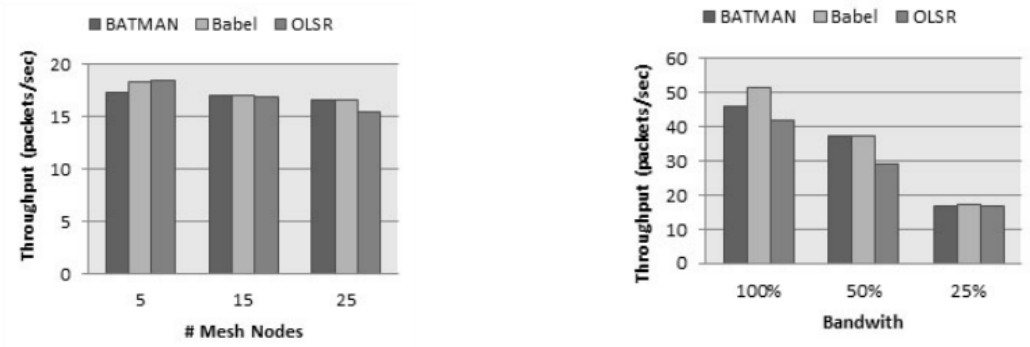


Figure 3.1: A Wireless Ad Hoc Mesh Network

3.2 B.A.T.M.A.N Ad Hoc Mesh Network

The Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.) is a routing protocol for multi-hop ad hoc networks that decentralizes the knowledge about the best route through the network so that no single node has to store all the data. The individual node is only concerned with the "direction" it received data from and sends its data accordingly. The B.A.T.M.A.N network differs from other ad-hoc networks as it performs its routing in the "Ethernet Layer" rather than in the "Network Layer"; Not only is the routing information transported using raw Ethernet frames, but also the data traffic is handled by the batman-adv. As a result, all nodes appear to be link local and are unaware of the network's topology. This also proves to improve the network bandwidth achieved by reducing overhead to only layer 2 processing. To facilitate establishing and controlling the B.A.T.M.A.N network, the "batctl" and "Alfred" tools are provided. Batctl is a configuration/debugging tool that can be used to add, delete, or configure nodes in the network and

can be used to run packet transfer and ping tests to verify connectivity. Alfred is a user space daemon used to broadcast data such as GPS locations or weather forecast.
It is found that for greater than 15 nodes, B.A.T.M.A.N protocol has the best performance in comparison to its competitors.



(a) Throughput vs. # Mesh Nodes

(b) Throughput vs. Bandwidth

Figure 3.2: Performance comparison of Batman-adv, Babel, and OLSR

3.3 Setting Up a B.A.T.M.A.N Node

Batman-adv kernel exists in Raspbian. It is loaded manually using Linux's "modprobe" instruction. If necessary, for Alfred, IPv6 can also be loaded. A Shell script is used to set up and execute the mesh network on system login. The script works as subsequently shown.

3.3.1 Code for Setting Up B.A.T.M.A.N Node

The on-board wireless adapter of the raspberry pi is defined as 'wlan-ob'. Any subsequent adapter connected to the Pi will be defined as wlan0, wlan1, and so on. The code first checks for the presence of a USB adapter and allots it a name as defined above. It is then defined as a 2.4GHz adapter if it is. Else, if it is a dual band adapter, it is defined as a 5GHz adapter.

```
sudo iw reg set IN
sudo modprobe batman-adv

name="wlan"
adap_24=""
adap_5=""
```

```
for i in 0 1 2 3 4 5 6 7 8 9
do
  break_var=$(ls /sys/class/net | grep $name$i)
  if [ -z $break_var ]; then
    break
  fi
  dualband_check_var=$(iwlist $name$i freq | grep "Channel 36")
  if [ -z "$dualband_check_var" ]; then
    if [ -z $adap_24 ]; then
      adap_24=$name$i
    fi
  else
    if [ -z $adap_5 ]; then
      adap_5=$name$i
    fi
  fi
done
```

If no USB adapter is connected or if one dual band USB adapter is connected, the default on-board adapter (2.4GHz) is used. If there are multiple (two or more) dual band USB adapters connected, one of them is used for 2.4GHz.

```
if [ -z $adap_24 ]; then
  if [ $i = 0 ] || [ $i = 1 ]; then
    adap_24="wlan-ob"
  else
    adap_24="wlan1"
  fi
fi
```

Next, the B.A.T.M.A.N network is set up by adding the appropriate wireless layers to the batctl interface. Other parameters such as the channel of operation, network security mode, MTU, etc. can be set.

```
sudo ip link set mtu 1532 dev $adap_24
sudo ifconfig $adap_24 down
sudo iwconfig $adap_24 mode ad-hoc essid my-mesh-network
sudo iwconfig $adap_24 ap 02:12:34:56:78:9A channel 6 key off
sudo ifconfig $adap_24 up
sudo batctl if add $adap_24
sudo ip link set up dev $adap_24
```

```
if[ ! -z $adap_5 ]; then
  sudo ip link set mtu 1532 dev $adap_5
  sudo ifconfig $adap_5 down
  sudo iwconfig $adap_5 mode ad-hoc essid my-mesh-network
  sudo iwconfig $adap_5 ap 02:12:34:56:78:9A channel 36 key off
  sudo ifconfig $adap_5 up
  sudo batctl if add $adap_5
  sudo ip link set up dev $adap_5
fi
```

Next, the network is bridged at the Ethernet layer with the bat0 layer. This allows for non-mesh clients to be connected via LAN to the Raspberry Pi nodes. These devices will also be added to the network and allotted IP addresses. These clients can be Linux, MacOS or Windows based devices.

```
sudo ip link add name mesh-bridge type bridge
sudo ip link set dev eth0 master mesh-bridge
sudo ip link set dev bat0 master mesh-bridge
sudo ip link set up dev eth0
sudo ip link set up dev bat0
sudo ip link set up dev mesh-bridge
```

Finally, the node is configured as either a gateway server or client based on a predetermined configuration. All gateway servers are connected to Routers acting as DHCP Servers and these Routers will be the sources of IP addresses, as well as the gateways to the internet.

```
# if node is gateway client
sudo batctl gw client
sudo ifconfig mesh-bridge down
sudo ifconfig mesh-bridge up

# if node is gateway server
sudo batctl gw server
sudo ifconfig eth0 down
sudo ifconfig eth0 up
```

3.4 Dynamic assignment of IPv4 Addresses

The batman-adv mesh network is set up so that IPv4 addresses are dynamically and automatically assigned to all mesh nodes and non-mesh clients (connected via Ethernet to any node in the network). There are a few mesh nodes set up as gateway servers, each of which would be connected via Ethernet to a router. The remaining mesh nodes are set up as gateway clients. Each router, apart from providing internet access, also acts as a DHCP server. The DHCP server on each router is configured to lease out IPv4 addresses in a certain range, to mesh nodes and non-mesh clients.
DHCP requests issued by mesh nodes (set up as gateway clients) and non-mesh clients will not be broadcasted by batman-adv but only sent to the chosen batman-adv gateway via unicast. The DHCP client will then select the gateway batman-adv prefers as only this gateway replied to the request.

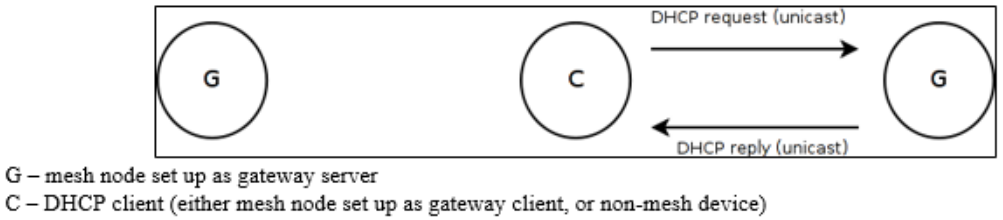


Figure 3.3: DHCP behavior with batman-adv gateway client mode enabled

3.5 Multilink Optimization

Mesh nodes with multiple physical links to other mesh nodes can use this capability for interface alternating. Interface Alternating basically tries to forward a packet on another interface than on which the frame was received. This helps to minimize multi-hop bandwidth degradation.

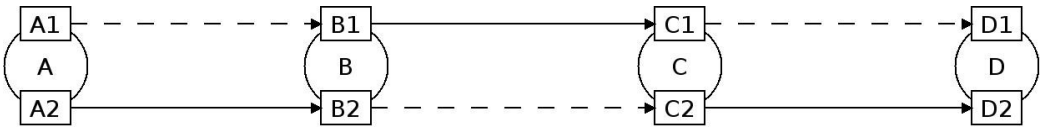


Figure 3.4: Mechanism of Interface Alternating

In the mesh network, each mesh node would be set up with two physical interfaces (each provided by a separate USB wifi adapter) on the 2.4 GHz band, and the 5 GHz band. Suppose a mesh node receives a packet on its

2.4 GHz interface, then through interface alternating, this packet would be forwarded on its 5 GHz interface and vice versa. This mechanism therefore avoids the classic 'store and forward' technique in the case of a single interface, and hence reduces bandwidth loss per hop.
Each mesh node would have three routing tables instead of just one:

- a routing table per incoming interface
- an additional 'default' routing table for packets generated locally (e.g. from the soft-interface)

Even when there is no hop, Multilink optimization considerably improves bandwidth through path diversity.

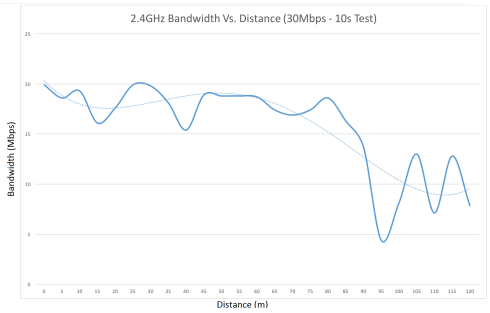
3.6 Networking Test Results

All the networking tests were carried out in the IISc campus in a practical environment that included real world obstacles such as trees and vehicles. Testing involves setting up one of the Raspberry Pi's as a test server on 'iperf3'. All other Raspberry Pi's are set up as test clients on 'iperf3'.

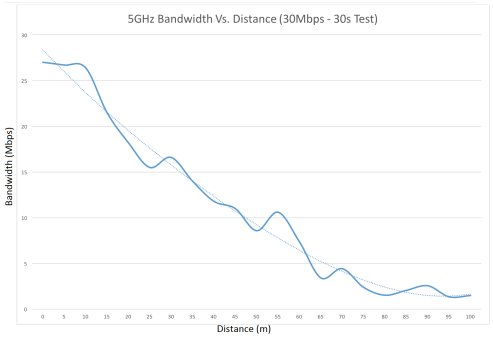
We conducted a bandwidth vs. distance test, seperately for 2.4GHz and 5GHz connections, to understand how bandwidth degrades with distance in each case. Fig 3.5(a) is an analysis of performance of an embedded node operating on 2.4GHz only. The test was conducted for 10s with a request for 30Mbps where the requested bandwidth is promised only if the set up can handle it. The best fit curve of the test results revealed that a channel with an average of 18Mbps bandwidth can be allotted to a single node up to 65m from the server and deteriorates further on. However, the channel sustains with 10Mbps even at 110m which is enough for a single stream. The 2.4GHz channel promises range but multiple streams at large distances from the server fail.

Fig 3.5(b) is a similar analysis of the 5GHz channel. The test was run for 30s with a single node requesting for 30Mbps. As expected, the 5GHz test performs poorly in terms of range compared to the above but is capable of handling higher data rates at closer distances to the server. The best fit curve indicates an exponential decrease in bandwidth with range and thus multiple streams on the 5GHz channel at large distances from the sever is impossible.

Fig 3.5(c) compares the performance on the 2.4GHz and 5GHz channels. The red lines indicate variation of bandwidth on 5GHz with distance and blue lines for the 2.4GHz channel. As seen in the figure, the 5GHz channel can maintain high bandwidth up to 25m from the server while 2.4GHz channel can maintain longer range. With interface alternating, a single node in motion away from the server would ideally start transmission on the 5GHz channel and would gradually shift to 2.4GHz as it crosses 30m. Although the node could manage a single stream at a distance greater than 110 meters from the server, this streaming would result in high data loss and latency. With multiple streams, transmission from nodes happens based on crowd on the interface and the distance it is at from the server.



(a) 2.4GHz Bandwidth vs. Distance



(b) 5GHz Bandwidth vs. Distance

(c) 2.4GHz and 5GHz Bandwidth vs. Distance Comparison

Figure 3.5: Bandwidth vs. Distance Tests

We conducted a bandwidth vs. time test with each Raspberry Pi having both 2.4GHz and 5GHz interfaces. This test was done using two different network topologies -

I.

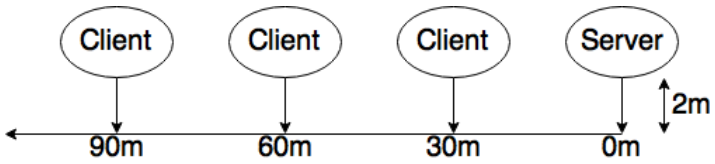
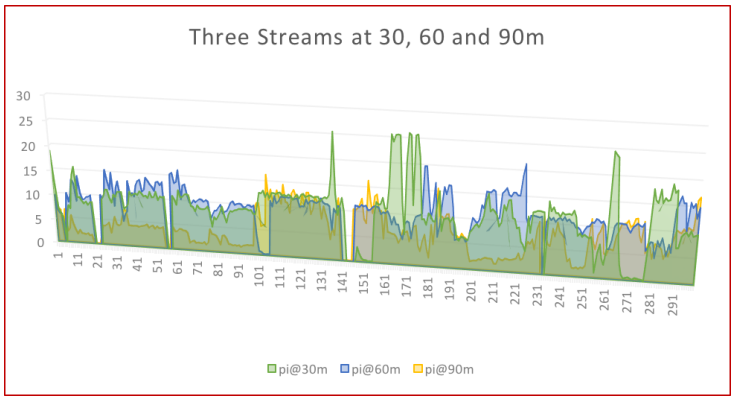


Figure 3.6: Network Topology 1

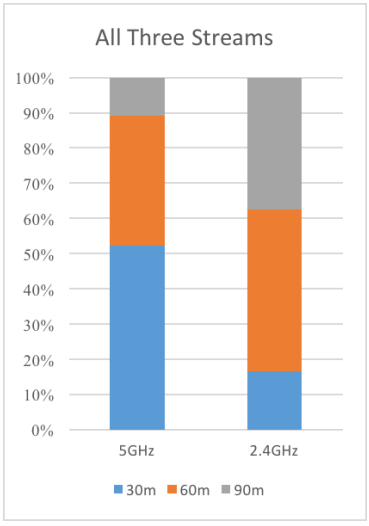
In this network topology, one mesh node is set up as test server and three other mesh nodes at 30m, 60m and 90m from the test server respectively are set up as test clients. All mesh nodes are at a height of 2m from the ground.

Fig 3.7(a) is a graph of bandwidth allotted to 3 different streams at any given instant over a 5-minute bracket, each placed at 30m, 60m and 90m from the server respectively. Multilink Optimization is employed, where any node can transmit on either 2.4GHz or 5GHz based on feasibility and availability. The graph reveals a balance in allotting bandwidth where sudden peaks of the node at 30m are accompanied by drops in bandwidth of the farther nodes and rise in bandwidth of farther nodes with drop in bandwidth of the closer nodes. The nulls in the figure are indicative of failure to transmit that instant which could be because of unstable environmental conditions to sustain wireless transmission such as vehicles, or interference from other existing wireless networks around the testing region. The average bandwidth available for the nodes were 10.7Mbps, 10.9Mbps, 6.51Mbps at for the Pi's at 30m, 60m and 90m from the server respectively.

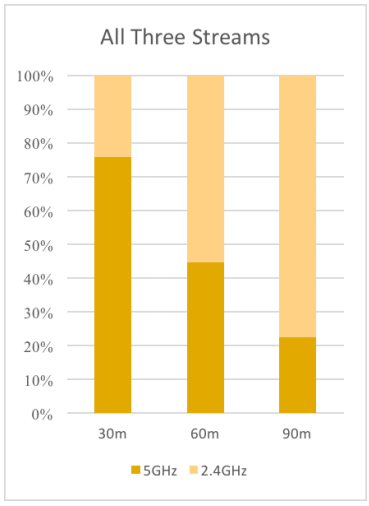
Fig 3.7(b) indicates percentage of traffic by each node on each channel. On the 5GHz channel, 52% of the traffic was by the node at 30m, 37% by the 60m node and only 11% by the node at 90m. Also, 17% of the traffic on the 2.4GHz channel was by the 30m node, 46% by the 60m node and 37% by the 90m node. Fig 3.7(c) depicts percentage data sent by each node on each channel. We see that the node at 30m sent 76% of its data on the 5GHz channel and rest on the 2.4GHz channel. Whereas, the 60m node sent 44% of its data and the 90m node sent only 22% of its data on the 5GHz channel. These numbers explain the higher bandwidth allotted to the node at 60m as it sends large amounts of data on both channels.



(a) Bandwidth vs. Time



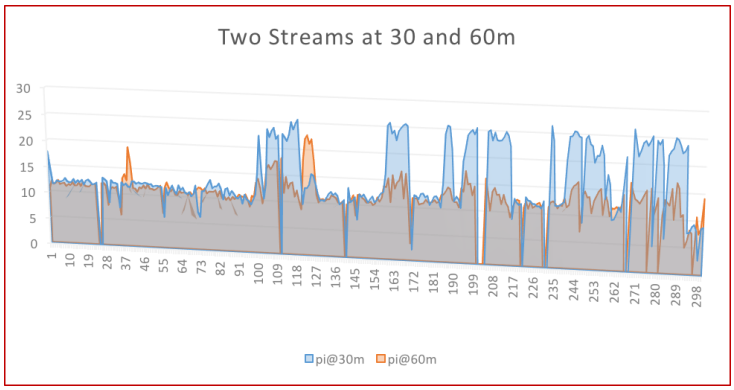
(b) Percentage traffic by each node on each channel



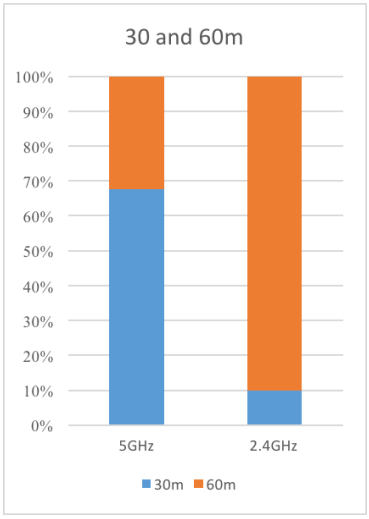
(c) Percentage data sent by each node on each channel

Figure 3.7: Tests for all 3 streams at 30m, 60m and 90m

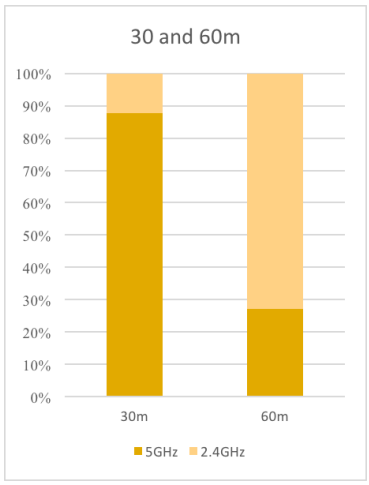
Fig 3.8(a) is bandwidth allotted to two nodes at 30m and 60m respectively over 5 minutes, with both nodes having access to 2.4GHz and 5GHz channels. The graph indicates that the node at 30m receives higher bandwidth than the other, as expected. The average bandwidth available for the 30m and 60m nodes were 15.9Mbps 12.3Mbps respectively.
Fig 3.8(b) indicates that 67% of traffic on the 5GHz channel and 9.43% of the 2.4GHz channel traffic was by the 30m node. Thus, the higher bandwidth for the 30m node was because of the time it spent on the 5GHz channel due to its proximity to the server. Fig 3.8(c) shows that of the data sent by the node at 30m, 88% was on the 5GHz channel and the rest 12% on the 2.4GHz whereas the 60m node sent only 27% of its data on the 5Ghz channel.



(a) Bandwidth vs. Time



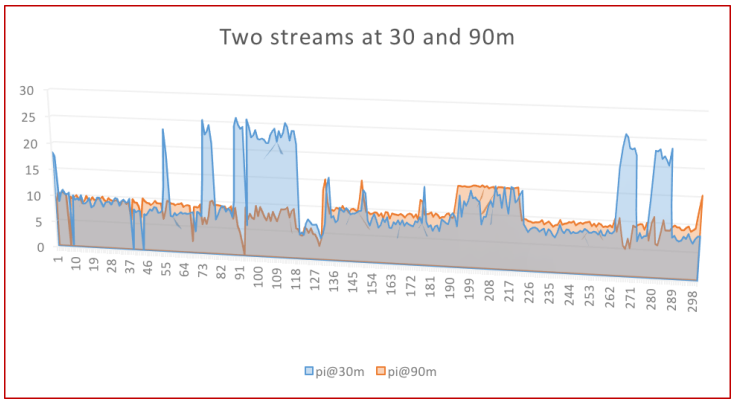
(b) Percentage traffic by each node on each channel



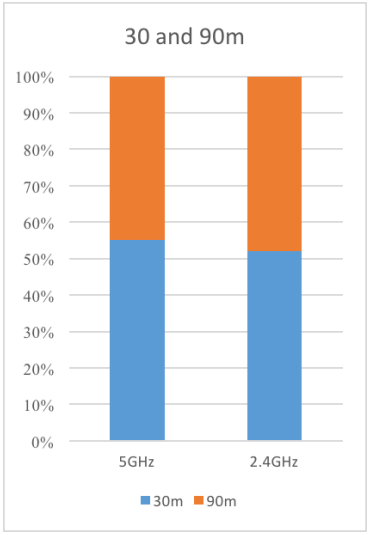
(c) Percentage data sent by each node on each channel

Figure 3.8: Tests for 2 streams at 30m and 60m

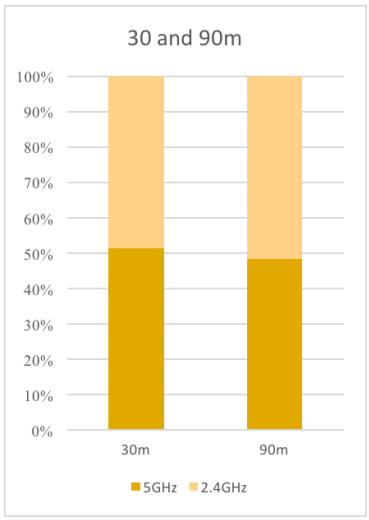
Fig 3.9(a) shows bandwidth vs. time test with two streams over 5 minutes, one at 30m from the server and the other at 90m. The 90m node maintains a stable low bandwidth whereas the 30m node experiences hikes in its rates. The average bandwidth allotted to the 30m node was 11.5Mbps which caused 55% traffic on the 5GHz channel and 52% traffic on the 2.4GHz channel as seen in Fig 3.9(b). The bandwidth allotted to the 90m node was 9.53Mbps which caused 44% of the traffic on 5GHz and 48% on the 2.4GHz channel. Some information exchange between the other nodes (gateway server node and the node at 60m) contribute to 1% traffic on the 5GHz channel. Fig 3.9(c) depicts that, of the data sent to the server by the node at 30m, 51% was on the 5GHz channel while the 90m node sent 48% of its data on it.



(a) Bandwidth vs. Time



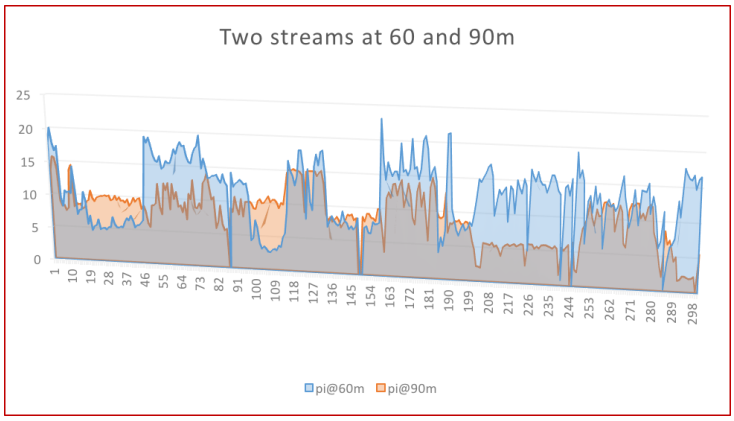
(b) Percentage traffic by each node on each channel



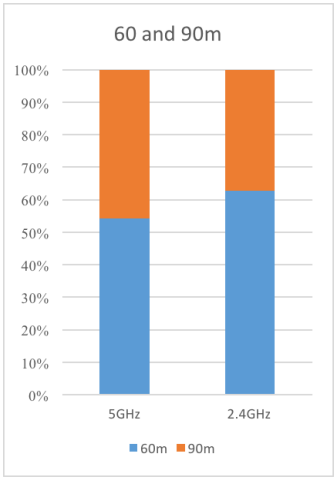
(c) Percentage data sent by each node on each channel

Figure 3.9: Tests for 2 streams at 30m and 90m

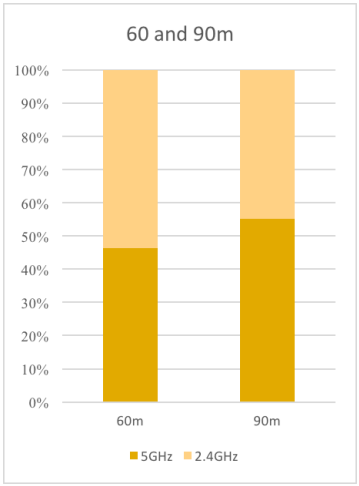
Fig 3.10(a) shows bandwidth variations over 5 minutes for nodes at 60m and 90m respectively. The 60m node experiences more peaks than the node at 90m. The average bandwidth allotted to the 60m node was 12.1Mbps and 9.07Mbps to the 90m node. Fig 3.10(b) shows that 54% of the traffic on the 5GHz and 63% on the 2.4GHz channels was by the 60m node and the 90m node contributed to 46% traffic on the 5GHz and 37% on the 2.4GHz. From Fig 3.10(c), we see that, of the data sent by the 60m node, 47% was on the 5GHz channel and rest on 2.4GHz while the 90m node sent 55% of its data on the 5GHz channel. This is because the 90m node was surrounded by other 2.4GHz networks while testing and thus chose the 5GHz channel.



(a) Bandwidth vs. Time



(b) Percentage traffic by each node on each channel



(c) Percentage data sent by each node on each channel

Figure 3.10: Tests for 2 streams at 60m and 90m

II.

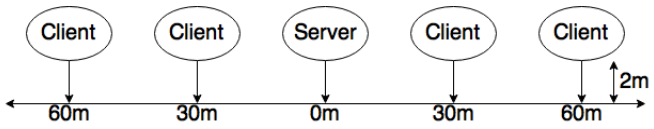
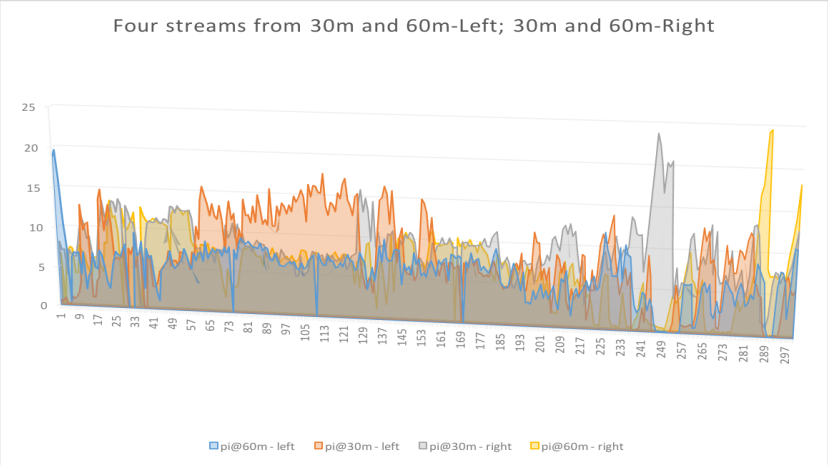


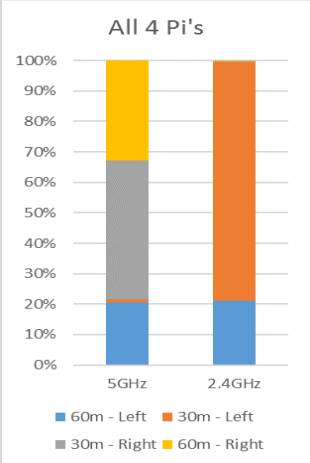
Figure 3.11: Network Topology 2

In this network topology, one mesh node is set up as test server and four other mesh nodes - two at 30m and 60m respectively from test server on each side, are set up as clients. All nodes are at a height of 2m from the ground.

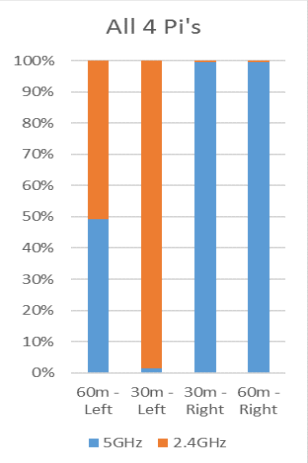
Fig 3.12(a) is bandwidth vs. time test over 5 minutes of four streams following topology in Fig 3.11. The average bandwidths starting from the leftmost client as in the diagram is 6.6Mbps, 9.04Mbps, 7.85Mbps and 6.09Mbps respectively where Pi at 60m (R) and Pi at 30m (R) sent mostly on the 5GHz channel and created 32.95 and 45.45 percent of the traffic on that channel.
From Fig 3.12(b) and 3.12(c) we conclude that the Pi at 30m (L) sent mostly on the 2.4GHz channel and created 1.07% of traffic on the 5GHz and 78% traffic on the 2.4GHz channel. The Pi at 60m (L) sent equally on both channels and contributed to 20% of traffic in each.



(a) Bandwidth vs. Time



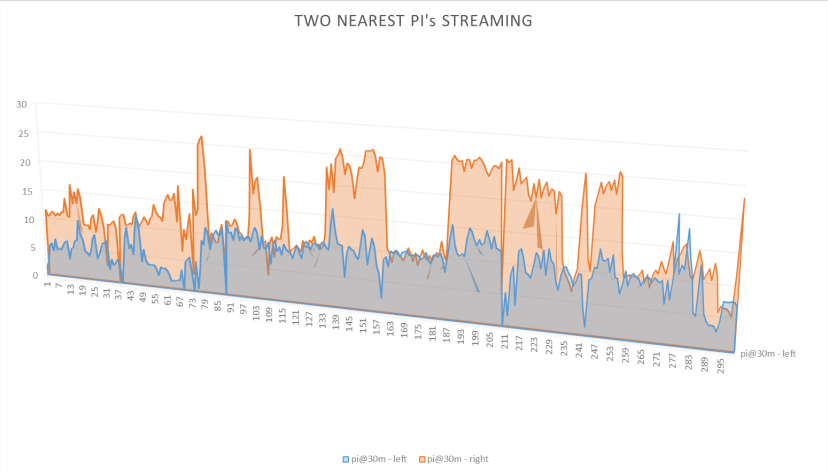
(b) Percentage traffic by each node on each channel



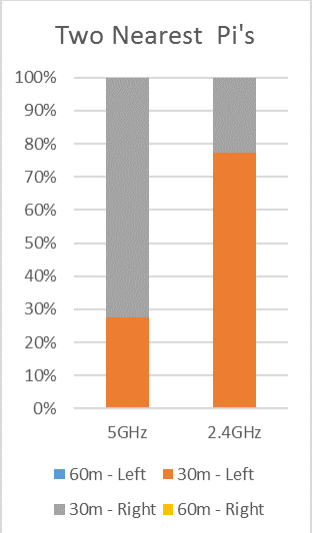
(c) Percentage data sent by each node on each channel

Figure 3.12: Tests for all 4 streams at 60m(L), 30m(L), 30m(R), 60m(R)

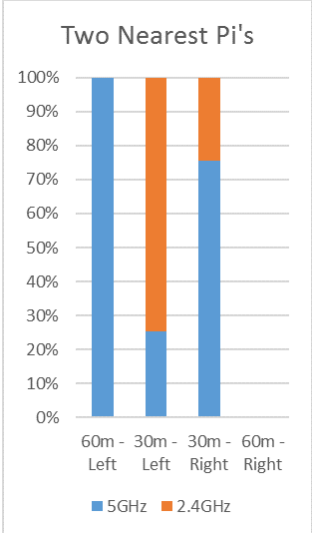
Fig 3.13(a) indicates bandwidths over 5 minutes time for two streams; by the 30m (L) and 30m (R) Pis. From 3.13(b) and 3.13(c) we know that Pi at 30m (R) sent around 75% of the data on the 5GHz channel that contributed as 73% of traffic on the channel and the Pi at 30m (L) sent around 75% of its data on the 2.4 GHz channel creating a 80% traffic on the 2.4GHz channel.



(a) Bandwidth vs. Time



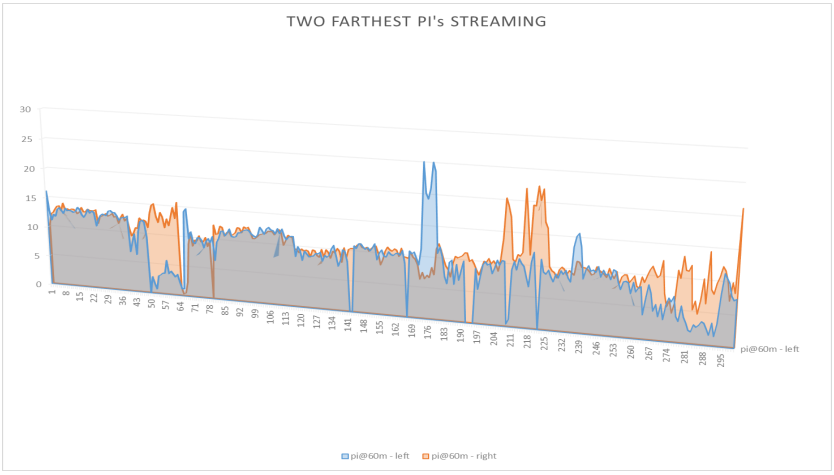
(b) Percentage traffic by each node on each channel



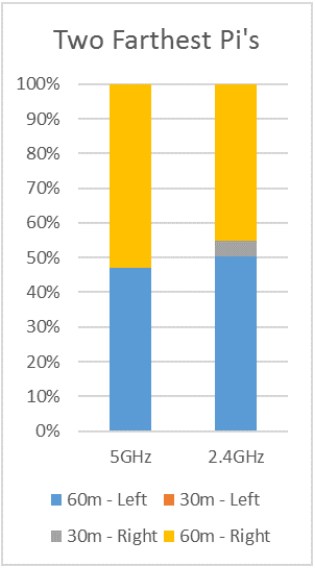
(c) Percentage data sent by each node on each channel

Figure 3.13: Tests for 2 streams at 30m(L) and 30m(R)

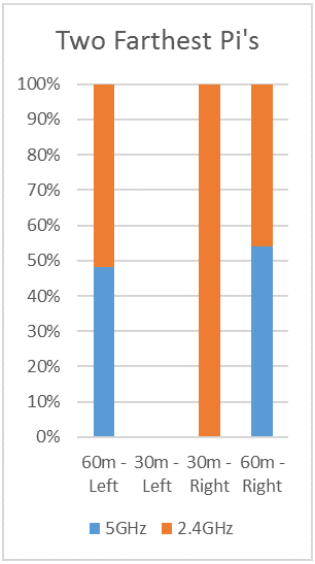
Fig 3.14(a) is representation of bandwidth variation over 5 minutes time of Pi at 60m (L) and Pi at 60m (R) actively streaming. Pi at 60m (L) created 47% traffic on the 5GHz channel by transmitting 48% of its data on that channel. It also contributed to 50.25% of the traffic on 2.4GHz channel by sending the rest 52% of its data on that channel. The Pi at 60m (R) created 53% of traffic on the 5GHz channel by sending around 52% of its data on that channel.



(a) Bandwidth vs. Time



(b) Percentage traffic by each node on each channel



(c) Percentage data sent by each node on each channel

Figure 3.14: Tests for 2 streams at 60m(L) and 60m(R)

Chapter 4

Video Streaming

Processing methods such as object detection need a high speed and low latency transfer of video data to a common server for detection, tracking and updating of parking information. The intended results would ideally demand around 25-30 frames per second.

4.1 Implementations

4.1.1 Motion

Motion is a service for video access through browser interface that follows TCP to server through wireless ports. It requires the source be specified: picam or video feed through port. Advantage of Motion lies in its simplicity and easy control of specifications through motion.conf. It uses ffmpeg encoding, mpeg4 and v4l2 pulgins. Latency in using Motion is caused by mpeg4 encoding, uploading frames onto the browser and the TCP connection, resulting in motion being discarded as a viable video streaming option.

4.1.2 Socket Programming

The Python Socket code uses a raw socket connection to access a TCP port and communicate through it. The sender script specifies the server IP address and the TCP port that it is uploading the data to, while the receiver script listens to connections on its server, and creates a thread to all devices requesting connection.

4.1.3 GStreamer

GStreamer is an open source Pipeline based multimedia framework that links a variety of media processing elements. GStreamer Pipelines Support both audio and video transmission, allowing audio based applications of the network in the future. It can be implemented on Raspbian Linux, Ubuntu and OSX. It provides encoder blocks for mpeg4, jpeg, h.264 and x.264 through TCP and UDP implementation.

SL. NO.	STREAMER	RESOLUTION	ENCODING	CONFIGURATION	BANDWIDTH USED/stream	LATENCY (seconds)	RECEIVER FRAMERATE
1	Motion	320x240 @ 30fps	mpeg4	No Hop	2Mbps	>1s	4-5fps
				One Hop	2Mbps	>2s	4-5fps
2	Socket Programming	640x480 @ 24fps	Jpeg frames	No Hop - 1 sender	6Mbps	<1s	24fps
				No Hop - 2+ senders	6Mbps each	<1s	24fps
				One Hop - 2+ senders	Sockets prevent multi-hop		
3	GStreamer TCP	640x480 @ 30fps	H.264 or Jpeg frames	No Hop - 1 sender	3Mbps	<0.5s	30fps
				No Hop - 2+ senders	3Mbps each	0.5-1s	30fps
				One Hop	3Mbps	<1s	25-30fps
4	GStreamer UDP	640x480 @ 30fps	H.264 or Jpeg frames	No Hop - 1 sender	2.5Mbps	<0.5s	30fps
				No Hop - 2+ senders	2.5Mbps each	<.7s	30fps
				One Hop	2.5Mbps	<1s	30fps

Figure 4.1: Comparison of Motion, Socket Programming, and GStreamer

As a result of the bandwidth restrictions and large latency on Motion, as well as multi hop deficiencies observed on Socket Programming, GStreamer was ascertained as a viable alternative. The low latency and approximately 5 Mbps bandwidth per stream make it a desirable alternative. Multiple configurations were tested such that a stable pipeline setup between adjacent nodes was reliable and produced a 25-30 fps video feed. The 5 Mbps bandwidth per stream accommodates a H.264 encoded 720p resolution video feed from the IP surveillance cameras being used.

4.2 What is GStreamer?

GStreamer is an open source Pipeline based multimedia framework that links a variety of media processing elements. Each element can read file from a source, process it, and output onto a sink.

4.2.1 Setting Up a GStreamer Pipeline

Consider a model GStreamer pipeline as shown. The first element of the pipeline acts as a source and reads data from a file or an input device. Each element's output acts as the input to the next. The filter's element can be used for processing audio or video, such as a multiplexer, encoder or decoder. Each GStreamer element has a number of properties that control the functionality of the element, called Capabilities (Caps). The Caps can be used to set various parameters of a filter, such as the framerate, resolution of the video, the transmission rate, or the host address and port of a TCP or UDP server.

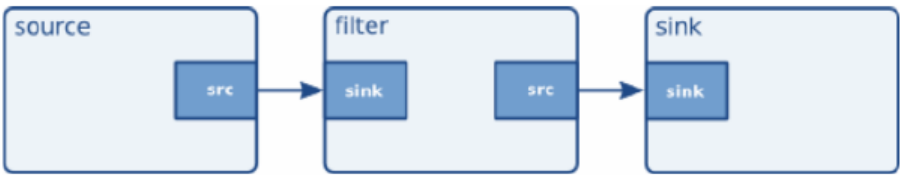


Figure 4.2: A Model GStreamer Pipeline

4.3 GStreamer Pipelines for UDP H.264 Streaming from Webcam

4.3.1 Sender Pipeline

The video source (a USB webcam) is read as raw frames using the V4L2 (Video4Linux2) drivers available natively on Linux. The framerate is set to 30 fps, while the resolution is set to the webcam's maximum of 640x480. This video feed is then sent to the OMX H.264 encoder, a video codec provided by the OpenMAX Library. The output is encapsulated using the Real-time Transport Protocol (RTP) H.264 Payload format as specified by RFC 6184. The output is sent to the GDP payloader which encapsulates the data in a GST Datagram Protocol (GDP) header. This allows for the sender code to

be executed without a listening receiver on the other end of the UDP link. The encoded data is finally sent out to the specified receiver IP address host through the specified UDP port.

```
gst-launch-1.0 -v v4l2src ! video/x-raw, framerate=30/1, width=640,
height=480 ! omxh264enc ! rtph264pay config-interval=1 pt=96 !
gdppay ! udpsink host=192.168.1.5 port=9001
```

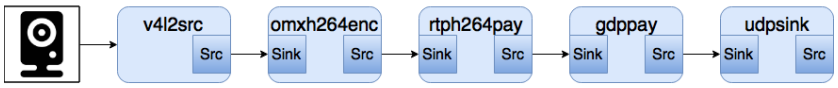


Figure 4.3: GStreamer Sender Code and Pipeline for Streaming H.264 Video from USB Webcam using UDP

4.3.2 Receiver Pipeline

The video feed of the sender is accessed by the UDP receiver host by specifying the particular UDP port. The GST Datagram Protocol (GDP) and Real-time Transport Protocol (RTP) headers are removed and the output is send to a H.264 video decoder. The output of H.264 video decoder is sent to videoconvert plugin which converts video from one colorspace to another. The output of videoconvert is sent to Motion JPEG encoder, and the output of this is dumped in a FIFO buffer (for real-time access in Python+OpenCV).

```
gst-launch-1.0 udpsrc port=9001 ! gdpdepay ! rtph264depay !
avdec_h264 ! videoconvert ! jpegenc ! filesink location=<file_loc>
```

Figure 4.4: GStreamer Receiver Code and Pipeline for Streaming H.264 Video from USB Webcam using UDP

4.4 GStreamer Pipelines for UDP H.264 Streaming from IP Camera

4.4.1 Sender Pipeline

The H.264 encoded video frames from IP Camera are read using rtspsrc plugin, where RTSP stands for Real Time Streaming Protocol. The output of rtspsrc is sent out to the specified receiver IP address host through the specified UDP port.

```
gst-launch-1.0 rtspsrc location=rtsp://10.156.14.101:554/live.sdp
latency=1 ! udpsink host=10.156.14.165 port=8090
```

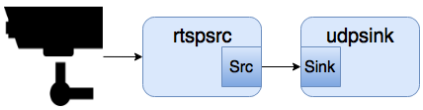


Figure 4.5: GStreamer Sender Code and Pipeline for Streaming H.264 Video from IP Camera using UDP

4.4.2 Receiver Pipeline

The video feed of the sender is accessed by the UDP receiver host by specifying the particular UDP port. The Real-time Transport Protocol (RTP) header is removed and the output is sent to H.264 video parser. The output of the H.264 video parser is decoded, which in turn is passed to videoconvert plugin to convert the video from one colorspace to another. The output of videoconvert is sent to Motion JPEG encoder, and the output of this is dumped in a FIFO buffer (for accessing in real-time in Python+OpenCV).

```
gst-launch-1.0 udpsrc port=8090 ! application/x-rtp, encoding-name=H264,
payload=96 ! rtph264depay ! h264parse ! avdec_h264 ! videoconvert !
jpegenc ! filesink location=<file_location>
```

Figure 4.6: GStreamer Receiver Code and Pipeline for Streaming H.264 Video from IP Camera using UDP

Chapter 5

Video/Image Processing

The main goal of the Video/Image Processing algorithm is to be able to dynamically recognize a space available for parking. This information, once obtained, can be relayed to a user through a web or mobile application who can then choose to avoid a road by observing the unavailability of parking and thus alleviate traffic congestion in the area.

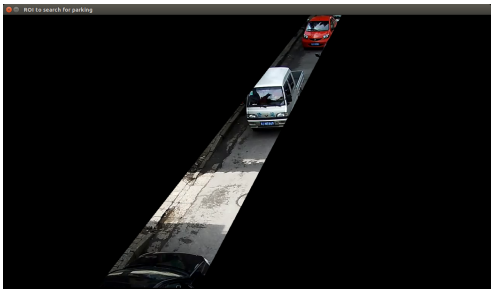
A surveillance camera is selected for this application, as using a radar sensor to find available parking space produces a large number of false positives given that it cannot differentiate between an obstruction and a car. A camera further provides a feed of visual data that opens up a large number of Computer Vision based future applications.

The first step in an algorithm to spot parking availability is to decipher where on a road a car can be legally parked. This process is trivial for the human mind as it differentiates between the main road and the areas available to parking by observing traffic patterns and other signs.

The initial approach built on this premise involved an administrator who can manually define the road where an incoming vehicle can park and this area would be continuously monitored for parking availability. As shown below, the admin would select the four points (Fig 5.1(b)) and a segmentation algorithm would attempt to find available space through histogram matching (Fig 5.1(d)).

(a) Original Video Frame

(b) ROI Defined using Four Marked Points



(c) Masking of Non-ROI regions

(d) Segmentation of ROI for Histogram Based Processing

Figure 5.1: Initial Image Processing Implementation

This approach, however, isn't accurate as it won't be able to differentiate between an object of constant color and a car and requires a technician to have information about parking availability on a specific road and further manually define the available road area.

Over time, this approach has been optimized to estimate the extent of the road by observing selective movement and dynamically learning the regions of legitimate parking using object detection. A distribution is further built to negate perspective skewing of observed objects. This method is completely automated and far more accurate as seen in Section 5.1

5.1 Algorithm

The Image Processing algorithm defined above requires human intervention to define the ROI. This drawback can be overcome by using a Video Processing algorithm that identifies ROI by first finding and then eliminating the region of road not parkable, which then allows for the parkable region to be filtered for further Processing.

The issue of differentiating objects of constant color from cars is done by using an object detection algorithm. This procedure can be explained using the following steps:

- Gaussian Mixture Model based background subtraction approach is used along with suitable thresholds for vehicle movement detection and tracking
- Areas of high road occupancy/movement are detected by using a probability distribution grid using on the centroids of detected blobs
- These regions are grown based on pixel intensity and distance from the grid points to find the entire region of road utilized by moving traffic and expanded to about half a car's distance width
- The final 'Region' obtained is used as a mask to XOR with the incoming frames to focus only on the section of the road suitable for parking
- Periodic car detections are done on the XOR'd image using the Caffe based Py-Faster-RCNN (Region-based Convolutional Neural Network)
- Centroid Distribution of these cars allows us to find the limits of the region for legitimate parking of cars (i.e. Region of Interest - ROI)
- Parts of ROI used for vehicle movement can be removed based on the mask obtained from GMM based region growing. The rest of the ROI is analyzed for the availability of parking space
- Car detections on the full incoming frame are used to divide the frame into vertical cross-sections based on average vehicle sizes
- The distribution of cars being parked in the unmasked region gives an estimate of the legal parking region
- Horizontal sections of the frame are defined to generate a distribution of car pixel area as a function of distance from the camera as and when cars are detected in the frame
- Based on the distance available between the centroid of two parked cars, and the sizes of cars in different cross-sectional regions, availability of space can be determined

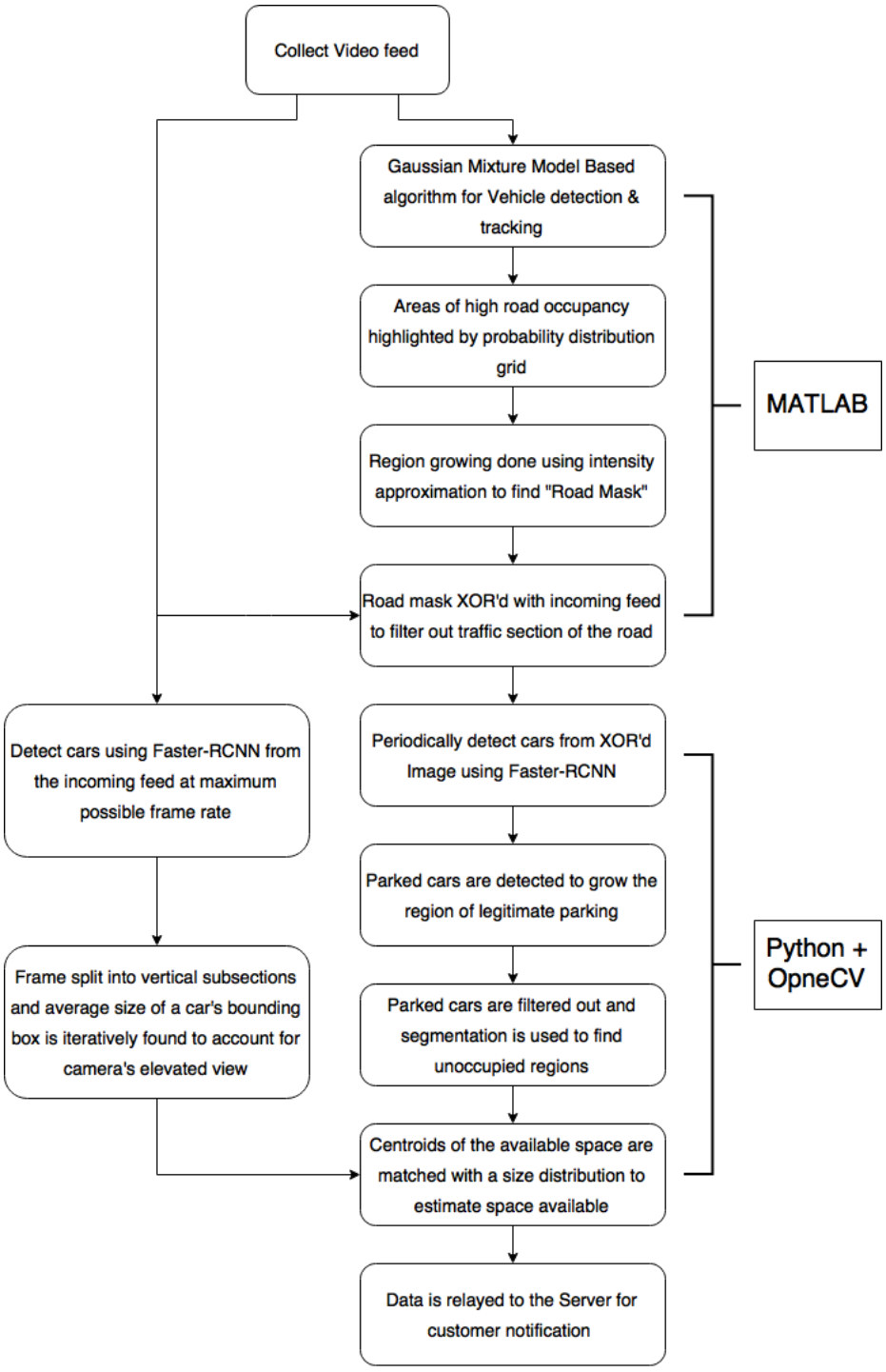


Figure 5.2: Flow of the Parking Detection Algorithm

5.2 Gaussian Mixture Model for Background Subtraction

One of the simplest approaches to maintain a background image, is as a cumulative average of the video stream and to segment moving objects by thresholding a per-pixel distance between the current frame and the background image. This method is the foundation of a collection of techniques generally known as *background subtraction*.

The algorithm based on a **Gaussian Mixture Model (GMM)** is representative of an adaptive method which uses a mixture of normal distributions to model a background image sequence. For each pixel, each normal distribution in its background mixture corresponds to the probability of observing a particular intensity or color in the pixel, here, the probability of it being either part of the foreground or background. This leads to a realistic and computationally efficient segmentation method. This Segmentation result can further be thresholded as needed to filter out different foreground objects.

5.3 Faster-RCNN based Object Detection

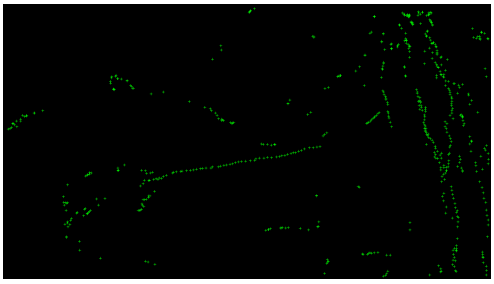
In a basic **Region-based Convolutional Neural Network (RCNN)** for an image, the region of interest (ROI) is first found. Then a warped image region is found, for each ROI, and forwarded to the Convolutional Network. Once each region has been forwarded to the network, bounding box regressors are applied and used for classification. In **Spatial Pyramid Pooling (SPP-net)**, the entire image is passed to the convolutional network which creates a feature map of the image. Regions of Interest are then formed using the map. Then this is passed on to the SPP layer and on top of that, bounding box regressors are applied.

Faster-RCNN takes qualities of both RCNN and SPP-net. It passes an entire image to the convolutional network to create a feature map. Then, the ROI is found out. On top of that a single layer of SPP is applied which is called the ROI pooling layer. Then it is connected with the fully connected layer. This makes the layer even below the ROI pooling layer (layers below the single level of SPP) trainable, allowing the implementation to reach real-time application speeds. Faster-RCNN is implemented on Python using the Caffe deep learning framework developed by the Berkeley Vision and Learning Center.

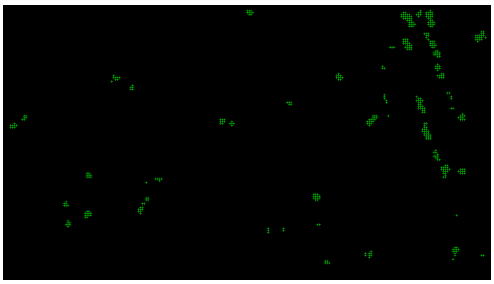
5.4 GMM and Faster-RCNN for Estimating RoI

Given the Indian conditions in which in which is project is to be deployed, a road cannot be simply defined by a grey region or by an area which isn't accessible to pedestrians. Hence we define the road as "an area where objects of a suitably large size move at a velocity greater or equal to 15 Km/Hr."
This area is about 1000-1500 pixels for a 720p image placed at a height of 25ft and 15 Km/Hr is taken as a standard speed limit for any average pedestrian walking.
The above assumption forms the basic threshold for the road detection model. Road detection is done using GMM. The GMM based algorithm is repeated multiple times. each pass being about 2000-5000 video frames and the entire process being 5 to 10 passes. The results of all passes are finally combined. In an iteration, the first fifty frames train the background, while the remainder of the video feed is used for processing. The bounding boxes obtained are then thresholded. These generate a distribution of centroids of moving objects (Fig 5.3(b)). A probability grid is used to reduce noisy pixels by increasing density around those points that have many neighbors (Fig 5.3(c)).

(a) Original Video Frame



(b) Centroids of all Objects that Passed Threshold

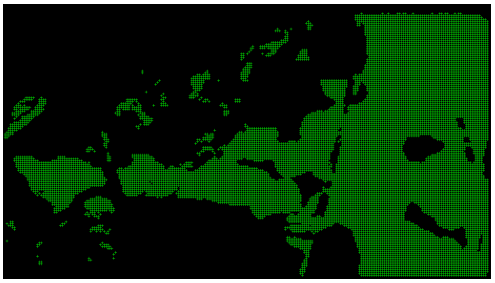


(c) Clean Points based on a Probability Density Grid

Figure 5.3: Background Subtraction and Object Tracking

The Road prediction algorithm is called once per iteration, and itself has 3-5 sub loops where it tries to learn the entire road. The algorithm waits a few frames between each loop, so as to give time of obstructing vehicles to move out of the way. The algorithm first determines the average intensity of all points that satisfied the threshold and got identified as a road. This is then used to find all surrounding points also within the same intensity range, but only within, say, a half-car distance from this mean, to implement a form of region growing. This makes use of a probability density grid. The score of each point in the grid increases by the presence of a large number of neighbors. High density points then pass the threshold for noisy detections and represent regions of high traffic. One pass of this loop would give a result (Fig 5.4(a)). Running this on the video for multiple passes would cover the entire region of the road used by moving vehicles (Fig 5.4(c)).

(a) Result of Intensity Growing over One Loop



(b) Result of Intensity Growing and Summing over Multiple Loops

(c) Resulting Road Regions Projected onto the Main Frame

Figure 5.4: Region Growing

The end result is a mask that gives the region of road used for travelling with good accuracy (Fig 5.5(a)). The mask can further be cleaned to fill gaps and smoothen edges to give a mask (Fig 5.5(b)). This mask is used to block off the motion in the road. The masked frame is then sent to the object detection algorithm.

(a) Mask Generated for the Road (b) Mask Obtained after Erosion

Figure 5.5: Mask Generation

The **Faster-RCNN** Algorithm is used to obtain the bounding boxes of all cars found in the frame. This can be applied to a masked frame in order to detect all vehicles in the parking region(Fig 5.6). This parking region consists of regions both legal and illegal for parking and hence, requires further processing to check for legitimacy. This is done using a **density grid** to predict regions of legal parking followed by lookup table based approach to eliminate **perspective skewing**. These procedures are further explained in the section below.

Figure 5.6: Object Detection Running on Masked Frame

5.5 Density Grid

As seen from Section 5.3, Faster RCNN consists of an object class that detects cars at an accuracy of close to 97%. This returns the centroid and size of a car. These detected centroids can be added to an array after the main road has been masked of by the GMM, which results in a distribution of parked cars. This results in a distribution (Fig 5.7(a)). It can be assumed that if a parking area is legal, then, the density of cars in that region will be higher. Therefore, by using a probability density grid that maps the area of legal parking based on the number of cars that are parked in the neighborhood

of each grid point, a distribution of legal parking is developed (Fig 5.7(b)). The ROI for legitimate parking can now be defined as approximately a half car size expansion around the central tendency of this grid, (Fig 5.7(c)).

(a) Expected Centroid Distribution of Parked Cars

(b) Probability Distribution removes Illegal Detections

(c) ROI Found based on Given Distribution

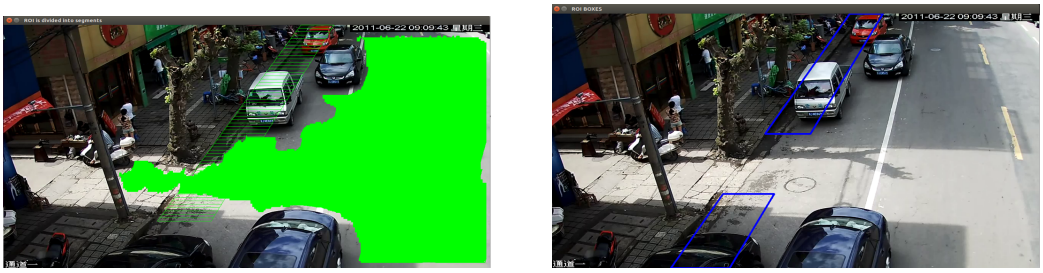
Figure 5.7: Density Equalization

5.6 Perspective Scewing

With a camera at a given height, the size of the car detected in the framediffers based on its distance from the camera. Therefore, to estimate the amount of pixel area occupied by a car and prevent perspective skewing, a distribution of car sizes can be created. The image frame is first divided into horizontal sections. These represent rows of the distribution table. A car detected in each section is added to the distribution table either as a sedan or as a hatchback, based on size. Each incoming car size is averaged with the corresponding column (hatchback or sedan column) based on its size correlation with the current value. Thus, after the training period in which hundreds of cars are detected, the average size of hatchback and sedan in each section of the frame will be available. This size distribution helps estimate the number of cars that can be parked based on pixel area.

5.7 Estimating Parking Availability

Once the region of interest has been obtained, the segmentation process checks for any overlap with high density road traffic area and any physical disturbance in each segment that prevents parking. To find if the segment overlaps with the moving road, the road-area mask in the segment is eliminated (Fig 5.8(a)), while physical disturbance is verified by checking if the region immediately around the segment has ever been used for parking. If there is no overlap with the mask and there have been vehicles parking in the segment previously, the segment is considered valid and combined with other valid segments to build regions where parking is possible (Fig 5.8(b)).



(a) ROI Segmentation

(b) Valid ROIs based on Ignoring Road in Previous ROI

Figure 5.8: Determination of Legitimate Parking Area

These regions are then cross-referenced to the current locations of cars and the distance between the boundary of the region and distance between cars within them is calculated to estimate available area. If the area between any 2 boundaries/centroids is greater than what is needed by a car in that horizontal section (size obtained from distribution), then parking area is available. These spots will be marked (Fig 5.9) and information regarding its location can be sent to a user through a suitable interface.

Figure 5.9: Identified Available Parking Spot

5.8 Real-Time Implementation using Python-MATLAB Bindings

The above video processing algorithm consists of two parts. Namely, the training algorithm implemented using GMM on MATLAB, relying on the object detection results from python based faster-RCNN, and a real time python based segmentation and object detection algorithm for finding free space on the road, both being executed on video feed from an IP camera.

MATLAB's inherent inability to communicate with an H.264 encoded video of the IP-camera ad utilize it in the Video Processing Toolbox requires intervention from python to facilitate the processing. As a result, the following sequence is followed to run the training algorithm on MATLAB using Python:

- The Py-faster-RCNN model is loaded and initialized using a python script
- The GStreamer pipeline used for reception of the IP camera feed is simultaneously fed to two outputs. One copy of the video is sent to a FIFO pipe for python processing. The other output is a file sink where the saved video can be used by MATLAB for processing
- The faster-RCNN object detection procedure keeps track of the numbers of objects of interest found in each frame, and, if the count is suitably high for many consecutive frames (i.e. there is considerable activity of vehicles on the road), the MATLAB script for GMM based road detection is called, with the average bounding box size as an input
- Before calling the MATLAB script, the incoming IP camera feed is saved for 10 to 15 minutes and this video file will be used by MATLAB
- The MATLAB script executes and saves the mask image in a predefined location while simultaneously deleting the saved video to conserve space
- The python script gains control yet again and continues real time execution of the Segmentation algorithm. Once the Parking ROI has been learnt based on the density distributions, the RCNN object detection can run in real-time to find parking spaces

5.9 MATLAB Engine API for Python

The MATLAB engine API allows for a MATLAB session to be started remotely from a python script. The engine can be setup from the root location of MATLAB on the system using the "setup.py" script provided. A python dependency, called "pymatlab" is then installed to allow transfer and sharing of variables between the python and MATLAB scripts. It also allows for MATLAB functions to be called from python and send python variables as input arguments to the function.

```
eng = matlab.engine.start_matlab()
x = matlab.double()
y = matlab.double()
z = matlab.double()
z = int(avg_box_size)
x,y,masked = eng.multiObjectTracking_roadDetector(z,nargout=3)
```

The above code creates a MATLAB session in python and defines three output variables in python and one input argument to MATLAB. The function call to road detection algorithm takes the expected bounding box size 'avg_box_size' as its input and returns the masked frame, as well as the mask image back to python.

```
maskim = np.zeros( (row, col,3), dtype=np.uint8)
for i in range(0,row):
    for j in range (0,col):
         maskim[i,j,0] = masked[i][j][0]
         maskim[i,j,1] = masked[i][j][1]
         maskim[i,j,2] = masked[i][j][2]

img  = Image.fromarray(maskim, 'RGB')
img.save('mask.png')
```

This mask received can be used for the python processing after converting to a Numpy array from the MATLAB matrix form, as shown by the script above.

Chapter 6

Project Highlights

6.1 Challenges Faced

The primary obstacles faced and their respective solutions are described in the following section:

- ***Inadequate Range*** **-** The internal Wi-Fi adapter on the Raspberry Pi 3 Model B failed to provide average pole-to-pole range (assumed to be 30m). The on-board adapters at best provided 10-15m range for direct transmission with enough bandwidth to sustain one single video stream. This undoubtedly would degrade in the case of multiple streams or multi-hop transmission. Also, as the in-built adapter operates on 2.4GHz, it is bound to combat interference from other devices which work on the same frequencies. This interference is rather large as most devices of current technology operate on the mentioned frequency. For a weak Wi-Fi adapter such as the Pi's, this poses a problem.
  An external TP Link adapter operating on 2.4GHz and a Panda adapter operating on 5GHz are attached to the Pi which have external antennas to extend range. The high gain adapters provides more than sufficient range to sustain multiple streams. Also, providing multiple adapters working on 2.4GHz and 5GHz facilitate the Pi using multilink optimization. The advantage of this method is that the Pi can now use the multiple interfaces to listen and send data accordingly making each node full duplex contradictory to the earlier node set-up.

- ***Static IP allotment & Non-Mesh Client Incompatibility*** **-** For a system with large number of nodes, it can be tedious to keep track of IP address allotments. Thus, it is necessary to make the addressing dynamic through a DHCP server.

Mesh clients are all nodes of the system that are part of the mesh network, essentially they are all nodes running the B.A.T.M.A.N protocol. In the case of raspberry pi and Linux based systems, it includes all systems that invoke the in-built B.A.T.M.A.N kernel. However, systems that do not have B.A.T.M.A.N support are devoid of connection to the network and hence fail to request for streams or data from the client-server system. If this problem is not catered to, we limit the type of systems that can connect to the network which is a big restriction on the proposed method.
The B.A.T.M.A.N. protocol allows for setting up an ad-hoc mesh network of Raspberry Pi nodes. The Network is also bridged on the Ethernet layer to allow connecting to DHCP Servers or Non-Mesh Clients (Support Windows, MacOS and Linux). A GStreamer Based Pipeline allows for streaming video from the Pi's to a server laptop using H.264 encoding and either TCP or UDP protocol, as required.
Gateway Server set-up includes a router (acting as a DHCP server) connected to the LAN and a pi declared as a gateway server. This pair takes care of IP address allotments spontaneously whenever a mesh or a non-mesh client joins.

- ***Human Intervention for ROI Specification & Indian Road Conditions*** **-** The original algorithm required specification of Region of Interest for parking spot detection. This method is inconvenient as, for a large system with several nodes, the marking will have to be done on every node and repetitively on reboot.
  Indian roads do not comply with ideal cases such as drivers abiding by lanes, pedestrians using the zebra-crossing or just vehicles and people on the road or the roads being grey. Thus, the road cannot be defined per se, but instead, will have to be learnt according to the location, crowd and weather.
  Video Processing techniques were used to model the road being occupied by cars for travelling. The algorithm is based on GMM based background subtraction and is able to find the region of road being used specifically of parking. Following this, Faster-RCNN based object detection allows for accurate growing of the ROI and identifying the regions of ROI not occupied by cars. A distribution based on the vehicle bounding boxes is used to map the available area to the approximate size of a car.

6.2 Novelty

- The cost effective wireless mesh network approach as opposed to many existing costly Ethernet or Optical fibre based systems makes the implementation more realizable from a financial perspective which in a country like India is very vital. The Government is looking for solutions to societal issues and Smarter Cities can provide immediate relief and also set up a framework for future Inter-networking. However, these solutions must be cost-effective.

- Plug-and-Play support for both mesh and non-mesh clients and automation of setup caters to hassle free operation, thus not necessitating intervention by an engineer or technician. The initial effort to set up systems isn't lacking in many Government projects, however, maintenance is an issue that is disconcerting. It is necessary to provide remote access and seamless extensibility to prevent the performance of the node to be hampered by the absence of constant attention.

- Dual band network allows for a multitude of applications based on varying data rates and coverage. Interference on one band of the network doesn't impact the performance of the system to a large extent because of the ability of the B.A.T.M.A.N protocol to dynamically switch between routing packets on different frequencies to optimize the network for its large bandwidth and stability requirement.

- Cameras, as sensors, provide for more accurate real-time video processing for road detection. This combination provides a framework for other surveillance applications. Our implementation is the only current solution that doesn't involve any ground-based sensors in its working. The usage of Cameras, over Radar, for complex road detection algorithms allows future application based on Computer Vision to use the existing infrastructure without any time-to-market.

6.3 Utility

- The proposed solution is almost completely wireless and thus reduces the NRE cost as opposed to one employing optical fibres. Optical Fibres are expensive and there is a need to reduce its consumption on a larger scale. Designing a system that is semi-wireless (wireless link between node and gateway server; optical link between gateway server and Remote Processing Server) allows the implementation of a hybrid technology that combines the best features of each style.

- Provides remote network and surveillance access to the administrator making it easier for access from an off-site location. This feature reduces the need for ground-based access and regular maintenance.

- Ease of maintenance and high extensibility; the network supports Windows, Mac and Linux Operating System based client nodes. Hence, this feature allows even a layman to avail the implementation's advantages.

Chapter 7

Learnings

- The entire project was done in the Linux environment, and hence we learnt several Linux commands, and became comfortable with an entirely new computing platform.
- First hand experience with BATMAN-ADV routing protocol helped us understand how ad-hoc mesh networks work, as well as their scope and importance.
- The boot script on each Raspberry pi for setting up the ad-hoc mesh network was written in Shell, which was a new programming language to us that we learnt on the fly.
- Video processing using Gaussian Mixture Models in MATLAB allowed us to get familiar with a powerful MATLAB toolbox, Computer Vision System Toolbox.
- Object detection using py-faster-rcnn enabled us to learn how convolutional neural networks work.
- Our knowledge in Python+OpenCV was strengthened through extensive programming on this versatile platform.
- The several networking tests we carried out in the IISc campus helped us understand the behaviour of the Batman-adv routing protocol.
- Finally, we realised the importance of proper documentation while working on such an ambitious project.

Chapter 8

Applications

- Implementation of a dynamic parking detection system tailored to non-ideal road conditions helps relieve traffic congestion as well as detect illegal parking in an Indian environment where existing solutions are bound to fail.
- The bridging facility in the system with proper API can allow for mobiles to send distress calls or emergency alerts over the network to personnel in-charge by which they can respond to dire situations immediately. This also acts as a preemptive measure to avoid such incidents.
- The network requires one or more routers acting as DHCP servers to allot IP addresses to the devices in the network, and these routers can provide internet access to any and all devices in the network. Thus, the realization can make available internet hotspots to other devices that join the network and can help towards digitization. Granting safe and legal access to such devices through authentication and verification by the administrator will be necessary.
- Data obtained from the camera feeds along with GPS location of the nodes allows for tracking of objects or people across multiple cameras to enable detection of illicit activities.
- Information from camera feeds can be used for traffic management to alleviate traffic congestion by rerouting traffic at hotspots of the city and also in data analytics to extract information such as vehicle count, crowd behavior, crowd and vehicle management and parking trends. Data Analytics for a restricted environment under private institutions such as in a mall can help them develop required amenities to cater to the masses.

- Multilink Optimization not only allows for interface alternating but also can handle multiple B.A.T.M.A.N interfaces. Thus, different interface layers spread over different areas can be managed by a single network enabling surveillance, tracking and monitoring various activities occurring in a large city under one roof of administration. This application can be particularly useful for governing bodies such as the Government of India.

- Audio support by GStreamer allows for applications where ambulance sirens could be detected to reroute traffic at signals or send emergency alerts to hospitals. Isolating the frequencies pertaining to sirens and detecting such frequencies will provide an efficient method to attend to emergencies which in normal conditions would take rather long to react to in congested traffic conditions.

- Video feed being the form of input, allows for possibilities of extracting information other than just regarding vehicles and parking availability, such as using feed for theft detection, crowd anomaly prediction, face detection for identification. The obtained information can be applied across various fields from Judicial Reinforcement to National Security

Chapter 9

Future Scope

- To explore the bindings between MATLAB and Python and port the entire algorithm to a single Integrated Development Environment (IDE) for convenience of implementing the final design.
- Test the implementation under different real-world working conditions with parameters such as Weather, Crowd Frequency and Wi-Fi interference impacting the performance.
- Test the network under erroneous working conditions in which efficiency would drop and design interrupt handling for each such condition.
- Explore the viability for cheaper alternates for each Hardware and Software requirement.

Chapter 10

References

[1] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks; Cornell University, 2016

[2] Tan Zhang, Aakanksha Chowdhery, Paramvir Bahl, Kyle Jamieson, Suman Banerjee: The Design and Implementation of a Wireless Video Surveillance System. At University of Wisconsin-Madison, 2015

[3] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan : Glimpse: Continuous, Real-Time Object Recognition on Mobile Devices. At Massachusetts Institute of Technology, 2015

[4] Kartik Bhargav under the supervision of Bharadwaj Amrutur : A report on Wireless Video Streaming in Ad-Hoc Mesh networks. At Indian Institute of Science, 2013

[5] Arnaud Loonstra: Videostreaming with Gstreamer. At Stichting z25.org, Leiden University, 2014

[6] Maria E Villapol, David Perez Abreu, Carolina Balderama and Mariana Colombo : Performance Comparison of Mesh Routing Protocols in an Experimental Network with Bandwidth Restrictions in the Border Router. At Central University of Venezuela, 2002

[7] https://www.open-mesh.org/projects/open-mesh/wiki: Set-up fundamentals

[8] https://github.com : Installation codes of project dependencies

[9] http://www.pyimagesearch.com: Image Processing Algorithms

[10] http://www.fastprk.com : FastPRK implementation

[11] http://www.sfpark.org : SFpark implementation

[12] http://www.cisco.com/go/smartconnectedcommunities : Cisco Smart Parking

[13] http://www.mobility.siemens.com/mobility/ : Siemens Smart City Vision

[14] http://amsmarterdamcity.com : Amsterdam Practical Trial implementation

[15] http://www.phoronix.com : Raspberry Pi 3 Benchmarks vs. Eight Other ARM Linux Boards, Michael Larabel, March 2016