DEVELOPMENT OF A SPATIO-TEMPORAL FRAME INDEXING ALGORITHM FOR IMPROVING THE QUALITY OF SERVICE IN LIVE LOW-MOTION VIDEO STREAMING

BY

Muyideen Omuya MOMOH, B.Eng. (FUTMINNA) 2014

P16EGCP8004

A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES, AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE MASTER OF SCIENCE (M.Sc.) DEGREE IN

COMPUTER ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING,

FACULTY OF ENGINEERING,

AHMADU BELLO UNIVERSITY,

ZARIA, NIGERIA

FEBRUARY, 2019

DECLARATION

I, Muyideen, Omuya MOMOH declare that this dissertation entitled "**Development of a Spatio-Temporal Frame Indexing Algorithm for Improving the Quality of Service in Live Low-motion Video Streaming**"has been carried out by me in the Department of Computer Engineering, Ahmadu Bello University, Zaria as part of the requirements for the award of the degree of Master of Science in Computer Engineering. The information derived from the literature has been duly acknowledged in the text and a list of references provided. No part of this dissertation was previously presented for another degree or diploma at this or any other institution.

Muyideen Omuya MOMOH Signature Date

CERTIFICATION

This dissertation entitled DEVELOPMENT OF A SPATIO-TEMPORAL FRAME INDEXING ALGORITHM FOR IMPROVING THE QUALITY OF SERVICE IN LIVE LOW-MOTION VIDEO STREAMINGby Muyideen, Omuya MOMOH meets the regulations governing the award of the degree of Master of Science (M.Sc.) in Computer Engineering of the Ahmadu Bello University and is approved for its contribution to knowledge and literary presentation.

Dr. E. A. Adedokun	_____	_____
(Chairman, Supervisory Committee)	Signature	Date
Dr. M.B Abdulrazak	_____	_____
(Member, Supervisory Committee)	Signature	Date
Prof. M. B. Mu'azu	_____	_____
(Head of Department)	Signature	Date
Prof. S. Z. Abubakar	_____	_____
(Dean, School of Postgraduate Studies)	Signature	Date

DEDICATION

This work is dedicated to all members of Computer Control Research Group, ABU Zaria

ACKNOWLEDGEMENT

All praises and adoration belong to Allah, the creatorof the universe. May the peace and blessings of Allah be upon the noblest Prophet Muhammad (SAW).

Foremost, I would like to express my sincere gratitude to my supervisors, Dr. E.A. Adedokun and Dr. M.B. Abdulrazak for their patience, motivation, enthusiasm, constant drive, immense knowledge. Their immeasurable guidance helped me shape the research problem and constantly provided me insight towards the research and writing of the dissertation. I feel so lucky and honoured having such a wonderful combination of supervisory committee.God bless you sirs.

My sincere appreciation goes to the Head of the Department, Prof. M.B. Mu'azu for his contributions and fatherly love. Thank you so much sir for the constant reminder of the need to complete this research work timely.

My appreciation also goes to Dr. I. J. Umoh, Dr. A.T. Salawudeen, Dr. B. O. Sadiq, Dr. Y. Basira, Engr. S.Y. Muhammed and Engr Y.Ibrahimfor all the research discussion, motivation and assistance.

I am thankful to all the lecturers of Computer Engineering Department, Ahmadu Bello University, namely; Dr. T.H. Sikiru, Dr. Y.A. Sha'aban, Dr. H. B. Salawu,Dr. I. A. Bello, Mrs Z.M Abubakar, Mr. H. Zaharadeen, Mr. A. Umar, Mr. O. Ajayiand Mr. S. Y. Ibrahim for their kind advice and motivation. Also to Dr. F.O Anafi for your wonderful contribution.

My profound gratitude to Prof. S.O.E. Sadiku, Prof A.M. Aibinu,the entire staff of Computer Engineering Department FUTMINNA, my spiritual leaders; Shaykh Umar Dada Paiko, Mallam Bashir Usman Yankuzo, my friends and colleague whom I called my brothers, Obari

Johnson,Umar Idris, Hudu Magaji, Saliu Shaba, Lukman Abdulquadri, Adekale Abdulfatai, Martins Bosun, Abdulkabir Abdulrazak, Ahmad Suleiman, Mohammed Anasand the entire class of P15 and P16 sets of Computer Engineering Department. Thanks for the support.

Last but not least, special thanks and deepest appreciation to my parents: Mr and Mrs Momoh for their endless love, unconditional support, kind advice, understanding and prayers. To my step mothers; Ummu Furqan and Ummu Shifa'u, and my siblings; Abdulwaheed, Khafila, Furqan, Shekira, Jemila, Shifa'u, Sidika and Mufidah. I love you all.

I am really grateful to everyone, for the support and love, May Almighty Allah bless you all.

Muyideen Omuya MOMOH

February, 2019.

ABSTRACT

This research presents the development of a spatio-temporal frame indexing algorithm for improving the Quality of Service (QoS) in live video streaming for low-motion videos. Live streaming refers to the process of delivering an events over a network in real time. Live streaming is difficult because the network offers best effort service. In order to guarantee effective QoS in low-motion videos, frame indexing algorithms have been proposed by researchers. These algorithms were implemented in the server and client's modules where the indexing and reconstruction were performed respectively. However, despite the achieved improvement in QoS, there exists significant redundancy of pixels in the generated difference buffer. Hence, optimal utilization of bandwidth is difficult to be achieved. In order to address this challenge, a spatio-temporal frame indexing algorithm is hereby developed. This algorithm works by eliminating redundancy that exists within and across frames. The performance of the developed spatio-temporal frame indexing was evaluated using the standard low-motion video and a local low-motion video as scenarios. Results were compared with the result obtained on the standard frame indexing using buffer size, compression ratio and frame built time as performance metrics. In terms of buffer size, the results show that the spatio-temporal frame indexing algorithm outperform the standard frame indexing by 5.13% and 15.8% using the standard low motion and local low motion video respectively. It also shows an improvement in compression ratio by 5% and 15.6% using the standard and local low motion video respectively. However, in terms of frame built time, the standard frame indexing outperforms the developed scheme by 10.8% and 8.71% in the respective standard and local low motion videos.

TABLE OF CONTENTS

CHAPTER TWO:LITERATURE REVIEW

CHAPTER THREE:MATERIALS AND METHOD

CHAPTER FOUR:RESULTS AND DISCUSSION

CHAPTER FIVE: SUMMARY, CONCLUSION AND RECOMMENDATION

LIST OF FIGURES

LIST OF TABLES

LISTS OF APPENDICES

LIST OF ABBREVIATIONS

Acronym	Definition
AP	Access Point
AVC	Advanced Video Coding
CD-ROM	Compact Disk Read Only Memory
DDS	Data Distribution Service
DNN	Deep Neural Network
GCMR	Guaranty Mechanism of Contingency Resource
HDTV	High Definition Television
IP	Internet Protocol
IPTV	Internet Protocol Television
ISO	International Organization of Standardization
ITU-T	International Telecommunication Union-Telecommunication
MPEG	Moving Picture Expert Group
P2P	Peer to Peer
PON	Passive Optical Network
PSTN	Public Switch Telephone Network
QoE	Quality of Experience
QoS	Quality of Service
RSSI	Receive Signal Strength
RTCP	Real Time Control Protocol
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
TCP	Transmission Control Protocol
TIPHONE	Telecommunication Internet Protocol Harmonization over Networks
UDP	User Datagram Protocol
VAST	Video Adaptive Streaming
VCR	Video Cassette Recorder

VGAST	Video Adaptive Streaming
VGAST-PB	Video Greedy Adaptive Streaming
VHS	Video Home System
VoD	Video on Demand

CHAPTER ONE

INTRODUCTION

1.1 Background

A video consists of multiple images called frames taken quickly over a period of time(Apostolopoulos *et al.*, 2002). In the past, video was captured and transmitted in analog form. Digitization of video became possible due to the advent of integrated circuit and as such digital video enable a revolution in the compression and communication video(Apostolopoulos *et al.*, 2002). In the early 2000s, the internet saw a booming escalation of network bandwidth in combination with an algorithm for compression and more powerful computer system, video streaming became possible (Zou *et al.*, 2015). The commercial video streaming applicationssuch as QuickTime, ActiveMovie, and RealPlayerbecame reality almost 35years after the internet was born (Zou *et al.*, 2015). Large computing and bandwidth requirementsassociated with video encoding and transmission are factors which hinder the development of video streaming(Kumar & Kumar, 2016). Streaming refers to transport of live or stored mediasuch as audio, video or any associated data over a network. A lot of images are required to create an illusion of continuity when transferring videos over the internet. The bandwidth requirement was just too high for the early days of the internet even for low resolution videos (Taksande *et al.*, 2015). In reducing the number of bytes required per frames, compression is being employed. However, the computational complexity of compression grows with number of bytes that can be saved. Thus, and therefore, powerful computers are needed.

Live video streaming refers to the real time transport of live content over the internet i.e. the media is captured, compressed and transmitted on the fly (Kumar & Kumar, 2016). A lot of

content providers now broadcast their media content live on the internet. The application of video streaming are numerous which include internet broadcasting, video conferencing, video lectures amongst others.Such applications provide end users convenient, abundant and interactive multimedia service (Zhang *et al.*, 2015). In 2014, more than 64% of the internet traffic resulted from datastreaming(Azhar *et al.*, 2016). Time constraint and variable bit rate property make video streaming application difficult, this is because the buffering time of video streaming depends on the bandwidth offered by the network(Apostolopoulos *et al.*, 2002).

Network congestion is the main cause of video degradation which occur due to heavy traffic, low bandwidth and delay in delivery, which in turn leads to poor video quality (Kumar & Kumar, 2016). Over the years, researchers are paying attention to higher demands of streaming due to the increasing needs of offering videos on the fly. Several algorithms have been employed in the compression techniques so as to reduce the amount of data to be transmitted over the network which in turn improved the bandwidth utilization and also saved transmission cost (Azhar *et al.*, 2016).

The indexing based streaming algorithm attempts to also improve the compression rate by sending a reference frame, the difference buffer and indexing information that is required for the successful reconstruction of a new frame(Kumar & Kumar, 2016). This in turn improved the bandwidth utilization and save transmission cost. However, despite the use of the index based streaming algorithm, the difference buffer is still observed to have redundancy of pixels (spatial redundancy) which can be further reduced by considering the spatial and temporal redundancy simultaneously.

1.2 Significance of Research

Video streaming has received a lot of attention from industry and academia due to an explosive growth of the internet and increasing demand for multimedia information on the web. However, the current best effort service does not guarantee effective QoS in video streaming. One of the challenging issues which still remain a bottleneck to researchers is congestion which occur due to heavy traffic, low bandwidth and delay in delivery which in turn leads to poor video quality. Thus this research aims to develop a spatio-temporal frame indexing algorithm for improving the QoS in live video streaming for low-motion videos. The developed algorithm exploits both the spatial and temporal redundancy that occur in video streaming (low-motion videos), hence, accounting for optimal bandwidth utilization. Thus, factors such as delay in delivery, resolution degradation and packet loss that adversely affect the QoS of live streaming can be reduced to the barest minimum even at low bandwidth.

1.3 Statement of Problem

Live video streaming over the internet is difficult because the internet offers best effort service. Frame indexing method has been proposed by researchers to improve the QoS in low-motion video. However, the redundancy of pixels in the generated difference buffer can never guarantee optimal utilization of bandwidth. There is, therefore, a need to develop an algorithm that exploits both spatial and temporal redundancyin frames so as to reduce heavy traffic andeliminate redundancy in order to achieve a higher compression efficiency at optimal use of bandwidth. This work is aimed at developing a spatio-temporal frame indexing algorithm for improving the QoS in live low-motion videos. Factors such as packet loss, delay in delivery, resolution degradation, heavy traffic that adversely affect the QoS in video streaming can be avoided even when the network bandwidth is low.

1.4 Aim and Objectives

The aim of this research work is to develop a spatio-temporal frame indexing algorithm for improving the quality of service in live low-motion videos. To achieve the stated aim, a number of objectives have been identified as follows:

i. To implement the standard temporal frame indexing algorithm by exploiting similarity in successive frames.

ii. To develop the spatio-temporal frame indexing algorithm by considering correlation in adjacency pixels in a frame and in successive frames.

iii. To simulateand compare (i) and (ii) based on the following: Buffer size, Compression ratio and frame built time.

1.5 Methodology

The step by step procedure adopted to achieve the proposed spatio-temporal frame indexing algorithm for improving the QoS in live video streaming are highlighted as follows:

i. To implement the standard temporal frame indexing algorithm by exploiting similarities in successive frames.

   a) Capture first frame, compressed and transmit (as reference)
   b) Capture next frame and do temporal redundancy check
   c) Generate difference pixels based on the temporal redundancy check
   d) Store the difference pixels in a buffer (Difference buffer)
   e) Send the difference buffer along with the indexing information required for the reconstruction of the new frame

f) Clients reconstruct the new frame based on the reference frame, difference buffer and indexing information and set the new frame

g) Repeat (b) to (e) until streaming ends

ii. Implementation of the spatio-temporal frame indexing algorithm

a) Repeat i (a)

b) Exploit both the spatial and temporal redundancy

c) Generate difference buffer based on ii (b)

d) Repeat i (d) to (f)

iii. Simulation and performance comparison of objective (a) and (b) using buffer size, compression ratio and frame built time.

1.6 Scope of Study

The scope of this research which developed a spatio-temporal frame indexing algorithm that exploit the redundancy in a frame and inter frames in live video streaming only considered a low motion video.

1.7 Dissertation Organization

Chapter one presents the general introduction of this work. The rest of the chapters are structured as follows: Chapter two gives details of the reviews of related literature and relevant fundamental concepts about frames, compression, video streaming and streaming protocols. Chapter three presents the methods for developing the standard frame indexing algorithm and the spatio-temporal indexing algorithm. The analysis, performance and discussion of the results are given in Chapter four. Chapter five comprises the conclusion and recommendation for further work. At

the end of the dissertation, the list of the cited references and MATLAB codes are provided in the appendices.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

The literature review consists of the review of fundamental concepts that are related to this work and the review of similar works.

2.2 Review of Fundamental Concept

In this subsection; fundamental conceptsthat are related to this worksuch as video frame, compression, streaming and streaming protocols among others are reviewed.

2.2.1 Video frame

Video frames (frame) are individual pictures in a sequence of images. This frames are what make up a video. A second video may contain about thirty frames. In video processing techniques, videos are being compressed using different algorithms. These algorithms are referred to as frame type or picture type. Each algorithm hasits advantages and disadvantages. These algorithms are Intra-coded frame (I-frame), predicted frame (P-frame) and bi-directional predicted frame (B-frame) (Kumar & Kumar, 2016).

i. **I-frame:** These types of video frames are used as reference for other types of frames. They are complete picture (MPEG, PNG etc). It does not require any other frame to decode; however, they are least compressible when compared to other types of frames.

ii. **P-frame:** The predicted frame are also known to as delta frame. In order for the frame to decode, it uses data from previous frame. The P frame only holds the part that changed from the previous frame. For example, in two successive frames, only the part that

changes in the first frame need to be encoded. It needs not to encode the unchanging part, thereby, saving more space.

iii. **B-frame:** The bi-directional predicted frame uses the previous and forward frame in order to decode. It is highly compressible when compared with I-frame and P-frame as it saves more space.

2.2.2 Compression

Compression is an important aspectof computer and communication which is employed in transferring data from the source to destinations. In the internet, transferring of data is time dependent, so the use of compression is necessary in order to reduce the amount of data that needs to be transferred. Compression in computer terms simply means reducing the physical size of data so that itoccupies less storage space.Compressedfiles are therefore, easier to transfer due to significant reduction in the size of data to be transferred. Hence, it also leads to reduction in time needed as well as reduction in the bandwidth utilizationin transferring data, thus providing good video quality even over a slow network (Mittal & Vetter, 2016)

2.2.2.1 Video compression

Video compression is the process of reducing the frame size so as to save cost of transmission, increase transfer speed and data compatibility with the internet. Video compression can be achieved by exploiting the similarities in intra frame or inter frame(Mittal & Vetter, 2016). In a single frame there is spatial redundancy (Kumar & Kumar, 2016) which exist due to high correlation that exist in a frame as the amplitude of nearby pixels are often correlated. More so, successive frames exhibit temporal redundancy (Kumar & Kumar, 2016) since they typically contain same object. The ultimate goal of video compression is to reduce redundancy.

2.2.2.2 Video compression standard

Video compression standards provide several benefits. It ensures interoperability and communication between encoders and decoders from different manufacturers. For these, it reduce the risk for both manufacturers and consumers, which leads to quicker acceptance and widespread use. In addition, these standards are designed for a large variety of application and the resulting economics of scale lead to reduced cost and further widespread use. There are two families of video compression standards. They are, the International Telecommunication Union-Telecommunication (ITU-T) and the international organization for standardization (ISO)(Taksande *et al.*, 2015).

i. International telecommunication union- telecommunication (ITU-T)

The first video compression standard to gain popularity under the ITU-T was the ITU-H.261. It was designed for video conferencing over the integrated service network (Taksande *et al.*, 2015). In 1990, it was adopted as standard. It was designed to operate at p=1,2,3,……30 multiples of the baseline ISDN data rate. The ITU-T initiated a standardization effort in 1993, with the goal of video telephony over the public switch telephone network (PSTN) where the overall data rate was about 33.6kb/s. The H.263 was the video compression standard portion of the standard and it was firstly adopted in 1996. An enhanced H.263 was developed in 1997, which is known as H.263 version 2. Now, a new algorithm, formerly referred to as H.26L is currently being finalized as H.264/AVC(Taksande *et al.*, 2015).

ii. International organization for standardization (ISO)

The ISO established the Moving Picture Expert Group (MPEG) in 1998. The standard was developed with the aim of compressing moving pictures (video) and audio on digital storage

media (CD-ROM). In 1991, they finalized the MPEG-1and it achieve approximately VHS quality video and audio at about 1.5Mb/s(Taksande *et al.,* 2015). The second phase known as MPEG-2 which is an extension of MPEG-1 was developed for digital television and for higher bit rates (Apostolopoulos *et al.,* 2002).MPEG-3 was envisioned for higher bit rate application such as HDTV but was wrapped into MPEG-2 because it can also recognized such application that is why MPEG-3 never gain popularity. In order toattain more functionality such as higher compression efficiency, error resilience features, natural and synthetic content, content based interactivity etc. The joint Video Team from both ITU and ISO MPEG finalized on H.26L standard. It was adopted by the two teams and they called it H.264 and MPEG-4 part 10, Advanced Video Coding (AVC)(Taksande *et al.,* 2015). The emerging and current video compression standard is given in Table 2.1

Table 2.1: Emerging and current video compression standard(Taksande et al., 2015)

Video coding format	Primary intended application	Bit rate
MPEG-1	Storage of digital video (CD-ROM)	1.5Mb/s
MPEG-2	DTV	2-20Mb/s
MPEG-4	Video streaming, synthetic content, interactivity, object based coding	Variable
H.261	Teleconferencing over ISDN, Video over IP	64kb/s
H.263	Video telephony over PSTN	33.6kb/s and up
H.264/MPEG	Improved video compression	10's to 100's of kb/s

The emerging and current video compression standards are shown in Table 2.1. Currently, MPEG-4 and H.263 V2 are now used for video streaming and video communication and the emerging H.264/MPEG-4 part 10 AVC will probably gain more popularity. There is also an

incoming format known as H.265 which is still at an infant stage that aims to achieve high compression efficiency and lower the computational overhead in the process of video compression.

2.2.2.3 Compression algorithm

The aim of compression algorithm is to reduce the number of parameters that is used to represent a signal, so as to lower the transmission cost, achieve a high compression efficiency, compatibility with the internet while delivering reasonable quality picture to the user(Kumar & Kumar, 2016).

Basically, there are two major types of redundancy that are present in videos, they are:-

i. Spatial redundancy(Intra frame redundancy): This is the type of redundancy that exist within a frame. Usually, in a frame, there exist similarities among the pixels. Exploiting the spatial redundancy will definitely lessen the amount of data that is to be encoded and transmitted. However, in a coarse image, spatial redundancy are ineffective (Ponlatha & Sabeenian, 2013).
ii. Temporal redundancy (Inter frame redundancy): This type of redundancy exist in corresponding pixels within successive frames. Higher compression is being achieved by exploiting the similarities within successive frames. However, in a high motion videos, temporal redundancy may be ineffective (Ponlatha & Sabeenian, 2013).

In spatio-temporal redundancy, both the spatial and temporal correlations are exploited simultaneously. This will further reduce the amount of data/information that is to be encoded and transmitted.

2.2.3 Streaming

Streaming means real time transport of live or prerecorded media (audio, video, and any associated data) over the internet between clients and server computer (Kumar & Kumar, 2016). Basically, it is classified into live streaming, pseudo streaming and video-on-demand(Austerberry, 2005).

2.2.3.1 Live streaming

Live streaming is a type of application that delivers live program over the internet in real time (Zhang *et al.,* 2015). It involves the camera for the media capture, an encoder to digitize the content and a content delivery network to distribute and deliver the content. The information travel in a form of stream of data from the server to the client. The decoder is a plugin or a standalone player that works as part of web browser (Kumar & Kumar, 2016). The information stream, decoder and the server work together to let people watch live telecast or broadcast. Thus, live streaming over the internet is difficult because the internet only offers best effort services. That is, there is no guarantee on delivery (Gangurde & Nikam, 2017).

2.2.3.2 Progressive download or pseudo streaming

Progressive download or pseudo streaming is a type of streaming in which some fraction of the received files starts playing (Kumar & Kumar, 2016). It is a fast start streaming, that is, a cross between conventional streaming and downloading. The media starts playing just after the start of the download. If the internet connection speed is faster than the media data rate, the user can watch the video while it is downloading. For the user, it looks like streaming but in effect, it is downloading. The information that the player needs to start playing the video is stored at the beginning of the file. The media content is interleaved throughout the file in such a way that parts that lay together are located together in the file. Progressive download is like watching a

program on TV but also recording it on your video cassette recorder (VCR), after it has been broadcast and taped,you can watch it again and again.Table 2.2 shows the comparison between streaming and downloading.

Table 2.2: Comparison between streaming and downloading(Kozamernik, 2011)

Requirement	Live Streaming	Progressive (Downloading)
Streaming server	Required	Not required, web server is sufficient
Protocol	UDP/IP	TCP/IP
Packet loss	Has tolerance for packet loss	No packet loss
Delivery time	Real time	May retransmit loss packets, which result to slow delivery
Delivery quality	In other for the time/bandwidth requirement to be met, packet may be jettisoned	No loss of packet
Playback	Immediately when playback start	Playback starts when all (in progressive: sufficient) file has been downloaded
Storage requirement	No	Files are saved in the user's PC

Table 2.2 shows the comparison between streaming and progressive downloading. The streaming and the downloading technologies have their advantages and uses, and should be used for different purpose. Streaming is most appropriate for delivering live event. Pseudo-streaming or progressive downloading is suitable pre-recorded or VoD.

2.2.3.3 Video on demand

Video on demand (VoD) are systems that allow users to select and watch video content of their choice on their TVs or computer rather than having to watch at a specific broadcast time. VoD is one of the futures offered by the internet protocol TV (IPTV). With VoD, users can access a wide range of videos on the internet such as entertainment, educational program, sports and features film. The transmission mode of VoD is unicast (Parodkar & Bade, 2015). Some of the features of VoD are as follows:

i. The video and audio are of high quality
ii. It is more costly than live stream
iii. It is compatible with computers and smart phones

2.2.4 Architecture of live video streaming

Video streaming architectures are classified based on the communication of the machines, which are peer to peer and client-server(Mahini *et al.*, 2017).

2.2.4.1 Peer to peer architecture

The peer to peer (P2P) streaming is one of the most popular internet application over the years. The aim of a P2P is to reduce the workload on the server and provide a scalable content distribution (Gangurde & Nikam, 2017). Current deployment of the P2P on the internet shows that it is capable of streaming video to a large user population at low server cost and with little infrastructure(Sasi & Madhavu, 2014). However, the user quality of experience in current P2P streaming systems are not comparable to the conventional TV services provided by cable and satellite broadcasting companies(Liu *et al.*, 2008). Also, they experience long startup and

channel delays. Video playback starts tens of seconds after a user select a channel. There are also large playback lags among peers. Some peers watch frames in minutes behind other peers. Due to the limited peer uploading capacity, most P2P streaming systems only support video rate to up 400kbps (Liu *et al.*, 2008). In P2P networks, all nodes act as clients as well as server, that is, there is no need for a dedicated server(Mahini *et al.*, 2017). Thus, it is easier to setup and it is less expensive. Based on overlay network structure, P2P can be broadly classified into two categories which are; tree based and mesh based (Liu *et al.*, 2008).

The tree based architecture have well organized overlay structures and it distributes videos by pushing data from a peer to its children peers. The major drawback of tree based streaming is that they are vulnerable to peer churn (i.e. a peer departure will temporarily disrupt video delivery). In contrast, the mesh based peer to peer streaming architecture dynamically connects to a subset of random peers in the system. Peers periodically exchange information about their data availability(Parodkar & Bade, 2015). Video content is pulled by a peer from its neighbor who have already maintained the content. Mesh based video streaming are highly robust to peer churn(Liu *et al.*, 2008), since multiple neighbors are maintained at any given moment. Figure 2.1 shows the tree base P2P streaming architecture.

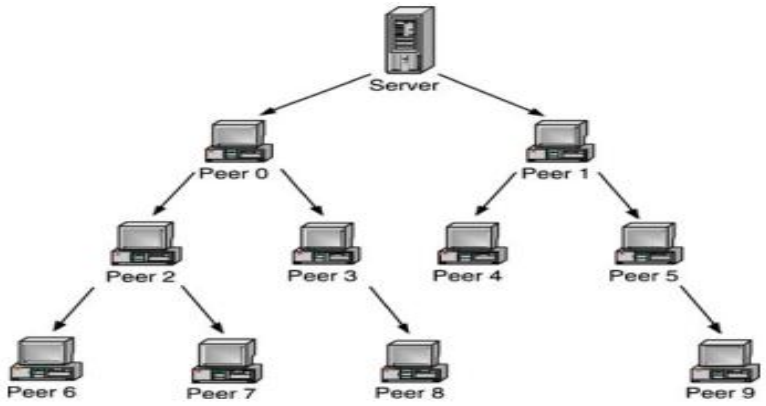


Figure 2.1 Tree based P2P streaming architecture(Liu *et al.*, 2008)

The tree base P2P streaming architecture is illustrated in Figure 2.1. In this architecture, a peer has one parent in a single streaming tree and it downloads the video stream directly from the parent. The streaming topology in P2P is static as such it introduces a single point of failure(Mahini *et al.*, 2017). When a peer depart, the peer and as well as its descendant cannot get stream until it connect back to another parent(Liu *et al.*, 2008). This is why they are vulnerable to peer churn. However, P2P is easy to implement and it has short latency of delivery. Figure 2.2 shows the mesh based streaming architecture.

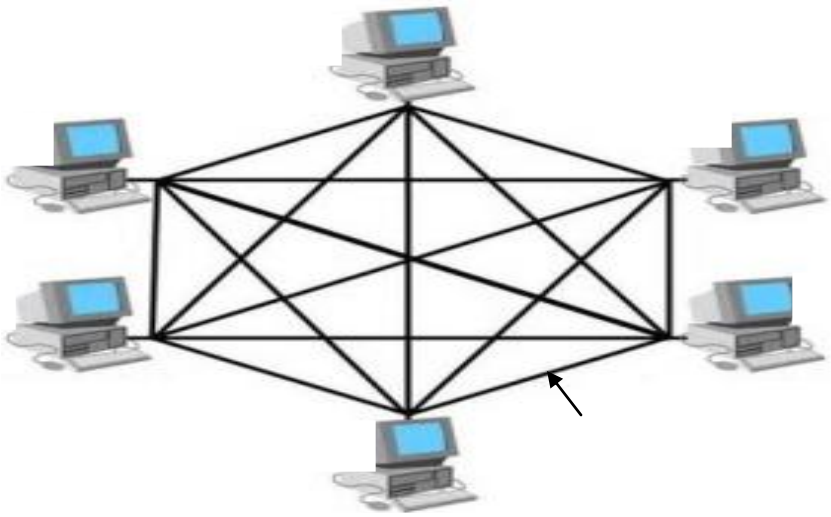


Figure 2.2 Mesh base P2P streaming architecture(Liu *et al.*, 2008)

As illustrated in Figure 2.2, a peer maintain multiple relationship with peers at any given time. A peer can simultaneously download or upload videos to neighbors. If a peer departs, the peer can still download video content from the remaining neighbors(Mahini *et al.*, 2017).The high peering degree in mesh basedarchitecture make it churn resilient.

2.2.4.2 Client and server architecturereduce space after paragraph

The client/server consists of a dedicated server and clients. The user's PC (client) is the requesting machine and the server is the supplying machine and both of them are connected through a network such as the internet (Zhang *et al.*, 2015). Figure 2.3 illustrates the client and server architecture.

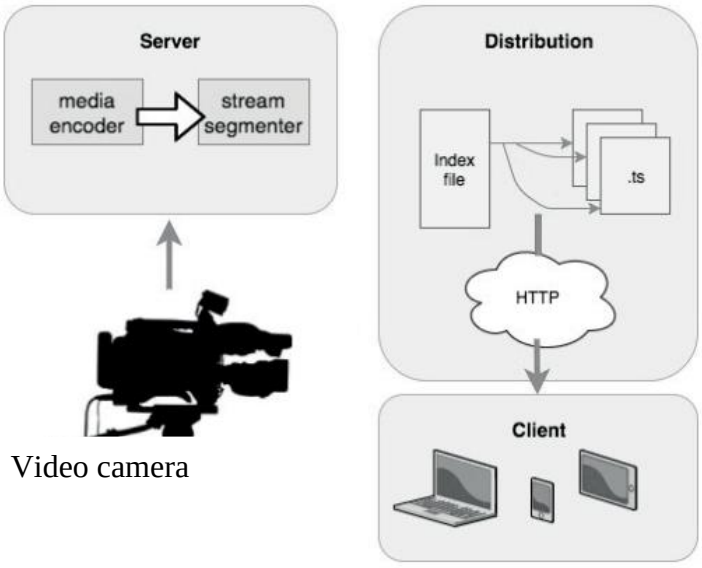


Figure 2.3 Clients and server streaming architecture(Zhang *et al.*, 2015).

The client and server architecture comprises of the three major component which are the server, distributor and the clients as illustrated in Figure 2.4

i. **Server:** The server handles input stream of media and encodes them digitally, encapsulates them in a formsuitable for of delivery and prepare the media for distribution.

ii. **Distributor:** The distributor consist of standard web servers. They handle accepting client request and deliver the prepared media and dedicated resources to the client.

iii. **Clients:** The clients request and download all the files and resources, it assembles them so that they can be presented to the user as a continuous flow video. At first the software downloads the index file from the universal resource locator (URL) and several media files available. The playback software reassembles the sequence to allow continuous display to the user.

2.2.5 Video streaming protocols

The streaming protocols are protocols used to transmit video as data packet over the internet and the IP network, such as the Real-time Transport Protocol (RTP), Real Time Control Protocol (RTCP) and Real time streaming protocol (RTSP)(Santos-González *et al.*, 2017). The different types of video streaming protocol are presented as follows.

2.2.5.1 Real-time transport protocol (RTP)

This protocol was developed basically for the purpose of streaming across the internet. It is the most important streaming protocol. All streaming media, irrespective of their contents and formats are all encapsulated in RTP packets(Santos-González *et al.*, 2017). It also provides several data field such as time stamp and a sequence number which are not present in the TCP(Kozamernik, 2011). It uses its multiplexing checksum functionalities as it runs on UDP. It is flexible, as such, it allows control of the media server thereby, allowing video stream to be

served at a correct speed. The media player reassembles the RTP packet orderly and play them at an appropriate speed. Transmission of packet using the RTP is in real-time, as such, it does not retransmit lost packets. Error concealment and packet replication strategies can be employed by client software to handlemissing bit. The transmission breaks when the connection speed is lower than the data rate which will cause the media to play poorly(Kumar & Kumar, 2016). The extra bandwidth remains untouched when the connection is fast, as such, the server load depends on the number and bandwidth of the stream.

2.2.5.2 Real time control protocol(RTCP)

This streaming protocol is used alongside with the RTP. It uses TCP for client/server connection. The service provider receive feedback from the RTP on each participant in the session. The received message includes, the total number of lost packets, jitter statistics, etc. The information received is used by some application when controlling and improving the session and transmission(Santos-González *et al.*, 2017).

2.2.5.3 Real time streaming protocol(RTSP)

This is an application level protocol for controlling real-time streaming data. It makes use of the RTP for data delivery(Kozamernik, 2011). It also helps the server to adjust the media bandwidth to the congested path to suit the available capacity. It has the ability to choose the optimum delivery channel to the client.

2.2.6 Quality of Service

Quality of service(QoS) refers to the ability of network to achieve maximum bandwidth and other network performance parameters such as delay, Jitter, packet loss, throughput etc.(Azhar *et*

al., 2016). The performance parameters have become a challenge in internet protocol (IP) based network. The fundamental of QoS is to meet the needs of various services, using similar infrastructure. With QoS, the attribute of the services provided can be defined both qualitatively and quantitatively(Zhang *et al.*, 2015). The performance of a network can vary due to several problems, such as latency, bandwidth, packet loss and jitter which may result to quality degradation in video streaming(Kumar & Kumar, 2016). For example, in voice communication (such as VoIP or IP telephony) as well as video streaming, user can get frustrated when the application data packets arestreamed over the network with insufficient bandwidth, excessive jitter or unpredicted latency(Kumar & Kumar, 2016).Some of the QoS parameters are as follows(Azhar *et al.*, 2016).

i. Delay

The delay in streaming network refers to the total time a frame takes to transverse from server to clients(Azhar *et al.*, 2016). The delay of access, hardware latency and delay in transmission. The transmission delay is the delay that is mostly experienced by the transmitted traffic (Lahbabi *et al.*, 2014). TIPHONE recommends that the delay time should not be longer than 150ms for various application, with a limit of 450ms for voice communication which is still receivable as shown in Table 2.3

Table 2.3: Delay standardization based on TIPHONE(TIPHONE, 1999)

Delay (ms)	Quality
<150	Very good
<250	Good
<350	Enough
<450	Poor

ii. Jitter

Jitter is the variation in delay. It occur due to time difference or interval between the arrivals of packet at the clients end. For example, the transmitter transmit packets in a continuous stream and space them evenly apart. Because of network congestion, configuration errors, improper queuing, the delay between packets can vary instead of remaining constant, therefore, the late arrival frame complicate the reconstruction of the video received. In other to resolve the jitter, the data packet is first collected in the buffer during a predetermine time until the packet is received at the receiver side in correct order. Jitter can result to data loss especially in high transmission speed (Azhar *et al.*, 2016). Table 2.4 show the jitter standardization based on TIPHONE.

Table 2.4Jitter standardization based on TIPHONE(TIPHONE, 1999)

Jitter (ms)	Quality
0	Very Good
<75	Good
<125	Enough
<255	Poor

iii. Packet loss

Packet loss occur when one or more packets of data traveling across a computer network fail to reach their destination. The major causes of packet loss is the queuing exceeding the capacity of buffers at each node(Kumar & Kumar, 2016). In streaming media, packet losses can affect user experience. Table 2.5 show the packet loss standardization based on TIPHONE.

Table 2.5: Packet loss standardization based on TIPHONE(TIPHONE, 1999)

Packet loss (%)	Quality
0	Very Good
3	Good
15	Enough
25	Poor

iv. Throughput

This is the speed or rate of the effective data transfer, it is measured in bits per second(Vinod & Babu, 2014). Throughput can sometimes be associated with bandwidth, because it reflect the actual bandwidth conditions. Throughput is the total number of successful packet arrival divided by the duration of the time interval (Azhar *et al.*, 2016). When transmission speed is faster than the available bandwidth there will be congestion on the network affecting the video quality that should be achieved.

2.3 Review of similar work

In this section, someworks that are relevant to this research are reviewed.

Chang *et al.* (2007)proposed the optimal cache algorithm that caches portion of a video in a relay video proxy close to clients. The proposed method guarantee the video playback quality and also address the problem of insufficient WAN bandwidth across the internet. Nevertheless, in streaming across the internet, data packet are mostly lost which degrades the video quality as

such affect users experience. In order to address the problem associated with the optimal cache algorithm, they also proposed a Priority Selected Cache algorithm which select maximum high priority video data for cache in a relay video proxy. The new approach minimize packet loss and guarantee video playback quality by reducing the decoding error. However, this method is not appropriate for time sensitive and bandwidth sensitive application due to the use of proxy server.

Choi *et al.* (2010)proposed an efficient VoD streaming algorithm for broadband access network such as passive optimal network (PON). They suggested three important elements for efficient VoD streaming in PON which are, efficient streaming schemes for bandwidth savings, an optimal use of deployed network bandwidth and proactive use of user storage facilities. To achieve these goals, they proposed three efficient algorithm, which are, video adaptive streaming (VAST), video greedy adaptive streaming (VGAST) and video greedy adaptive streaming with proactive buffering (VGAST-PB). These algorithm used a method that adaptively increase the streaming speed when the request is high by considering the video popularity, patching scheme and VoD request model to maximize the use of available network bandwidth. The VGAST-PB which is the most efficient among the three algorithms exploits the user storage facilities proactively in addition to the adaptive streaming strategy. The efficiency of the algorithm was shown by analyzing the user waiting time in broadband access networks. From the results obtained, it is evident that when the available network bandwidth is reduced below the required level due to background traffic, the algorithm can considerably reduce the average user waiting time and the number of waiting request which implies that the scheme can also reduce the network bandwidth for the same scheme. However, this work did not focus on real time streaming.

Shen *et al.* (2013)implemented a DHT-aided chunk driven overlay (DCO) for scalable and efficient peer-to-peer live streaming. It is well known that the tree based peer to peer are vulnerable to churn while the mesh based system are churn resilient but suffers from high delay and overhead. In order to address the aforementioned problem, a DCO was proposed. DCO comprises of a video provider selection algorithm, a chunk sharing algorithm and a hierarchical DHT- based infrastructure. The video provider selection algorithm aids in proper utilization of the system bandwidth, the chunk sharing algorithm provides service for chunk index collection and discovery that guarantee high availability and the hierarchical DHT based infrastructure offers high scalability. The experimental results shows that the DCO outperforms the tree based and mesh based systems in terms of latency, availability, scalability and memory overhead. However, there is high cost in implementing the DHT infrastructure.

Zhang *et al.* (2015)proposed a novel approach for improving the QoS in live streaming known as guarantee mechanism of contingency resource (GMCR), by deploying a contingency server to provide contingency service for urgent chunks in order to make sure they can arrive at user's buffer in time. In implementing the GMCR, the chunk scheduling mechanism was adjusted. The chunks that are yet to be delivered were partitioned into urgent and non-urgent chunk. The urgent chunk are those that have reached the given playback deadline while the non-urgent chunk are those that have not yet reached the deadline. The peers request for the non-urgent chunks based on P2P paradigm. The contingency server provides service for those chunks (non-urgent chunks) that have exceeded the playback deadline to become urgent chunks. They also establish a queuing model to analyze the quantitative relation between the amount of contingency server resources and the level of user's QoS in a peer to peer live streaming. However, the approach is not suitable for time and bandwidth sensitive application due to the use of contingency server.

Parodkar and Bade (2015)presented an efficient application layer handover scheme to support seamless mobility for P2P live streaming by employing multichannel communication. The signal strength of access points (APs) that is, RSSI was adopted in their research to predict the handover. When the difference of signal strength between the current and the target AP is less than the given threshold, the mobile peer predict the handover. Once the handover is predicted, neighbor peers can transmit data to the mobile peer at a faster speed by switching their transmission mode. The data unit for data delivery and display is a video block. Each video is divided into small blocks, which are distributed to other peers through the mesh structure. Each peer display video after buffering and sequencing received blocks in memory. The peers exchange distributed video with each other on virtual overlay network. The experimental result shows that P2P live streaming system can considerably improve the playback continuity significantly as the number of participating peers increases. Nevertheless, at a low bandwidth it can't provide QoS for time and bandwidth sensitive application due to the handover mechanism.

Kumar and Kumar (2016)implemented an index base streaming algorithm for enhancing the quality of service in live video streaming for low-motion and ensure optimized bandwidth utilization. The algorithm is implemented in two modules. The first module is located at the server machine in which the camera is connected. As the camera captures the first frame, it sends it to the first module at the server end, the server sends it as a reference frame to the clients which request the live video. The server also receives the second frame from camera and identifies all the pixel which differ in values from the reference frame and compress those values along with the indexing information required to reconstruct the frame on the client side; then it sends them to the clients. The second frame which is taken as reference repeats itself in the process till the live stream requirement is completed. The reconstruction algorithm is located in

the client, it receives three kinds of information, the first one is the first frame which is the reference for the successive frame to be built, the second one is the difference buffer and the last one is the index buffer; in their compressed form. The reconstruction algorithm decompress all the information and builds the frame; then it display the video to the user. The main idea is that instead of sending entire frame, only pixels which differ from the subsequent one will be sent and it will be reconstructed at the client ends. Despite the use of the indexing based streaming algorithm, the difference buffer is still observed to have redundancy of pixels which can further be reduced by exploiting both the spatial and temporal redundancy simultaneously so as to achieve a higher compression efficiency.

Duanmu *et al.* (2017)examined the behavior of humans to the overall effect of video compression, stalling and initial buffering by building a streaming video database and making subjective approach of user study. A new approach of QoE prediction was also proposed that reduces the instantaneous quality degradation which are caused by perceptual video presentation impairments, the playback stalling events and the instantaneous interaction between them. Although the research work emphasis more on the subjective approach than the objective approach. However, the results of the work is at near perfect alignment with the subjective operations and offers superior performance over the known QoE model.

Al-Madani *et al.* (2017)proposed a Data Distribution Service (DSS) middleware for live streaming of full motion video. DDS improves video streaming quality through its efficient and high performance data delivery mechanism. DDS is based on publish/subscribe pattern unlike the conventional client/server model. The middleware layer mitigate network congestion by setting a deadline QoS policy. The deadline QoS policy detect and control congestion. For instance, if the server subscriber waiting time for the net packet exceeds a certain predefined

deadline, it sends a notification to the subscriber who will start minimizing the codec rate to avoid congestion on the subsequent streams. It also used the time based filtering policy to reduce application load at the subscriber side. The experimental result shows that the DDS is a promising technology for distributing video over networks, since it consumes low bandwidth, has low jitter and cause lesser packet loss. However, there is high cost in implementing the design due to the use of DSS.

Canel *et al.* (2018)proposed an algorithm for detectinginteresting frames in video by identifying the relevant ones. The relevant frames was decided by the algorithm by extracting the semantic content using the Deep Neural Network (DNN). The algorithm uses a hierarchy of filters to trade-off between end-to-end latency and aggressive decimation. The semantic diversity of the selected frame was also maximize by the algorithm, thereby handles bursty event in streaming videos. The advantage of the algorithm over others is that it select exactly the desired number of relevant frames, creating a uniform output frame rate from a non-uniform streaming of interesting frames. This is in contrast to a more dynamic algorithm that adjust some selected frames based on the content of the video. Although, the algorithm succeeds in its requirement of achieving exactly the desired reduction factor. However, the characteristic of the result depends on the size of the buffer. Using a very large buffer results in delay and memory overhead due to the polynomial complexity of the longest path algorithm.

In view of these limitations identified from literatures, it is evident that streaming on a low bandwidth result to poor video quality due to heavy traffic, delay in delivery and packet loss. To achieve a smooth streaming through low bandwidth, there is need to exploit both spatial and temporal redundancy that exit in video frames. This will significantly reduce the heavy traffic

and as a consequence eliminate packet loss and delay in delivery that adversely affect user's experience.

CHAPTER THREE

MATERIALS AND METHOD

3.1 Introduction

In this chapter, the materials, the methods and reported procedure used for the implementation and simulation of the spatio-temporal frame indexing algorithm for improving the quality of service in live low motion video are described based on the outline developed in section 1.5. The spatio-temporal frame indexing algorithm for improving the quality of service in live low-motion videos streaming was implemented using MATLAB.

3.2 Materials

In this section, the materials used for the implementation of the research presented in this report are discussed. These materials involve the specification of the computer system used and the software used for the implementation.

3.2.1 Computer System

Simulations performed in this research were carried out using HP Pro desktop computer situated at the ICT laboratory of the NLNG multi-user laboratory, Ahmadu Bello University Zaria. The specifications are discussed.

i. HP desktop computer with the following features:

Processor: Intel (R) Core (TM) i5-3470 CPU @ 3.20GHz

Installed memory (RAM): 8.00GB (7.88 GB usable)

System type: 64-bit Operating System, x64-based processor

3.2.2 MATLAB

The MATrix LABoratory (MATLAB) is the software which is used to implement all the models developed by MathWorks. For the purpose of this research, MATLAB 2018a version was used.

3.3 Implementation of the Standard Frame Indexing Algorithm

The process involved in the implementation of the standard temporal frame indexing algorithm are discussed in details in the following subsection.

3.3.1 Indexing algorithm

The indexing algorithm is located at the server. The algorithm send the first frame to the client as reference frame, the server receives the second frame and identifies all pixels that differs from the reference frame and compress those values along with the indexing information that is required for the reconstruction of a new frame on the client. The new reconstructed frame (second frame) would serve as reference to the third frame. This process continues until streaming stops. The indexing algorithm is given in subsection 3.2.1.1

3.3.1.1 Temporal indexing algorithm

The following steps are necessary in order to perform the indexing

Step 1: Establishment of connection between client and server.

Step 2: Initialize video camera.

Step 3: Send first frame.

Step 4: Client received first frame in compressed form.

Step 5: Set the current frame as reference frame.

Step 6: Receive new frame and do temporal redundancy check.

Step 7: Store the difference pixels in buffer (difference buffer) and create the index buffer based on the specified indexing value.

Step 8: Send the result of step 6 and step 7 to client in a compressed form.

Step 9: The process continues from step 5 till streaming end.

The first frame captured is shown in Figure 3.1

156	159	158	154	151	152	156	154	152	156
156	162	158	159	161	149	149	141	153	154
156	157	158	157	155	154	159	157	155	154
152	152	152	153	154	151	156	151	153	154
152	151	150	153	157	151	159	158	159	154
155	154	152	150	149	149	149	149	151	154
156	159	158	159	157	154	146	148	147	150

Figure 3.1 First frame captured

Figure 3.1 Illustrate the first frame captured. The frame was compressed and sent to the client as a reference frame for the second frame that will be captured. Figure3.2 shows the new (second) frame captured.

156	159	158	154	**155**	**155**	**155**	**155**	152	156
156	162	**157**	**157**	**157**	**157**	**158**	**159**	**158**	**156**

155	**158**	155	157	155	154	**159**	**153**	**152**	**156**
156	**156**	**156**	**156**	**158**	**159**	**154**	**152**	**150**	**150**
150	151	150	153	157	151	159	158	159	154
155	154	152	150	149	149	149	149	151	154
156	159	158	159	157	154	146	148	147	150

Figure 3.2: New frame captured

The second frame is shown in Figure 3.2. The shaded portion represent the part that changed when compared with the first frame. These part was stored in the difference buffer. Figure 3.3 shows the difference buffer of the second frame.

155	155	155	155	157	157	157
157	158	159	158	156	155	158
153	152	156	156	156	156	156
158	159	154	152	150	150	150

Figure 3.3: Difference buffer of the new frame

Figure 3.3 shows the difference buffer of the second frame. The difference buffer was compressed along with the indexing buffer. The indexing buffer was built with the aid of the specified indexing value. The indexing value that is needed to build the indexing buffer is illustrated in Table 3.1.

Table 3.1: Values to build index buffer

Indexing value	Translation
-1	Two successive equal frame
-2	Copy pixel value from difference buffer
-3	Copy pixel value from reference frame

The indexing value is illustrated in Table 3.1. The index buffer was generated using the indexing value. Figure 3.4illustrate the indexing buffer.

-3 4	-2 4	-3 4	-2 10	-3 5	-2 14	-3 29

Figure 3.4: Index buffer of the new frame

Figure 3.4 shows the indexing buffer. The indexing buffer was compressed along with the difference buffer and was sent to the client, the client then reconstruct and display.

3.3.2 The reconstruction algorithm

The reconstruction algorithm is located in the client, it receives three kinds of information, the first frame which is the reference for the next frame to be built, second is the difference buffer and third is the index buffer all in compressed form. The reconstruction algorithm decompress all the information and built the frame; then it display the video. The reconstruction algorithm is shown in subsection 3.3.2.1

3.3.2.1 Temporal reconstruction algorithm

The following steps are taken in order for the client to reconstruct and display a frame.

Step 1: Client request for video.

Step 2: Receives first (reference) frame in compressed form.

Step 3: Receives the difference and index buffer in compressed form.

Step 4: Decompress the reference frame and buffers (difference and index).

Step 5: Reconstruct new frame using the two buffers.

Step 6: Display the new frame and set as reference.

Step 7: Repeat same process from step 3 until streaming end.

The reconstruction process are as follows.

Form Figure 3.4, -3 4 signifies that the first four pixels are same when compared with the reference frame, this means that the first four pixels can be copied from the reference frame. Figure 3.5 shows the reconstruction of the frame after the first indexing value.

156	159	158	154						

Figure 3.5: Reconstruction of the frame after first index value

From Figure 3.5, the second index value is -2 4 which implies that the next four pixels value are differed from the reference andcan be copied from difference buffer. Figure 3.6 shows the construction of the frame after second index value.

156	159	158	154	155	155	155	155		

Figure 3.6: Construction after second index value

The third index value from Figure 3.4 is -3 4 which signifies that the next four pixels are same as reference, these means that the next four can be copied from the reference frame. The construction of the frame after third index value is illustrated in Figure 3.7

156	159	158	154	155	155	155	155	152	156
`156	162								

Figure 3.7: Construction of the frame after third index value

The fourth index value from Figure 3.4, which is -2 10, it signifies that the next ten pixels value are differed from the reference frame and can be copied from the difference buffer. Figure 3.8 shows the construction of the frame after fourth index value.

156	159	158	154	155	155	155	155	152	156
156	162	157	157	157	157	158	159	158	156
155	158								

Figure 3.8 Construction of the frame after fourth index value

After the fourth index value, the next value in the index buffer in Figure 3.4, which is -3 5, means that the next five pixelsvalues are same as reference and can be copied from the reference frame. The illustration in Figure3.9 shows the construction after the fifth index value.

156	159	158	154	155	155	155	155	152	156
156	162	157	157	157	157	158	159	158	156
155	158	155	157	155	154	159			

Figure 3.9: Construction of the frame after fifth index value

From Figure 3.4, it can be seen that the sixth index value is -2 14, which implies that the next fourteen pixels value are differed from the reference and can be copied from the difference buffer. Figure 3.10 shows the construction after sixth index value.

156	159	158	154	155	155	155	155	152	156
156	162	157	157	157	157	158	159	158	156
155	158	155	157	155	154	158	153	152	156
156	156	156	156	158	159	154	152	150	150
150									

Figure 3.10: Construction of the frame after sixth index value

It can be seen from Figure 3.4 that the last index value which is -3 29, signifies that the next 29 pixels value are same as reference and can be copied from the reference frame.Figure 3.11 shows the construction of the frame after the seventh index value.

156	159	158	154	155	155	155	155	152	156
156	162	157	157	157	157	158	159	158	156
155	158	155	157	155	154	158	153	152	156
156	156	156	156	158	159	154	152	150	150
150	151	150	153	157	151	159	158	159	154
155	154	152	150	149	149	149	149	151	154
156	159	158	159	157	154	146	148	147	150

Figure 3.11: Construction of the new frame after the seventh index pixel

Similarly, the reconstruction of other frames follows the same process and display the video to the user, however, the difference buffer is observed to be large due to redundancy of pixels which can be further reduced by considering both the spatial and temporal redundancy simultaneously.

3.4 Development of the Spatio-Temporal Frame Indexing Algorithm

In implementing the spatio-temporal frame indexing algorithm both the temporal and spatial redundancy are considered in order to reduce the difference buffer size, increase the compression ratio and provides smooth streaming even at a low bandwidth. Subsection 3.3.1 shows the indexing algorithm for the implementation.

3.4.1 Spatio-temporal indexing algorithm

In order to perform the indexing using spatio-temporal frame indexing algorithm, the following steps are necessary

Step 1: Establishment of connection between client and server.

Step 2: Initialize video camera.

Step 3: Send first frame.

Step 4: Client received first frame in compressed form.

Step 5: Set the current frame as reference frame for the next frame.

Step 6: Receive new frame and exploit both the spatial and temporal redundancy.

Step 7: Store the difference pixels in buffer (difference buffer) and create the index buffer based on the specified indexing value.

Step 8: Send the result of step 6 and step 7 to client in a compressed form.

Step 9: The process continues from step 5 till streaming end.

The indexing value and its translation are given in Table 3.2

Table 3.2: the indexing value and its translation

Indexing value	Translation
-1	Two successive equal frame
-2	Copy pixel values from difference buffer
-3	Copy pixel values from reference frame
-5	Copy pixel value from difference buffer and replicate for x numbers of time

Using the first and second frame captured shown in Figure 3.1 and Figure 3.2 respectively.

The difference buffer and index buffer are illustrated in Figure 3.12 and Figure 3.13 respectively

155	157	158	159
158	156	155	158
153	152	156	158
159	154	152	150

Figure 3.12: Difference buffer using the spatio-temporal techniques

-3 4	-5 4	-3 4	-5 4	-2 6	-3 5	-2 2	-5 5	-2 4	-2 3	-3 29

Figure 3.13: The index buffer using the spatio-temporal frame indexing techniques

The reconstruction process is as follows: From the index buffer in Table 3.13, the first index value which is -3 4, means that the new frame consist of first four pixels which are the same when compared with the reference frame, this means that the first four pixels can be copied from the reference frame. Figure 3.14 shows the construction of the frame after the first index value.

156	159	158	154						

Figure 3.14: Construction of the frame after first index value

From Figure 3.13, the second index value -5 4, signifies that the next four pixels values differs from the reference but spatially related. This means that the next pixel value can be copied from the difference buffer and duplicate four times. After the second indexing value, the construction of the frame is shown in Figure 3.15

156	159	158	154	155	155	155	155		

Figure 3.15: The construction of the frame after second indexing value

The third index value in Figure 3.13, which is -3 4 implies that the next the next four are same as the reference, this means that the next four pixels can be copied fromthe reference frame. Figure 3.16 shows the construction after the third indexing values.

156	159	158	154	155	155	155	155	152	156
156	162			-					

Figure 3.16: The construction of the frame after the third indexing value

The fourth index value in Figure 3.13, which is –5 4 implies that after the construction of the third index value the next four differs from the reference but spatially related. This means that

the next pixel value can be copied from the difference and duplicate four times. After the fourth indexing value, the construction of the frame is shown in Figure 3.17

156	159	158	154	155	155	155	155	152	156
156	162	157	157	157	157				

Figure 3.17: The construction of the frame after the fourth indexing value

From Figure 3.13, the-2 6 means that the next six pixels value after the construction of the fourth index value differs from the reference frame and as such can be copied from the difference buffer. Figure 3.18 shows the construction after the fifth index values.

156	159	158	154	155	155	155	155	152	156
156	162	157	157	157	157	158	159	158	156
155	158								

Figure 3.18: The construction of the frame after the fifth indexing value

After the fifth indexing value from Figure 3.13, the next which is -3 5 denotes that the next five pixels are same as the reference and can be copied from the reference frame. Figure 3.19 shows the construction after the sixth indexing value.

156	159	158	154	155	155	155	155	152	156
156	162	157	157	157	157	158	159	158	156
155	158	155	157	155	154				

Figure 3.19: The construction of the frame after the sixth indexing value

After the sixth indexing value from Figure 3.13, the next two pixels differs from the reference frame which result to -2 2; this implies that the next two pixels can be copied from the difference buffer. Figure 3.20 shows the construction after the seventh index value.

156	159	158	154	155	155	155	155	152	156
156	162	157	157	157	157	158	159	158	156
155	158	155	157	155	154	154	153	152	

Figure 3.20: The construction of the frame after the seventh indexing value

From Figure 3.13, after the seventh indexing value, the next which is -5 5 signifies that after the first 29th pixels value, the next five pixels differs from the reference but spatially related. This means that the next pixel value can be copied from the difference buffer and duplicate five times. The construction of the frame after the eight index value is shown in Figure 3.21

156	159	158	154	155	155	155	155	152	156
156	162	157	157	157	157	158	159	158	156

155	158	155	157	155	154	154	153	152	156
156	156	156	156						

Figure 3.21: The construction of the frame after the eight indexing value

After the eighth indexing value from Figure 3.13, the next four pixels differs from the reference frame which results to -2 4; this signifies that the next four pixels value can be copied from the difference buffer. Figure 3.22 shows the construction after the ninth index values.

156	159	158	154	155	155	155	155	152	156
156	162	157	157	157	157	158	159	158	156
155	158	155	157	155	154	154	153	152	156
156	156	156	156	158	159	154	152		

Figure 3.22: The construction of the frame after the ninth indexing value

From Figure 3.13, the-2 3 means that the next three pixels value differs from the reference frame but spatially related. This means that the next pixel value can be copied from the difference buffer and duplicate three times. The construction of the frame after the tenth index value is shown in Figure 3.23

156	159	158	154	155	155	155	155	152	156

156	162	157	157	157	157	158	159	158	156
155	158	155	157	155	154	154	153	152	156
156	156	156	156	158	159	154	152	150	150
150									

Figure 3.23: The construction after the tenth indexing value

The last indexing value from Figure 3.13 which is -3 29, signifies that after the first 41st pixels value, the next twenty nine are same as the reference, this means that the next twenty nine can be copied from the reference frame. Figure 3.24 shows the construction after the eleventh pixels value.

156	159	158	154	155	155	155	155	152	156
156	162	157	157	157	157	158	159	158	156
155	158	155	157	155	154	158	153	152	156
156	156	156	156	158	159	154	152	150	150
150	151	150	153	157	151	159	158	159	154
155	154	152	150	149	149	149	149	151	154
156	159	158	159	157	154	146	148	147	150

Figure 3.24: The construction after the eleventh indexing value

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, the results and discussion for the research work are presented. The performance of the standard frame indexing and spatio-temporal frame indexing are evaluated using the performance metrics as discussed in subsection 4.3

4.2 Simulation result

Figure 4.1 depict the results of the 1st and 100thframe captured respectively when tested on a standard low-motion video and Figure 4.2 shows the result when applied on a local video. The result is compared with (Kumar& Kumar, 2016)using the difference buffer size, compression ratio and time to build up frame.

(a)1st captured(b) 100th frame

Figure 4.1: Result of 1st and 100thframe whenapplied on a standard low-motion video

(a) 1stframe captured (b) 100thframe captured

Figure 4.2 Result of 1st and 100thframe when applied on a local video

The first and hundredth frame of a standard and local videos are shown in Figure 4.1 and Figure 4.2 respectively. The server sent the first frame as a reference frame to the client, successive frames are reconstructed based on the reference frame, difference buffer and indexing information.

4.3 Performance of the algorithm on a standard low-motion video

In other to evaluate the performance of the spatio-temporal frame indexing algorithm, the difference buffer size, compression ratio and time needed to build up frame are used as metrics.

4.3.1 Comparison using difference buffer size

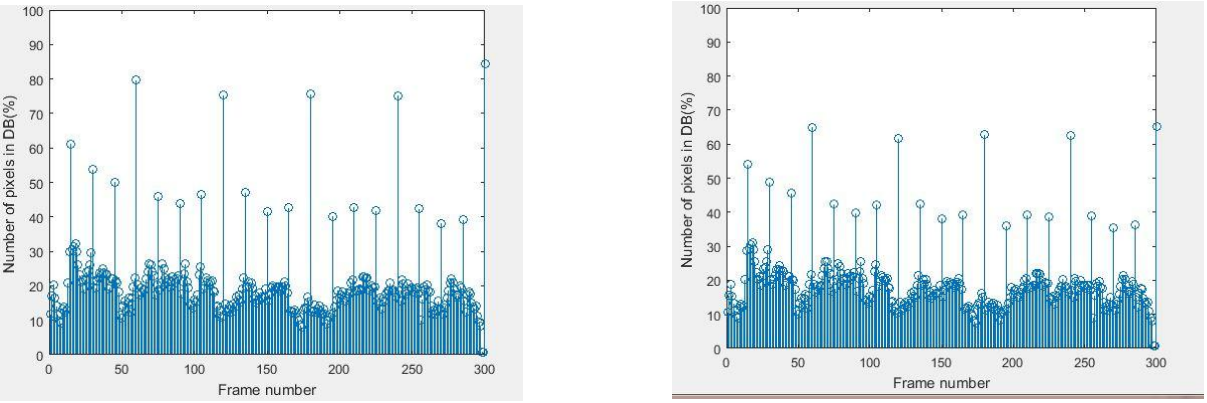
With frame size of 264 X 352 X 3, Table 4.1 shows the comparison of the buffer size of some randomly selected frames in the standard low-motion video depicted in Figure 4.1

Table 4.1: Comparison of the buffer size of some randomly selected frames in the video in Figure 4.1

Number of frames	Standard indexing (%)	Spatio-temporal indexing (%)
3	17.1746	15.4812
8	9.8969	9.4966
30	29.7327	29.0128
50	11.4569	10.9712
65	17.9253	17.1972
72	24.1610	23.4483
86	22.6506	22.1089
100	13.6755	13.1614
250	19.4111	18.6532
280	18.9523	18.6245

Table 4.1 presents the comparison of the buffer size of some randomly selected frames in the standard low-motionvideo depicted in Figure 4.1.It can be seen that streaming using spatio-temporal frame indexing techniques requires a lesser size of difference buffer when compared

with the standard frame indexing techniques. Figure 4.3 shows the plots of the difference buffer sizes of both the standard and spatio-temporal frame indexing.



(a) Standard indexing_ Difference buffer (b) Spatio-temporal_ Difference buffer

Figure 4.3Difference buffer of standard and spatio-temporal frame indexing.

From Figure 4.3 thepercentage of pixels that is required for the reconstruction of the low-motion video depicted in Figure 4.1 for standard and spatio-temporal frame indexing are 20.08% and 19.05% respectively. It can be seen that streaming using spatio-temporal frame indexing required a lesser size of difference buffer when compared with the standard frame indexingmethod and also instead of exploiting only the temporal redundancy, the spatial and temporal redundancy are exploited simultaneously for the optimal bandwidth utilization during streaming.

4.3.2 Comparison using the time to build frame

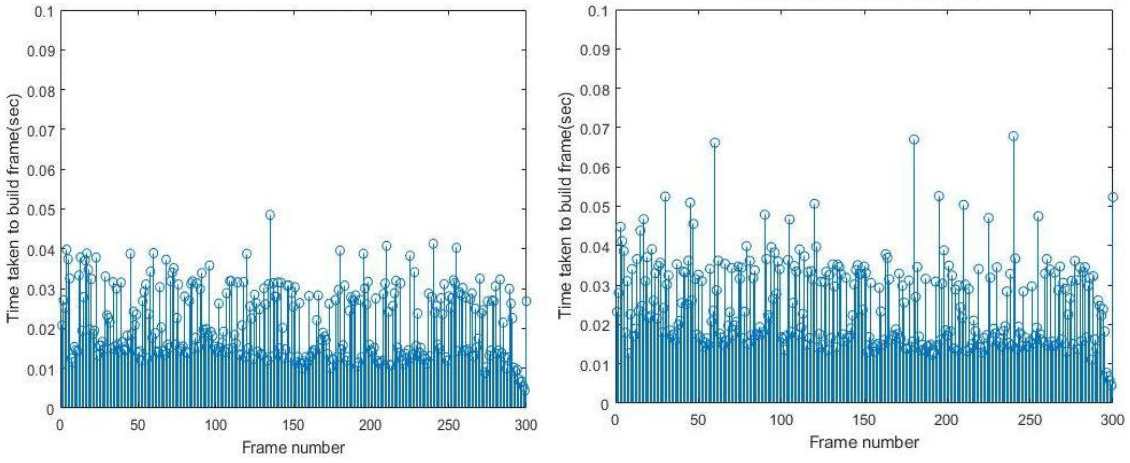
The time to build of some randomly selected frames in the video depicted in Figure 4.1 are shown in Table 4.2

Table 4.2: Comparison of the Time to buildof the standard and the spatio-temporal frame indexing

Number of frames	Standard indexing (sec)	Spatio-temporal indexing

		(sec)
3	0.0281	0.0312
8	0.0152	0.0161
30	0.0365	0.0219
50	0.0188	0.0197
65	0.0170	0.0229
72	0.0180	0.0194
86	0.0184	0.0197
100	0.0351	0.0204
250	0.0319	0.0325
280	0.0314	0.0339

From Table 4.2 it can be seen that there is a slight increase in the time to build up frames using the developed spatio-temporal frame indexing methodwhen compared with the standard frame indexing. Figure 4.4 shows the plots of the time to build of the standard and spatio-temporal frame indexing techniques.



(a) Time to build frame_ Standard indexing (b) Time to build_ Spatio Temporal

Figure 4.4: Time to build frame of standard and spatio temporal indexing.

From Figure 4.4 the average time to build are 0.0241seconds and 0.0262 seconds respectively. The slight increase in the time to build of the developed spatio-temporal frame indexing is due an increase in the indexing information. However, setting a fixed window value can minimize the time to build up a frame.

4.3.3 Comparison using compression ratio

Table 4.3 shows the compression ratio of some randomly selected video frames as depicted in Figure 4.1.

Table 4.3: Compression ratio of some randomly selected video frame in Figure 4.1

Number of frames	Standard indexing	Spatio-temporal indexing
3	0.17	0.15
8	0.99	0.95
30	0.30	0.29
50	0.12	0.11
65	0.18	0.17
72	0.24	0.23
86	0.23	0.22
100	0.14	0.13
250	0.19	0.19
280	0.19	0.19

From Table 4.3 it can be seen that the spatio-temporal frame indexing achieves a better compression when compared with the standard frame indexing using the standard low-motion video.

4.4 Performance of the algorithm on a local video

The performance of the algorithm was also verified on a local video and the results were discussed in the subsection 4.4

4.4.1 Comparison using difference buffer size

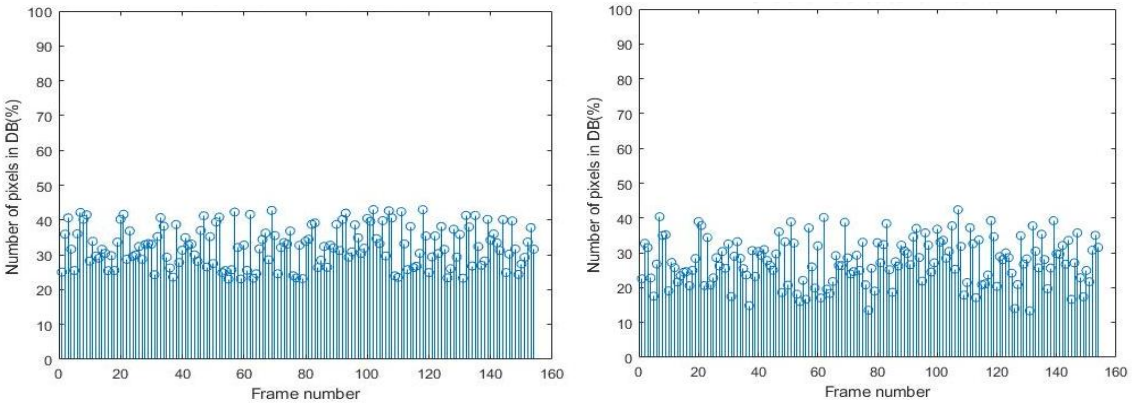
With an initial frame size of 373 X 597 X 3, Table 4.4 shows the comparison of the buffer size of some randomly selected frames in the local video depicted in Figure 4.2

Table 4.4: Comparison of the buffer size of some randomly selected frames in the video in Figure 4.2

Number of frames	Standard indexing (%)	Spatio-temporal indexing (%)
4	31.8470	27.7311
11	36.6630	35.8371
14	24.2988	20.7534
20	28.1396	25.9823
33	39.9416	32.6465
40	40.6100	38.8384
54	27.1202	24.0001
60	27.6670	26.6515
100	41.2634	31.7217

123	23.2913	15.2237

It is seen that streaming using spatio-temporal frame indexing techniques requires a lesser size of difference buffer when compared with the standard frame indexing techniques on a local video. Figure 4.5 shows the plots of the difference buffer sizes of both the standard and spatio-temporal frame indexing.



(a) Standard indexing_ Difference buffer (b) Spatio-temporal_ Difference buffer

Figure 4.5 Difference buffer of standard and spatio-temporal frame indexing.

From the plots in Figure 4.5 the percentage of pixels that was usedfor the reconstruction of the of the low-motion video depicted in Figure 4.2 are 32.80% and 27.61%. It is seen that streaming using the spatio-temporal frame indexing techniques requires a lesser buffer size when compared with the standard frame indexing techniques.

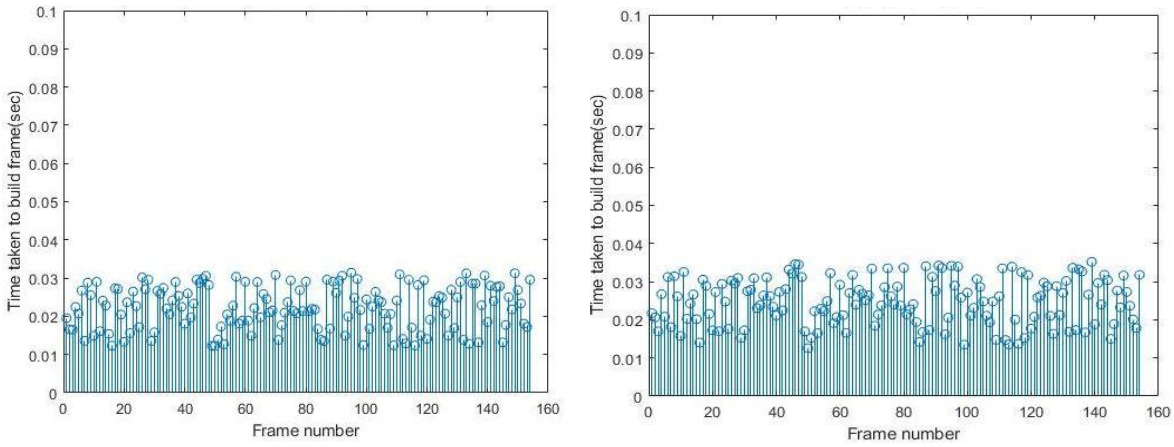
4.4.2 Comparison using the time to build frame

Table 4.5 presents the comparison of the time to build of some randomly selected frames in the video depicted in Figure 4.2 using the standard and the spatio-temporal frame indexing

Table 4.5:Comparison of the Time to build of the standard and the spatio-temporal frame indexing

Number of frames	Standard indexing (sec)	Spatio-temporal indexing (sec)
4	0.0235	0.0237
11	0.0245	0.0259
14	0.0290	0.0318
20	0.0125	0.0131
33	0.0160	0.0166
40	0.0177	0.0215
54	0.0204	0.0214
60	0.0155	0.0185
100	0.0178	0.0185
123	0.0181	0.0190

It can be seen from Table 4.5 that there is a slight increase in the time to build up frame using the developed spatio-temporal frame indexing method when compared with the standard frame indexing. The slight increase in the time to build up frames is due to an increase in the indexing information. Figure 4.4 shows the plots of the time to build of the standard and spatio-temporal frame indexing techniques.



(a) Time to build frame_ Standard indexing (b) Time to build_ Spatio Temporal

Figure 4.6: Time to build frames of standard and spatio temporal indexing.

The average time taken to build up frames in Figure 4.6 are 0.0213 seconds and 0.0236 seconds respectively. The slight increase in the time to build is due to an increase in the index information.

4.4.3 Comparison using compression ratio

The compression ratio of some randomly selected frames in Figure 4.2 are shown in Table 4.6

Table 4.6: Compression ratio of some randomly selected video frame in Figure 4.2

Number of frames	Standard indexing (%)	Spatio-temporal indexing (%)
4	0.32	0.28
11	0.37	0.36
14	0.24	0.21
20	0.28	0.26
33	0.40	0.30
40	0.41	0.39
54	0.27	0.24
60	0.28	0.27
100	0.41	0.32

123	0.23	0.15

From Table 4.6 It is seen that the developed scheme achieves a good compression when compared with the existing scheme on a locallow-motion video.

4.4.4 Results of comparison between the standard and spatio-temporal frame indexing using standard and local low-motion video

This subsection presents the results obtained using the standard and spatio-temporal frame indexing techniques and the comparison between the two techniques. The summary of the results is presented in Table 4.7

Table 4.7 shows the summary of the comparison between the result obtained using standard and local low motion video

Performance metrics	Initial frame	Standard low-motion video			Initial frame	Local low-motion video		
		Standard frame indexing	Spatio-temporal frame indexing	Percentage improvement (%)		Standard frame indexing	Spatio-temporal frame indexing	Percentage improvement (%)

Buffer size (Ave. val)	278784	55980 (20.08%)	53108 (19.05%)	5.13	668043	219118 (32.80%)	184447 (27.61%)	15.8
Frame built time (Ave. val)		0.0213 (sec)	0.0236 (sec)	-10.8		0.0241 (sec)	0.0262 (sec)	-8.71
Compression ratio (Ave.val)		0.20	0.19	5%		0.32	0.27	15.6

Table 4.7 presents the summary of the results obtained in streaming using the standard and local low-motion video. It can be concluded the developed scheme outperform the standard indexing in the difference buffer size and compression ratio. However, there is a slight increase in the time to build up frames which results due to high computation.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

In this dissertation, the development of an index based streaming algorithm called the spatio-temporal frame indexing algorithm has been presented. The developed algorithm exploits both the spatial and temporal redundancy that occur in video streaming (low-motion videos), hence, accounting for optimal bandwidth utilization. Thus, factors such as delay in delivery, resolution degradation and packet loss that adversely affect the QoS of live streaming can reduced to the barest minimum even when streaming on a low bandwidth.

5.2 Conclusion

The research presents the development of a spatio-temporal frame indexing algorithm for improving the quality of service in live low-motion video streaming. The spatio-temporal frame indexing was developed by exploiting both the spatial and temporal correlations in a frame and successive frames as against the conventional streaming techniques which transmit an entire frames or only exploit temporal correlations. Only the dissimilar pixels are transmitted along with the buffers information. Reconstruction occur at the client end using the difference pixels and the buffers information. This aid in optimal utilization of bandwidth of video streaming network and a reduction in heavy traffic that is being sent during video streaming.The performance were compared with the standard frame indexing using the difference buffer size, time to build and compression ratio. The developed scheme outperform the standard indexing in the difference buffersize and compression ratio. However, this is at the expense of slight increase in the time to build.

5.3 Significant Contributions

The significant contribution to this research is as follows:

1. A spatio-temporal frame indexing algorithm for improving the QoS of live video streaming was developed.
2. The developed algorithmexploit both the spatial and temporal redundancy that occur in a frame and successive frames and as such only the dissimilar pixels are identified and transmitted over the network for optimal bandwidth utilization.
3. The developed algorithm achieves a 5.4% and 15.8% improvement on the size of difference buffer when tested on a standard and local low-motion video respectively, also,it is more compressible when compared with the existing scheme.

5.4 Recommendations for further work

The following possible areas of further work are recommended for consideration for further research:

1. The length of the index buffer can be reduced by setting a fixedwindow size. This can also reduce the time taken to build up frames.
2. The research work only considered a low-motion videos, high motion videos can be considered in future research.

REFERENCES

Al-Madani, B., Al-Roubaiey, A., & Baig, Z. A. (2017). Real-time QoS-aware video streaming: a comparative and experimental study. *International Journal of Electronics and Communications (AEU), 73*, 34-45. doi: *http://dx.doi.org/10.1155/2014/164940*

Apostolopoulos, J. G., Tan, W.T., & Susie, W. J. (2002). *Video Streaming: Concepts, Algorithms, and Systems*. Palo Alto, CA, USA: Hewlett-Packard Laboratories.

Austerberry, D. (2005). *The Technology of Video and Audio Streaming*. USA: Elsevier.

Azhar, A. Z., Pramono, S., & Supriyanto, E. (2016). An Analysis of Quality of Service (QoS) In Live Video Streaming Using Evolved HSPA Network Media. *JAICT, 1*(1), 1-6.

Canel, C., Kim, T., Zhou, G., Li, C., Lim, H., Andersen, D. G., Dulloor, S. R. (2018). Picking Interesting Frames in Streaming Video. *2018 SysML Conference*1(1), 1-3.

Chang, S.-H., Chang, R.-I., Ho, J.-M., & Oyang, Y.-J. (2007). A priority selected cache algorithm for video relay in streaming applications. *IEEE transactions on broadcasting, 53*(1), 79-91. doi: 10.1109/TBC.2006.887170

Choi, J., Yoo, M., & Mukherjee, B. (2010). Efficient video-on-demand streaming for broadband access networks. *IEEE/OSA journal of optical communications and networking, 2*(1), 38-50

Duanmu, Z., Zeng, K., Ma, K., Rehman, A., & Wang, Z. (2017). A quality-of-experience index for streaming video. *IEEE Journal of Selected Topics in Signal Processing, 11*(1), 154-166. doi: 10.1109/JSTSP.2016.2608329

Gangurde, T. P., & Nikam, S. (2017). A Survey on Live Video Streaming Over Peer to Peer Network. *International Journal of Computer Trends and Technology (IJCTT), 43*(1), 5-7.

Kozamernik, F. (2011). *Media streaming over the internet* (EBU Ed.). UK: European Bradcasting Union.

Kumar, R. J. N., & Kumar, A. P. (2016). Improving QOS in Live Video Streaming for Low-motion Videos Through Frame Indexing Method. *International Journal of Communicaton Technology for Social Networking Services, 4*(1), 9-22. doi: *http://dx.doi.org/10.21742/ijctsns.2016.4.1.02*

Lahbabi, Y., Elhaj, E. H., & Hammouch, A. (2014). Quality adaptation using Scalable Video Coding (SVC) in Peer-to-Peer (P2P) Video-onDemand(VoD) Streaming. *2014 International Conference on Multimedia Computing and system (ICMCS)*. doi: 10.1109/ICMCS.2014.6911236

Liu, Y., Guo, Y., & Liang, C. (2008). A survey on peer-to-peer video streaming systems. *Peer-to-Peer Networking and Applications, 1*(1), 18-28. doi: 10.1007/s12083-007-0006-y

Mahini, H., Dehghan, M., Navidi, H., & Rahmani, A. M. (2017). Peer-assisted video streaming based on network coding and Beer-Quiche game. *AEU-International Journal of ElectronicsandCommunications,73*,34-45.

Mittal, S., & Vetter, J. S. (2016). A survey of architectural approaches for data compression in cache and main memory systems. *IEEE Transactions on Parallel and Distributed Systems, 27*(5), 1524-1536. doi: DOI 10.1109/TPDS.2015.2435788

Parodkar, M. S., & Bade, D. (2015). Improve the performance of p2p live streaming. *International Journal of Computer Science and Mobile Computing, 4*(2), 28-32.

Santos-González, I., Rivero-García, A., Molina-Gil, J., & Caballero-Gil, P. (2017). Implementation and Analysis of Real-Time Streaming Protocols. 1(1), 1-17 doi: 10.3390/s17040846

Sasi, L. A., & Madhavu, M. L. (2014). A Survey on Peer to Peer Video Streaming Systems. *International Journal of Research in Computer Engineering & Electronics, 3*(1), 18-29

Shen, H., Li, Z., & Li, J. (2013). A DHT-aided chunk-driven overlay for scalable and efficient peer-to-peer live streaming. *IEEE Transactions on Parallel and Distributed Systems, 24*(11), 2125-2137. doi: 10.1109/TPDS.2012.302.

Taksande, S., Joshi, K., Chikaraddi, V., & Raksha, S. (2015). Video Streaming Techniques and Issues.*International Journal of Advanced Research in Computer Science & Technology, 3*(1), 141-144.

TIPHONE Recommendation (1999): General aspect of QoS (pp. 1-37).

Vinod, S. H., & Babu, K. R. M. (2014). Improvement of Bandwidth Efficient Video and Live Streaming in Mobile using AWS Cloud. *International Journal of Computer Science and Information Technologies (IJCSIT), 5*(4), 5415-5421.

Zhang, G., Hu, C., Xing, C., Wang, N., & Wei, X. (2015). A novel scheme for improving quality of service of live streaming.*Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*.*39*(4), 457-462. doi: 10.1109/3PGCIC.2014.93

Zou, X. K., Erman, J., Gopalakrishnan, V., Halepovic, E., Jana, R., Jin, X., Sinha, R. K. (2015). *Can accurate predictions improve video streaming in cellular networks?* Paper presented at the Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications.57-62.

APPENDIX A

MATLAB CODE FOR THE STANDARD INDEXING ALGORITHM

```
%% Standard Algorithm
vidName = 'akiyo';
Camera = VideoReader([vidName, '.mp4']);
nFrames = Camera.NumberOfFrames;
%% server
if ~exist(strcat(pwd, '\', vidName, '_S.mat'), 'file')
display('Server Side')
DeviceA = ServerClass;
data = {};
for f = 1:nFrames
display(['Frame-' num2str(f)])
%receive frame from camera
frame = read(Camera,f);
DeviceA = DeviceA.receiveFrame(frame)
%compress and send it to client
data{f} = DeviceA.compressData();
end
save([vidName, '_S.mat'], 'data')
end
%% client
load([vidName, '_S.mat'])
display('Client Side')
DeviceB = ClientClass;

figure
currAxes = axes;

for f=1:nFrames
display(['Frame-' num2str(f)])

%receive compressed data
DeviceB = DeviceB.receiveData(data{f});
```

```
frame = DeviceB.Reference;

%stream video
image(frame, 'Parent', currAxes);
currAxes.Visible = 'off';

pause(1/Camera.FrameRate);
end

figure; stem(DeviceB.TimeToBuild)
title('Time to build standard')
ylim([0 0.3])

figure; stem(DeviceB.SamePixels)
title('Number of same pixels standard')
ylim([0 100])

figure; stem(DeviceB.BufferSize)
title('size of difference buffer standard')
ylim([0 100])

result=[[2:nFrames]' DeviceB.TimeToBuild' DeviceB.SamePixels' DeviceB.BufferSize']
mean(result)
```

APPENDIX B

MATLAB CODE FOR THE SPATIO-TEMPORAL INDEXING ALGORITHM

```
% close all
clear
clc

%% Improved algorithm
vidName = 'akiyo';
Camera = VideoReader([vidName, '.mp4']);

nFrames = Camera.NumberOfFrames;

%% server
if ~exist(strcat(pwd, '\', vidName, '_I.mat'), 'file')
   display('Server Side')
   DeviceA = ImprovedServerClass;
   data = {};
   for f = 1:nFrames
      display(['Frame-' num2str(f)])
      %receive frame from camera
      frame = read(Camera,f);
      DeviceA = DeviceA.receiveFrame(frame);

      %compress and send it to client
      data{f} = DeviceA.compressData();
   end
   save([vidName, '_I.mat'], 'data')
end


%% client
load([vidName, '_I.mat'])
display('Client Side')
DeviceB = ImprovedClientClass;
```

```
figure
currAxes = axes;

for f=1:nFrames
    display(['Frame-' num2str(f)])

    %receive compressed data
    DeviceB = DeviceB.receiveData(data{f});
    frame = DeviceB.Reference;

    %stream video
    image(frame, 'Parent', currAxes);
    currAxes.Visible = 'off';

    pause(1/Camera.FrameRate);
end

figure; stem(DeviceB.TimeToBuild)
title('Time to build _ spatio-temporal')
ylim([0 0.3])

figure; stem(DeviceB.SamePixels)
title('Number of same _ spatio-temporal')
ylim([0 100])

figure; stem(DeviceB.BufferSize)
title('size of difference buffer _ spatio-temporal')
ylim([0 100])

result=[[2:nFrames]' DeviceB.TimeToBuild' DeviceB.SamePixels' DeviceB.BufferSize']
mean(result)
```

APPENDIX C

MATLAB CODE FOR THE SERVER CLASS

```
classdef ServerClass
properties
FrameIndex = 0;
Reference
DifferenceBuffer = {};
IndexBuffer ={};
end
methods
function obj = indexFrame(obj, frame)
%for each channel in a frame
for c = 1:size(obj.Reference,3)
newFrame = frame(:,:,c);
if all(isequal(obj.Reference(:,:,c), newFrame))
% assign indexing value of -5 if two frames are thesame
differenceBuffer{c} = [];
indexBuffer{c} = [-5 numel(newFrame)];
else
% create difference buffer
newFrame = newFrame';
difference = obj.Reference(:,:,c)'~=newFrame;
index = find(difference);


closedIndex = [];
for i=2:numel(index)
d = index(i)-index(i-1)-1;
if d > 0 && d < obj.WindowSize
closedIndex  = vertcat(closedIndex , (index(i-1):index(i))');
end
end
index = unique(vertcat(index, closedIndex));
differenceBuffer{c} = newFrame(index);



%create index buffer
indexBuffer{c} = [];
```

```
count = 1;

%assign indexing value to new frame starting with
if ~ismember(1, index)
% same pixel compared with the reference
indexingValue = -1;
status = 0;
else
% different pixel compared with the reference
indexingValue = -2;
status = 1;
end
for i=2:numel(newFrame)
if (ismember(i, index) ~= status)
indexBuffer{c} = vertcat(indexBuffer{c}, [indexingValue, count]);
count = 1;
if ismember(i, index)
% different pixel values can be used from
% difference buffer
indexingValue = -3;
status = 1;
else
% same pixel valuse can be used from
% reference
indexingValue = -4;
status = 0;
end
else
count = count + 1;
end
end
indexBuffer{c} = vertcat(indexBuffer{c}, [indexingValue, count]);

end
end
obj.DifferenceBuffer = differenceBuffer;
obj.IndexBuffer =indexBuffer;
```

```
end

function obj = receiveFrame(obj, frame)

if obj.FrameIndex == 0
% receive first frame
obj.Reference = frame;
else
% receive other frames and index
obj = indexFrame(obj, frame);
obj.Reference = frame;
end
obj.FrameIndex = obj.FrameIndex + 1;
end

function data = compressData(obj)
if obj.FrameIndex == 1
data = {obj.Reference};
else
data = {obj.DifferenceBuffer, obj.IndexBuffer};
end

end
end
end
```

APPENDIX D

MATLAB CODE FOR THE CLIENT CLASS

```
classdef ClientClass

properties

FrameIndex = 0

Reference

TimeToBuild

SamePixels

BufferSize

end

methods

function obj  = reconstructFrame(obj, differenceBuffer, indexBuffer)

for c=1:size(obj.Reference,3)

ref = obj.Reference(:,:,c)';

newFrame = [];

buffInd1 = 1;

% for each channel

for i=1:size(indexBuffer{c},1)

switch  indexBuffer{c}(i,1)


case -1

% obtain 1st set of pixels from reference

refInd1 = numel(newFrame)+1;

refInd2 = refInd1 + indexBuffer{c}(i,2) - 1;

newFrame = vertcat(newFrame, ref(refInd1:refInd2)');


case -2

% obtain 1st set of pixels from difference
```

```
% buffer
buffInd2 = buffInd1 + indexBuffer{c}(i,2) - 1;
newFrame = vertcat(newFrame, differenceBuffer{c}(buffInd1:buffInd2));
buffInd1 = buffInd2 + 1;

case -3
% obtain set of pixels from difference
% buffer
buffInd2 = buffInd1 + indexBuffer{c}(i,2) - 1;
newFrame = vertcat(newFrame, differenceBuffer{c}(buffInd1:buffInd2));
buffInd1 = buffInd2 + 1;

case -4
% obtain set of pixels from reference
refInd1 = numel(newFrame)+1;
refInd2 = refInd1 + indexBuffer{c}(i,2) - 1;
newFrame = vertcat(newFrame, ref(refInd1:refInd2)');

case -5
% obtain all pixels from reference
newFrame = ref(1:end)';
end
end
obj.Reference(:,:,c) = uint8(reshape(newFrame, size(ref))');
end
end

function obj = receiveData(obj, data)
```

```
if obj.FrameIndex == 0
% receive 1st frame
obj.Reference = data{1};
else
% receive differnce buffer and index buffer and reconstruct
% frame
tic
obj = reconstructFrame(obj, data{1}, data{2});
elapsedTime = toc;
obj.TimeToBuild = [obj.TimeToBuild elapsedTime];
sP = numel(obj.Reference) - (numel(data{1}{1}) + numel(data{1}{2})
+numel(data{1}{3}));

obj.SamePixels = [obj.SamePixels (sP/numel(obj.Reference))*100];

db = numel(data{1}{1}) + numel(data{1}{2}) +numel(data{1}{3});
obj.BufferSize = [obj.BufferSize, (db/numel(obj.Reference))*100];
end
obj.FrameIndex = obj.FrameIndex + 1;
end
end
end
```

APPENDIX E

MATLAB CODE FOR THE SPATIO TEMPORAL INDEXING SERVER CLASS

```
classdef ImprovedServerClass
properties
FrameIndex = 0;
Reference
DifferenceBuffer = {};
IndexBuffer ={};
end
methods
function obj = indexFrame(obj, frame)
%for each channel in a frame
for c = 1:size(obj.Reference,3)
newFrame = frame(:,:,c);
if all(isequal(obj.Reference(:,:,c), newFrame))
% assign indexing value of -5 if two frames are thesame
differenceBuffer{c} = [];
indexBuffer{c} = [-5 numel(newFrame)];
else
% create difference buffer
newFrame = newFrame';
difference = obj.Reference(:,:,c)'~=newFrame;
index = find(difference);


closedIndex = [];
for i=2:numel(index)
d = index(i)-index(i-1)-1;
if d > 0 && d < obj.WindowSize
closedIndex  = vertcat(closedIndex , (index(i-1):index(i))');
end
end
index = unique(vertcat(index, closedIndex));
differenceBuffer{c} = newFrame(index);



%create index buffer
```

```
indexBuffer{c} = [];
count = 1;

%assign indexing value to new frame starting with
if ~ismember(1, index)
% same pixel compared with the reference
indexingValue = -1;
status = 0;
else
% different pixel compared with the reference
indexingValue = -2;
status = 1;
end
for i=2:numel(newFrame)
%checks if i is in difference buffer and status has
%changed
if (ismember(i, index) ~= status)
indexBuffer{c} = vertcat(indexBuffer{c}, [indexingValue, count]);
if size(indexBuffer{c},1) >2
ind1 = size(indexBuffer{c},1) - 1;
ind2 = size(indexBuffer{c},1);

if (indexBuffer{c}(ind1,1) == indexBuffer{c}(ind2,1)) &&...
(indexBuffer{c}(ind1,2)>0) &&(indexBuffer{c}(ind2,2)>0)
indexBuffer{c}(ind1,2) = indexBuffer{c}(ind1,2) + indexBuffer{c}(ind2,2);
indexBuffer{c}(ind2,:) = [];
end
end
count = 1;
if ismember(i, index)
% different pixel values can be used from
% difference buffer
indexingValue = -3;
status = 1;
oldVal = newFrame(i);
sCount = 1;
else
```

```
% same pixel valuse can be used from
% reference
indexingValue = -4;
status = 0;
end
else
if indexingValue == -3
if newFrame(i) == oldVal

sCount = sCount + 1;

else
if sCount >= 5
ind2 = find(index == (i-1));
ind1 = ind2 - sCount + 2;

differenceBuffer{c}(ind1:ind2) = [];
index(ind1:ind2) = [];
indexBuffer{c} = vertcat(indexBuffer{c}, [indexingValue, -sCount]);



else
indexBuffer{c} = vertcat(indexBuffer{c}, [indexingValue, sCount]);

ind1 = size(indexBuffer{c},1) - 1;
ind2 = size(indexBuffer{c},1);

if (indexBuffer{c}(ind1,1) == indexBuffer{c}(ind2,1)) &&...
(indexBuffer{c}(ind1,2)>0) &&(indexBuffer{c}(ind2,2)>0)
indexBuffer{c}(ind1,2) = indexBuffer{c}(ind1,2) + indexBuffer{c}(ind2,2);
indexBuffer{c}(ind2,:) = [];
end

end
count = 0;
oldVal = newFrame(i);
```

```
sCount = 1;
end
end
count = count + 1;
end
end
indexBuffer{c} = vertcat(indexBuffer{c}, [indexingValue, count]);
end
end
obj.DifferenceBuffer = differenceBuffer;
obj.IndexBuffer =indexBuffer;


end


function obj = receiveFrame(obj, frame)


if obj.FrameIndex == 0
% receive first frame
obj.Reference = frame;
else
% receive other frames and index
obj = indexFrame(obj, frame);
obj.Reference = frame;
end
obj.FrameIndex = obj.FrameIndex + 1;
end
function data = compressData(obj)
if obj.FrameIndex == 1
data = {obj.Reference};
else
data = {obj.DifferenceBuffer, obj.IndexBuffer};
end
end
end
end
```

APPENDIX F

MATLAB CODE FOR THE SPATIO TEMPORAL INDEXING CLIENT CLASS

```
classdef ImprovedClientClass
properties
FrameIndex = 0
Reference
TimeToBuild
SamePixels
BufferSize
end
methods
function obj  = reconstructFrame(obj, differenceBuffer, indexBuffer)
for c=1:size(obj.Reference,3)
ref = obj.Reference(:,:,c)';
newFrame = [];
buffInd1 = 1;
% for each channel
for i=1:size(indexBuffer{c},1)
switch  indexBuffer{c}(i,1)

case -1
% obtain 1st set of pixels from reference
refInd1 = numel(newFrame)+1;
refInd2 = refInd1 + indexBuffer{c}(i,2) - 1;
newFrame = vertcat(newFrame, ref(refInd1:refInd2)');

case -2
% obtain 1st set of pixels from difference
% buffer
buffInd2 = buffInd1 + indexBuffer{c}(i,2) - 1;
newFrame = vertcat(newFrame, differenceBuffer{c}(buffInd1:buffInd2));
buffInd1 = buffInd2 + 1;
```

```
case -3
% obtain set of pixels from difference
% buffer

if indexBuffer{c}(i,2) < 0
newFrame = vertcat(newFrame,...
repmat(differenceBuffer{c}(buffInd1), abs(indexBuffer{c}(i,2)), 1));
buffInd1 = buffInd1 + 1;
else
buffInd2 = buffInd1 + indexBuffer{c}(i,2) - 1;
newFrame = vertcat(newFrame, differenceBuffer{c}(buffInd1:buffInd2));
buffInd1 = buffInd2 + 1;
end

case -4
% obtain set of pixels from reference
refInd1 = numel(newFrame)+1;
refInd2 = refInd1 + indexBuffer{c}(i,2) - 1;
newFrame = vertcat(newFrame, ref(refInd1:refInd2)');

case -5
% obtain all pixels from reference
newFrame = ref(1:end)';
end

end

obj.Reference(:,:,c) = uint8(reshape(newFrame, size(ref))');
end
end

function obj = receiveData(obj, data)
```

```
if obj.FrameIndex == 0
% receive 1st frame
obj.Reference = data{1};
else
% receive differnce buffer and index buffer and reconstruct
% frame
tic
obj = reconstructFrame(obj, data{1}, data{2});
elapsedTime = toc;
obj.TimeToBuild = [obj.TimeToBuild elapsedTime];
sP = numel(obj.Reference) - (numel(data{1}{1}) + numel(data{1}{2}) +numel(data{1}{3}));

obj.SamePixels = [obj.SamePixels (sP/numel(obj.Reference))*100];

db = numel(data{1}{1}) + numel(data{1}{2}) +numel(data{1}{3});
obj.BufferSize = [obj.BufferSize, (db/numel(obj.Reference))*100];
end
obj.FrameIndex = obj.FrameIndex + 1;
end
end
end
```

APPENDIX G

ANALYSIS OF FRAME WHEN APPLIED ON A STANDARD LOW-MOTION VIDEO

Frame number	Standard time to build	Spatio-temp time to build	Standard same pixel	Spatio-temp same pixel	Difference Buffer-_standard	Difference Buffer-_spatio-temp
2	0.0321	0.0361	88.2945	89.399	11.7055	10.601
3	0.0281	0.0312	82.8254	84.5188	17.1746	15.4812
4	0.0282	0.0564	79.5996	81.3264	20.4004	18.6736
5	0.0417	0.0437	83.4539	84.5877	16.5461	15.4123
6	0.0394	0.0411	85.8005	86.7087	14.1995	13.2913
7	0.0194	0.0368	88.3756	88.8448	11.6244	11.1552
8	0.012	0.0161	90.1031	90.5034	9.8969	9.4966
9	0.0303	0.0313	90.7086	91.0608	9.2914	8.9392
10	0.0186	0.0206	86.6553	87.6912	13.3447	12.3088
11	0.0173	0.0193	86.2528	87.0936	13.7472	12.9064
12	0.017	0.0345	87.6446	88.028	12.3554	11.972
13	0.0199	0.0211	87.0419	87.552	12.9581	12.448
14	0.038	0.0228	79.2861	79.8321	20.7139	20.1679
15	0.0238	0.0258	70.2149	71.1917	29.7851	28.8083
16	0.0337	0.0468	38.7935	45.9686	61.2065	54.0314
17	0.0414	0.043	69.6407	70.7785	30.3593	29.2215
18	0.0382	0.0239	68.7801	69.6141	31.2199	30.3859
19	0.0384	0.0246	67.6588	68.9867	32.3412	31.0133
20	0.0209	0.0229	70.1672	70.975	29.8328	29.025
21	0.0225	0.0409	74.0057	74.4792	25.9943	25.5208
22	0.0225	0.0247	76.8247	77.611	23.1753	22.389
23	0.0234	0.0417	78.9332	79.8148	21.0668	20.1852
24	0.0211	0.039	78.2724	78.8331	21.7276	21.1669
25	0.0328	0.034	81.0064	81.5362	18.9936	18.4638
26	0.0189	0.0201	78.2193	78.7764	21.7807	21.2236
27	0.0239	0.0203	75.9993	76.5345	24.0007	23.4655
28	0.0367	0.022	75.6148	76.1669	24.3852	23.8331
29	0.0189	0.0199	73.8751	74.5606	26.1249	25.4394
30	0.0365	0.0219	70.2673	70.9872	29.7327	29.0128

31	0.0289	0.0396	46.1845	51.1999	53.8155	48.8001
32	0.0392	0.0371	78.6075	79.661	21.3925	20.339
33	0.0162	0.0177	80.5505	81.3045	19.4495	18.6955
34	0.0184	0.0368	78.5809	79.6681	21.4191	20.3319
35	0.0342	0.0194	77.5898	78.2161	22.4102	21.7839
36	0.0187	0.0362	76.2716	76.8893	23.7284	23.1107
37	0.0173	0.0182	76.4018	76.9603	23.5982	23.0397
38	0.0193	0.021	75.0667	75.7627	24.9333	24.2373
39	0.0176	0.0191	76.2834	76.9822	23.7166	23.0178
40	0.0182	0.0195	76.5948	77.1644	23.4052	22.8356
41	0.0371	0.0222	76.6766	77.3269	23.3234	22.6731
42	0.0181	0.0355	80.9971	81.4351	19.0029	18.5649
43	0.0182	0.0234	80.3307	80.6219	19.6693	19.3781
44	0.0203	0.0389	78.0418	78.9048	21.9582	21.0952
45	0.0201	0.0294	77.988	78.8155	22.012	21.1845
46	0.0272	0.0529	49.9523	54.4152	50.0477	45.5848
47	0.027	0.0315	78.5536	79.7406	21.4464	20.2594
48	0.0383	0.04	79.0544	79.9874	20.9456	20.0126
49	0.0239	0.0256	81.7999	82.556	18.2001	17.444
50	0.0188	0.0197	88.5431	89.0288	11.4569	10.9712
51	0.0167	0.0181	89.3548	90.0608	10.6452	9.9392
52	0.0339	0.0354	85.964	86.5989	14.036	13.4011
53	0.0147	0.0175	84.5138	85.1875	15.4862	14.8125
54	0.015	0.0345	87.3849	87.8949	12.6151	12.1051
55	0.0163	0.0173	84.86	85.3378	15.14	14.6622
56	0.0197	0.035	83.5493	84.0719	16.4507	15.9281
57	0.0244	0.0332	87.5355	88.1051	12.4645	11.8949
58	0.0184	0.0249	83.5722	84.2703	16.4278	15.7297
59	0.0218	0.0434	80.3403	81.1991	19.6597	18.8009
60	0.0261	0.0378	77.691	78.3366	22.309	21.6634
61	0.0407	0.0511	20.24	35.0293	79.76	64.9707
62	0.0209	0.0408	79.7001	81.2098	20.237	18.7902
63	0.03387	0.0354	82.763	83.4173	17.237	16.5827
64	0.0174	0.0429	80.68	81.8454	19.32	18.1546
65	0.017	0.0229	82.0747	82.8028	17.9253	17.1972
66	0.0149	0.0353	81.1514	81.8297	18.8486	18.1703
67	0.0374	0.0348	81.075	81.5965	18.925	18.4035
68	0.0327	0.0212	77.9847	78.7014	22.0153	21.2986
69	0.0209	0.0291	76.0833	76.6701	23.9167	23.3299
70	0.0196	0.0376	73.7445	74.3816	16.2555	25.6184
71	0.0187	0.0366	73.841	74.4329	26.159	25.5671
72	0.018	0.0194	75.839	76.5517	24.161	23.4483
73	0.0215	0.0227	77.2039	77.8585	22.7961	22.1415

74	0.0386	0.0233	82.522	83.1088	17.478	16.8912
75	0.0192	0.0204	79.8816	80.4261	20.0084	19.5739
76	0.0439	0.0345	54.2212	57.5431	45.7788	42.4569
77	0.0203	0.0254	81.2956	82.4717	18.7044	17.5283
78	0.0239	0.0211	79.0834	79.7334	20.9166	20.2666
79	0.0343	0.0208	73.6355	74.9609	26.3645	25.0391
80	0.0185	0.0208	75.0384	75.8928	24.9616	24.1072
81	0.0155	0.0172	77.6332	78.1171	22.3668	21.8829
82	0.0174	0.0186	79.3557	79.6	20.9643	20.4
83	0.0168	0.0193	78.3187	79.3528	21.6813	20.6472
84	0.0149	0.0164	80.4601	81.1431	19.5399	18.8569
85	0.0344	0.0363	78.7438	79.4178	21.2562	20.5822
86	0.0184	0.0197	77.3495	77.8911	22.6505	22.1089
87	0.0183	0.0196	78.3237	78.7879	21.6763	21.2121
88	0.0257	0.03	79.0228	79.8001	20.9772	20.1999
89	0.0343	0.0421	78.1268	78.8732	21.8732	21.1268
90	0.0355	0.0206	77.1859	77.8208	22.8141	22.1792
91	0.0408	0.0336	56.2489	60.3162	43.7511	39.6838
92	0.0271	0.0387	82.4986	83.3635	17.5014	16.6365
93	0.0406	0.026	78.8496	79.3704	21.1504	20.6296
94	0.0479	0.0247	76.5611	77.2828	23.4389	22.7172
95	0.024	0.0257	73.7571	74.5301	26.2429	25.4699
96	0.022	0.0232	78.789	79.3015	21.211	20.6985
97	0.0231	0.0254	804892	80.9788	19.5108	19.0212
98	0.0316	0.0243	853539	85.9235	14.6461	14.0765
99	0.0289	0.0226	85.5777	86.0096	14.4223	13.9904
100	0.0351	0.0204	86.3245	86.8386	13.6755	13.1614
101	0.0178	0.0199	84.1045	84.5881	15.8955	15.4119
102	0.0174	0.0199	84.5698	85.1728	15.4302	14.8272
103	0.0147	0.0166	82.6242	82.9761	17.3758	17.0239
104	0.0347	0.0205	76.8627	77.4148	23.1373	22.5852
105	0.0201	0.0243	74.496	75.4728	25.504	24.5272
106	0.0233	0.0331	53.5917	57.7375	46.4083	42.2625
107	0.0167	0.0186	81.0086	81.7504	18.9914	18.2496
108	0.0158	0.018	79.8887	80.4512	20.1113	19.5488
109	0.0343	0.0362	77.7695	78.5805	22.2305	21.4195
110	0.0176	0.0189	78.6451	79.284	21.3549	20.716
111	0.0187	0.0199	79.62226	80.0914	20.3774	19.9086
112	0.0182	0.0445	78.8349	79.78	21.1651	20.22
113	0.0205	0.0309	78.62	79.3231	21.38	20.6769
114	0.0365	0.0456	81.5423	82.0043	18.4577	17.9957
115	0.0218	0.019	81.82	82.3717	18.18	17.6283
116	0.0268	0.017	86.0336	86.3446	13.9664	13.6554

117	0.0354	0.0201	85.77	86.1857	14.23	13.8143
118	0.0343	0.0261	88.7666	89.206	11.2334	10.794
119	0.0175	0.0407	87.4537	87.9573	12.5463	12.0427
120	0.0156	0.0185	89.1027	89.5679	10.8973	10.4321
121	0.0266	0.0697	24.7607	38.4258	75.2393	61.5742
122	0.0346	0.0369	85.3205	86.3981	14.6795	13.6019
123	0.0307	0.0159	87.7504	88.4018	12.2496	11.5982
124	0.0205	0.0362	87.665	88.8476	12.335	11.1524
125	0.022	0.0168	86.8676	87.4315	13.1324	12.5685
126	0.0324	0.0367	85.9597	86.373	14.0403	13.627
127	0.0151	0.0159	87.4645	87.9276	12.5355	12.0724
128	0.0337	0.0352	85.3259	86.0196	14.6741	13.9804
129	0.014	0.0314	86.2395	86.6524	13.7605	13.3476
130	0.0358	0.0215	83.3129	83.8986	16.6871	16.1014
131	0.0166	0.0274	84.1978	84.8044	15.8022	15.1956
132	0.0335	0.0347	82.6382	83.1569	17.3618	16.8431
133	0.0312	0.0334	84.7735	85.082	15.2265	14.918
134	0.0162	0.0345	80.9652	81.4971	19.0348	18.5029
135	0.0381	0.0191	77.6056	78.4511	22.3944	21.5489
136	0.0446	0.0534	53.0296	57.6615	46.9704	42.3385
137	0.0166	0.0346	80.4002	81.3415	19.5998	18.6585
138	0.0345	0.0196	78.8273	79.78	21.1727	20.22
139	0.017	0.034	80.3181	81.2726	19.6819	18.7274
140	0.034	0.0169	79.2061	79.7976	20.7939	20.2024
141	0.0174	0.0184	80.6287	81.1133	19.3713	18.8867
142	0.0348	0.0194	83.7857	84.3897	16.2143	15.6103
143	0.0342	0.0189	82.8387	83.4237	17.1613	16.5763
144	0.0159	0.0164	85.729	85.4091	14.871	14.5909
145	0.0148	0.0322	84.0927	84.4464	15.9073	15.5536
146	0.0184	0.0247	82.2131	82.8308	17.7869	17.1692
147	0.0173	0.0348	83.2372	83.6816	16.7628	16.3184
148	0.0172	0.0182	83.3886	83.776	16.6114	16.224
149	0.0322	0.017	83.2049	83.8226	16.7951	16.1774
150	0.0177	0.0342	80.8759	81.4494	19.1241	18.5506
151	0.0562	0.0382	58.4678	61.9422	41.5322	38.0578
152	0.0167	0.0348	84.3312	84.9719	15.6688	15.0281
153	0.0141	0.0147	82.5919	82.8394	17.4081	17.1606
154	0.0324	0.0206	80.7191	81.566	19.2809	18.434
155	0.0147	0.0322	79.9709	80.458	20.0291	19.542
156	0.0301	0.0149	81.1427	81.5061	18.8573	18.4939
157	0.0262	0.0129	80.288	80.614	19.712	19.386
158	0.014	0.0152	80.7076	81.3189	19.2924	18.6811
159	0.0208	0.0368	80.3278	80.8188	19.6722	19.1812

160	0.0304	0.0317	81.5327	81.9409	18.4673	18.0591
161	0.0158	0.0194	81.269	81.8185	18.731	18.1815
162	0.0157	0.0232	80.3809	80.6922	19.6191	19.3078
163	0.0169	0.0179	79.0042	79.3831	20.9958	20.6169
164	0.0177	0.0242	80.1212	81.0204	19.8788	18.9796
165	0.0344	0.0357	82.1844	82.7935	17.8156	17.2065
166	0.0272	0.0505	57.3846	60.7216	42.6154	39.2784
167	0.0325	0.0344	87.3719	88.5133	12.6281	11.4867
168	0.0182	0.0193	87.7041	88.8857	12.2959	11.1143
169	0.0213	0.019	87.755	88.2343	12.245	11.7657
170	0.0365	0.0209	87.052	87.467	12.948	12.533
171	0.0191	0.0365	88.0829	88.4635	11.9171	11.5365
172	0.0175	0.0185	89.4445	89.8778	10.5555	10.1222
173	0.0164	0.0176	89.8968	90.4173	10.1032	9.5827
174	0.0146	0.0151	91.5451	91.9045	8.4549	8.0955
175	0.0122	0.0132	92.1254	92.6179	7.8746	7.3821
176	0.0313	0.0155	86.8927	87.2145	13.1073	12.7855
177	0.0147	0.0321	86.3009	86.7037	13.6991	13.2963
178	0.015	0.0172	86.5129	86.9939	13.4871	13.0061
179	0.0157	0.036	83.3215	83.8158	16.6785	16.1842
180	0.0165	0.0176	84.6347	85.105	15.3653	14.895
181	0.0279	0.0681	24.4447	37.1736	75.5553	62.8264
182	0.0182	0.0375	87.4878	88.8943	12.5122	11.1057
183	0.0189	0.02	87.2959	87.839	12.7041	12.161
184	0.0172	0.019	85.5555	86.3977	14.4445	13.6023
185	0.0145	0.0152	87.5104	87.8017	12.4896	12.1983
186	0.0162	0.0168	86.6962	87.0427	13.3038	12.9573
187	0.0302	0.0316	85.8087	86.5896	14.1913	13.4104
188	0.0245	0.0166	87.2701	87.825	12.7299	12.175
189	0.0316	0.0162	88.0639	88.6891	11.9361	11.3109
190	0.0147	0.0326	86.7668	87.3257	13.2332	12.6743
191	0.0158	0.0187	88.3286	88.7545	11.6714	11.2455
192	0.0129	0.0298	91.1889	91.7015	8.8111	8.2985
193	0.0301	0.0309	89.8007	90.0493	10.1993	9.9507
194	0.015	0.016	88.9732	89.4384	11.0268	10.5616
195	0.0342	0.0331	88.2239	88.6615	11.7761	11.3385
196	0.0314	0.0386	59.8377	64.0367	40.1623	35.9633
197	0.0167	0.034	88.0172	88.4975	11.9828	11.5025
198	0.0311	0.0157	86.1613	86.7779	13.8387	13.2221
199	0.0184	0.0197	84.1806	84.9453	15.8194	15.0547
200	0.0194	0.0372	81.3497	82.1173	18.6503	17.8827
201	0.0202	0.0181	83.6651	84.0636	16.3349	15.9364
202	0.028	0.0184	82.5686	83.0101	17.4314	16.9899

203	0.0263	0.0205	81.8741	82.7174	18.1259	17.2826
204	0.0319	0.0166	84.2602	84.8643	15.7398	15.1357
205	0.0143	0.0156	84.4858	85.0214	15.5142	14.9786
206	0.0135	0.0313	82.9083	83.5199	17.0917	16.4801
207	0.0135	0.0149	82.3168	82.7458	17.6832	17.2542
208	0.014	0.0152	81.6621	82.117	18.3379	17.883
209	0.0326	0.018	79.2474	80.124	20.7526	19.876
210	0.0175	0.0196	78.2079	79.3238	21.7921	20.6762
211	0.03	0.0532	57.2267	60.8597	42.7733	39.1403
212	0.0293	0.0319	81.4699	82.2978	18.5301	17.7022
213	0.0133	0.0145	81.4563	82.0273	18.5437	17.9727
214	0.0282	0.0145	81.2611	82.0614	18.7389	17.9386
215	0.0143	0.0157	80.9455	81.5362	19.0545	18.4638
216	0.017	0.017	81.2274	81.6299	18.7726	18.3701
217	0.0403	0.0365	77.5235	78.1311	22.4765	21.8689
218	0.0184	0.0201	77.2921	77.8786	22.7079	22.1214
219	0.0334	0.0178	77.583	78.1017	22.417	21.8983
220	0.018	0.0193	77.7756	78.4034	22.2244	21.5966
221	0.0147	0.0156	81.142	81.6245	18.858	18.3755
222	0.0291	0.0309	81.5029	81.8616	18.4971	18.1384
223	0.0334	0.0185	79.9655	80.595	20.0345	19.405
224	0.0151	0.0158	80.6452	80.9157	19.3548	19.0843
225	0.0355	0.0189	80.416	80.9214	19.584	19.0786
226	0.0401	0.0504	58.0327	61.274	41.9673	38.726
227	0.0152	0.0191	84.4424	85.1785	15.5576	14.8215
228	0.0172	0.0174	84.9034	85.4859	15.0966	14.5141
229	0.036	0.0373	83.5651	84.7226	16.4349	15.2774
230	0.019	0.02	86.6642	87.1452	13.3358	12.8548
231	0.0193	0.0203	83.9424	84.3707	16.0576	15.6293
232	0.0366	0.0176	84.4023	84.7387	15.5977	15.2613
233	0.0171	0.0347	82.2917	82.8979	17.7083	17.1021
234	0.0316	0.0157	81.3433	81.7242	18.6567	18.2758
235	0.0324	0.0189	81.3766	81.7845	18.6234	18.2155
236	0.015	0.0265	79.2384	79.7259	20.7616	20.2741
237	0.0135	0.0312	81.6184	81.976	18.3816	18.024
238	0.016	0.0192	81.2238	81.6995	18.7762	18.3005
239	0.0158	0.0169	81.8354	82.3189	18.1646	17.6811
240	0.0158	0.0331	83.1335	83.5582	16.8665	16.4418
241	0.0463	0.0531	24.8221	37.514	75.1779	62.486
242	0.0361	0.0225	81.4821	82.8433	18.5179	17.1567
243	0.0143	0.0154	84.3624	84.8478	15.6376	15.1522
244	0.042	0.0377	78.3076	79.3844	21.6924	20.6156
245	0.0347	0.0197	79.464	80.279	20.536	19.721

246	0.0156	0.0163	81.9071	82.2719	18.0929	17.7281
247	0.0131	0.0147	82.2684	82.8771	17.7316	17.1229
248	0.0171	0.0186	79.4888	80.0369	20.5112	19.9631
249	0.0166	0.0163	81.1323	81.6216	18.8677	18.3784
250	0.0319	0.0325	80.5889	81.3468	19.4111	18.6532
251	0.0338	0.0173	80.5782	81.4433	19.4218	18.5567
252	0.0303	0.0149	81.8419	82.3978	18	17.6022
253	0.0154	0.0166	80.7769	81.5079	19.2231	18.4921
254	0.0208	0.0192	80.6804	81.4878	19.3196	18.5122
255	0.0302	0.0355	82.4441	83.4878	17.5559	16.5122
256	0.0574	0.0395	57.7131	83.004	42.2869	16.996
257	0.034	0.0357	89.9653	61.1685	10.0347	38.8315
258	0.0161	0.0199	83.7688	90.9758	16.2312	9.0242
259	0.016	0.0235	80.3113	84.3718	19.6887	15.6282
260	0.0365	0.0215	81.1306	81.0136	18.8694	18.9864
261	0.0178	0.0233	79.737	81.7741	20.263	18.2259
262	0.0155	0.033	81.505	80.3059	18.495	19.6941
263	0.0178	0.0191	81.7999	81.9652	18.2001	18.0348
264	0.0165	0.0177	85.9612	82.49	14.0388	17.51
265	0.0154	0.0165	88.1679	86.4856	11.8321	13.5144
266	0.0161	0.0337	87.1653	88.8064	12.8347	11.1936
267	0.0323	0.0166	88.2174	87.6596	11.7826	12.3404
268	0.0137	0.0149	86.6323	87.0513	13.3677	12.9487
269	0.0406	0.0223	84.3294	85.1265	15.6706	14.8735
270	0.0198	0.0417	86.8106	87.3461	13.1894	12.6539
271	0.0534	0.0147	61.826	64.6149	38.174	35.3851
272	0.0286	0.0189	88.1137	88.8875	11.8863	11.1125
273	0.0136	0.0119	86.4035	86.906	13.5965	13.094
274	0.0105	0.0137	85.6269	86.0505	14.3731	13.9495
275	0.0123	0.0175	83.5073	84.0837	16.4927	15.9163
276	0.015	0.0368	81.349	81.8099	18.651	18.1901
277	0.0323	0.0352	79.0935	79.6043	20.9065	20.3957
278	0.02	0.0179	77.9679	78.5167	22.0321	21.4833
279	0.0324	0.0179	79.0476	79.6007	20.9524	20.3993
280	0.0314	0.0339	81.0477	81.3755	18.9523	18.6245
281	0.017	0.0181	82.7881	83.168	17.2119	16.832
282	0.0167	0.034	83.616	84.0927	16.384	15.9073
283	0.0343	0.0352	82.7124	83.1735	17.2876	16.8265
284	0.0234	0.026	79.9874	80.4612	20.0126	19.5388
285	0.0334	0.0348	80.5782	81.306	19.4218	18.694
286	0.0263	0.0498	60.7054	63.6504	39.2946	36.3496
287	0.032	0.0192	84.2721	85.035	15.7279	14.965
288	0.0117	0.0125	87.6223	87.92	12.3777	12.08

289	0.015	0.0339	82.7278	83.9291	17.2722	16.0709
290	0.0172	0.0183	81.7439	82.4455	18.2561	17.5545
291	0.0147	0.0155	82.2303	82.7795	17.7697	17.2205
292	0.0272	0.0135	84.993	85.3169	15.007	14.6831
293	0.0118	0.0277	86.0476	86.4078	13.3638	13.5922
294	0.0111	0.0117	86.6362	86.9652	14.1063	13.0348
295	0.0245	0.0123	85.8937	86.3425	11.3104	13.6575
296	0.0188	0.0203	88.6835	89.0428	11.3165	10.9572
297	0.0166	0.0111	90.6035	90.9722	9.3965	9.0278
298	0.0153	0.0096	91.6896	91.8711	8.3104	8.1289
299	0.0057	0.0061	99.2234	99.3249	0.7766	0.6751
300	0.0048	0.0029	99.3282	99.3282	0.6718	0.6718
301	0.0365	0.0695	15.719	34.8189	84.281	65.1811

APPENDIX H

ANALYSIS OF FRAME WHEN APPLIED ON A LOCAL LOW-MOTION VIDEO

Frame number	Standard time to build	Spatio-Temporal Time	Standard_ same Pixel	Spatio-Temp. same Pixels	Difference Buffer_ standard	Difference Buffer_ Spatio Temp.
2	0.022	0.0257	65.4396	72.2413	34.5605	27.7587
3	0.0299	0.03	64.1788	69.4572	35.8213	30.5428
4	0.0235	0.0238	68.153	72.269	31.8471	27.7311
5	0.0288	0.0322	71.8376	77.8639	28.1625	22.1361
6	0.0267	0.0297	65.4733	72.9785	34.5268	27.0215
7	0.0237	0.0264	65.5738	71.4091	34.4263	28.591
8	0.0171	0.0207	59.4841	65.002	40.516	34.998
9	0.0253	0.0288	57.4753	63.311	42.5248	36.689
10	0.0139	0.0178	62.7904	67.9086	37.2097	32.0915
11	0.0245	0.026	63.3371	64.163	36.663	35.8371
12	0.0252	0.0287	70.0605	77.2562	29.9396	22.7439
13	0.0266	0.0293	76.1254	86.0869	23.8747	13.9131
14	0.0297	0.0317	75.7013	79.2466	24.2988	20.7534
15	0.0315	0.0318	66.1444	75.857	33.8557	24.1431
16	0.0273	0.0312	61.7962	65.2607	38.2039	34.7393
17	0.0236	0.0253	72.2646	81.13	27.7355	18.87
18	0.0304	0.0335	72.1732	76.7201	27.8269	23.2799
19	0.0236	0.0273	71.7996	75.9339	28.2005	24.0662
20	0.0126	0.0131	71.8604	74.0377	28.1397	25.9623
21	0.0146	0.0153	59.1051	60.3616	40.895	39.6384
22	0.0292	0.0319	70.6179	73.707	29.3822	26.293
23	0.0217	0.0242	66.2519	73.5129	33.7482	26.4871
24	0.0288	0.0333	61.2299	69.0586	38.7702	30.9414
25	0.0164	0.0203	58.957	65.8949	41.0431	34.1051
26	0.0231	0.0267	73.4581	73.5562	26.542	26.4439
27	0.0246	0.0249	60.4868	68.919	39.5133	31.0811
28	0.0129	0.0132	60.258	69.4814	39.7421	30.5187
29	0.0243	0.0247	70.3064	78.0159	29.6937	21.9841
30	0.0194	0.0233	74.8744	75.301	25.1257	24.699
31	0.0132	0.0179	67.3178	71.0997	32.6823	28.9004
32	0.0218	0.0253	71.0407	78.0841	28.9594	21.916
33	0.016	0.0167	60.0585	67.3536	39.9416	32.6465

34	0.0147	0.0183	76.0558	78.2986	23.9443	21.7015
35	0.0163	0.0168	67.8043	70.4949	32.1958	29.5052
36	0.0151	0.0157	70.5813	77.3116	29.4188	22.6884
37	0.016	0.0192	57.7179	62.4928	42.2822	37.5072
38	0.0131	0.0147	73.1707	79.4079	26.8294	20.5922
39	0.0247	0.028	71.959	74.3235	28.0411	25.6766
40	0.0178	0.0215	59.3904	61.1616	40.6097	38.8384
41	0.0228	0.0257	67.4876	75.784	32.5125	24.2161
42	0.0259	0.0296	63.5033	71.1725	36.4968	28.8275
43	0.022	0.0232	67.9156	77.2604	32.0845	22.7396
44	0.0228	0.0264	64.9643	66.0432	35.0358	33.9569
45	0.021	0.0258	65.2885	67.1108	34.7116	32.8893
46	0.0147	0.019	60.6014	61.5924	39.3987	38.4077
47	0.0219	0.0223	62.0944	66.992	37.9057	33.008
48	0.029	0.0308	57.3974	59.3298	42.6027	40.6702
49	0.0294	0.0312	75.4602	84.4191	24.5399	15.581
50	0.0175	0.021	70.0606	71.0515	29.9395	28.9485
51	0.0163	0.0193	75.6389	76.0806	24.3612	23.9195
52	0.0233	0.0273	60.2569	65.8298	39.7432	34.1702
53	0.0248	0.0266	75.4086	83.1335	24.5915	16.8665
54	0.0204	0.0215	72.8798	75.9992	27.1203	24.0008
55	0.0163	0.0167	68.5345	70.3244	31.4656	29.6757
56	0.0308	0.0347	65.7874	69.1769	34.2127	30.8231
57	0.0139	0.0149	62.1389	64.2404	37.8612	35.7597
58	0.0143	0.0163	72.0256	77.1271	27.9745	22.8729
59	0.015	0.0178	61.56	70.6237	38.4401	29.3764
60	0.0155	0.0167	58.2704	64.5596	41.7297	35.4404
61	0.0244	0.0276	72.3332	73.3486	27.6669	26.6515
62	0.0235	0.0259	70.4107	74.3192	29.5894	25.6808
63	0.0133	0.014	71.2909	71.8371	28.7092	28.163
64	0.0305	0.0344	69.8279	74.8407	30.1722	25.1594
65	0.0265	0.027	65.3676	69.6848	34.6325	30.3152
66	0.0267	0.0282	64.8019	74.7775	35.1982	25.2226
67	0.0135	0.0147	73.3095	81.4255	26.6906	18.5746
68	0.0291	0.0318	63.3352	68.1917	36.6649	31.8083
69	0.0306	0.031	73.2774	82.2219	26.7227	17.7781
70	0.0315	0.0336	72.7681	74.1436	27.232	25.8565
71	0.0291	0.0296	74.0319	77.932	25.9682	22.0681
72	0.0276	0.0282	67.0994	76.3729	32.9007	23.6271
73	0.0223	0.0262	69.6999	78.8748	30.3002	21.1252
74	0.0157	0.0172	76.0045	83.1403	23.9956	16.8598
75	0.0201	0.0231	65.8659	72.0493	34.1342	27.9507
76	0.0149	0.0197	58.187	61.6199	41.8131	38.3801
77	0.0129	0.015	74.3216	83.6819	25.6785	16.3181
78	0.0307	0.0341	69.6104	70.8582	30.3897	29.1419
79	0.0182	0.0219	64.0881	71.394	35.912	28.6061

80	0.018	0.0202	76.9267	83.3915	23.0734	16.6086
81	0.0188	0.0221	61.4701	69.8016	38.53	30.1985
82	0.0214	0.022	70.0357	74.0185	29.9644	25.9816
83	0.025	0.0296	69.0865	76.5847	30.9136	23.4154
84	0.0127	0.0137	64.7316	73.0838	35.2685	26.9163
85	0.0288	0.0301	59.8304	63.055	40.1697	36.9451
86	0.0232	0.0272	57.4893	63.012	42.5108	36.9881
87	0.029	0.0314	65.4089	75.2002	34.5912	24.7999
88	0.0191	0.0229	60.6687	66.1617	39.3314	33.8383
89	0.021	0.023	71.5022	74.8064	28.4979	25.1937
90	0.0133	0.0147	64.3939	70.5886	35.6062	29.4114
91	0.0157	0.0159	73.8179	77.4242	26.1822	22.5758
92	0.0252	0.0286	71.6712	79.2363	28.3289	20.7637
93	0.0187	0.0209	68.4072	72.5462	31.5929	27.4539
94	0.0299	0.0321	60.5238	65.4472	39.4763	34.5528
95	0.0146	0.0176	76.1343	83.0818	23.8658	16.9183
96	0.0316	0.0319	62.2931	72.0204	37.707	27.9796
97	0.0228	0.0244	75.4783	78.7558	24.5218	21.2442
98	0.0261	0.03	61.4621	69.8401	38.538	30.16
99	0.0318	0.0353	64.4579	71.8486	35.5422	28.1514
100	0.0179	0.0185	58.7367	68.2784	41.2634	31.7216
101	0.0204	0.021	69.789	70.1082	30.2111	29.8918
102	0.0214	0.0218	60.599	64.1677	39.4011	35.8324
103	0.0272	0.0273	57.8877	64.5142	42.1124	35.4858
104	0.0283	0.0304	71.4501	74.2651	28.55	25.7349
105	0.0142	0.0175	63.9354	66.2392	36.0647	33.7608
106	0.0157	0.0194	70.199	77.3103	29.8011	22.6898
107	0.0193	0.022	64.664	70.9098	35.3361	29.0903
108	0.0134	0.0139	69.5336	75.4397	30.4665	24.5604
109	0.0225	0.0256	57.4196	64.024	42.5805	35.976
110	0.0188	0.0195	75.198	75.6736	24.8021	24.3265
111	0.0157	0.0164	72.9978	76.4857	27.0023	23.5144
112	0.0163	0.0168	71.9036	76.417	28.0965	23.583
113	0.03	0.0307	73.2489	75.658	26.7512	24.3421
114	0.0255	0.0263	64.6528	71.8032	35.3473	28.1968
115	0.0214	0.0224	69.3322	77.8941	30.6679	22.106
116	0.0301	0.0317	68.4965	71.3116	31.5036	28.6884
117	0.0143	0.0159	67.5877	74.8982	32.4124	25.1019
118	0.0269	0.028	62.488	63.8657	37.5121	36.1344
119	0.0267	0.0279	61.9592	70.3265	38.0409	29.6736
120	0.0233	0.0277	66.0194	67.4054	33.9807	32.5946
121	0.0159	0.0194	61.5409	67.423	38.4592	32.577
122	0.024	0.0267	73.0756	76.7372	26.9245	23.2628
123	0.0181	0.0191	76.7087	84.7763	23.2914	15.2237
124	0.0149	0.0159	57.5865	62.6243	42.4136	37.3758
125	0.0164	0.0168	67.6999	72.5959	32.3002	27.4042

126	0.0298	0.0344	58.7282	67.4987	41.2719	32.5014
127	0.0137	0.0172	73.0285	76.5599	26.9716	23.4402
128	0.017	0.0198	76.7695	81.264	23.2306	18.7361
129	0.0133	0.0149	58.3256	67.9609	41.6745	32.0392
130	0.0209	0.0217	75.7746	76.1976	24.2255	23.8025
131	0.0125	0.0156	57.3502	67.0798	42.6499	32.9203
132	0.0298	0.0348	70.6634	72.5555	29.3367	27.4446
133	0.0161	0.017	72.6614	79.3326	27.3387	20.6675
134	0.0141	0.0154	67.6694	73.5338	32.3307	26.4663
135	0.0183	0.0203	74.6938	81.445	25.3063	18.5551
136	0.0212	0.0216	74.9667	78.577	25.0334	21.4231
137	0.0142	0.0177	69.5054	75.7082	30.4947	24.2919
138	0.0318	0.0338	59.744	67.8555	40.2561	32.1445
139	0.0188	0.0237	61.3427	61.5353	38.6574	38.4648
140	0.0181	0.0201	60.6295	61.4682	39.3706	38.5318
141	0.0135	0.0166	57.823	67.5711	42.1771	32.429
142	0.0181	0.0189	59.1255	65.639	40.8746	34.3611
143	0.0132	0.0151	69.3155	71.6279	30.6846	28.3722
144	0.0222	0.023	75.7799	79.8148	24.2202	20.1853
145	0.0272	0.031	64.0758	65.296	35.9243	34.7041
146	0.0246	0.029	65.1992	67.8836	34.8009	32.1164
147	0.014	0.0158	76.6736	79.2521	23.3265	20.7479
148	0.0138	0.0173	75.8982	79.2149	24.1019	20.7852
149	0.0275	0.029	70.5195	72.0419	29.4806	27.9582
150	0.03	0.0326	76.7527	80.2328	23.2474	19.7673
151	0.0227	0.0269	72.3233	73.5399	27.6768	26.4602
152	0.0144	0.0174	63.7206	72.5622	36.2795	27.4379
153	0.0284	0.0301	70.2343	71.1771	29.7658	28.823
154	0.0189	0.0204	61.87	71.1704	38.1301	28.8297
155	0.018	0.0203	62.8968	66.887	37.1033	33.1131