VRIJE UNIVERSITEIT

Network Streaming and Compression for Mixed Reality Tele-Immersion

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam
op gezag van de rector Magnificus
prof.dr. V. Subramaniam,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de Faculteit der Exacte Wetenschappen
op maandag 23 januari 2017 om 13.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Rufail Negatu Mekuria
geboren te Amsterdam

promotor: prof.dr. D.C.A. Bulterman,
copromotor: dr. P.S. Cesar

promotiecommissie:

Dr. Thilo Kielmann, Vrije Universiteit Amsterdam

Prof dr. Rob van der Mei, Vrije Universiteit Amsterdam, Centrum Wiskunde Informatica

Prof. dr.ir. Maarten van Steen, Universiteit Twente

Prof. dr. Klara Nahrstedt, University of Illinois

Prof.dr. Fernando Manuel Bernardo Pereira, U. Lisboa, Instituto de Telecomunicações

ISBN 978-90-9030147-1

Acknowledgements

This thesis would not have been possible without the help, advice and feedback of many others. I thank Dick Bulterman for giving me the chance to work in an environment with a lot of freedom in collaboration with experts in 3D Multimedia Systems. I thank Pablo Cesar for supporting me throughout the years and investing a lot of resources (time, money, supervision) in my work. Mostly, I thank him for letting me on my own track as a researcher to work on a timely topic and helping me all the time when in need. The Reverie project funded me throughout and forced me to spend time on 3D systems and provided a large foundation to this work. Therefore, I sincerely thank the European Union for supporting this action. Perhaps, the most special thing about my experience in this thesis is that I feel like I have benefitted from the advice and collaboration of the most prominent experts in 3D Media and I want to thank them. First, I thank the experts that I met and worked with in the Reverie project: Aljoscha Smolic, Peter Eisert, Tamy Boubekeur, Slim Essid, Petros Daras, Dimitrios Alexiadis, Noel O Connor and Ebroul Izquierdo. Thank you for your support and advice in each respective area of expertise. I thank prof. Klara Nahrstedt for her advice and appreciation of my work and having her in my group at the University of Illinois. I also very much thank Cha Zhang and Dinei Florencio at Microsoft research for recognising my work and the invited presentation at IEEE ICASSP. I thank very much the chairs of the MPEG group Marius Preda and Leonardo Chiariglione for collaborations in MPEG. I thank you for assisting our entry into the MPEG standardisation organization and supporting my exploration activities. I very much thank Gauthier Lafruit, Francisco Moran, Euee Yang, Itaru Taneko, Marius Preda, Phil Chou, Zhu Li, Joern Ostermann and Leonardo Chiariglione for providing very contructive feedback during MPEG meetings. I thank Lazar Bivolarsky and Christian Tulvan for the close collaboration in the AhG on 3D Graphics compression in MPEG. I want to thank prof. Touradj Ebrahimi, prof. Fernando Pereira and prof. Peter Schelkens for inviting me in the JPEG AhG on JPEG innovations. The discussions on JPEG PLENO have been useful to this work and your consideration for my inputs to the point cloud part of the JPEG PLENO requirements is appreciated. I very much thank Atanas Gotchev and Gregor Schiele for invitations in Tampere and Duisburg respectively during my PhD. But mostly, I thank all the people that I could work together with in the large scale software developed

in the Reverie project: Michele Sanna, Patriciya Klavdianos, Frank Foerster, Dimitrios Alexiadis, Kostas Apostolakis, Christoph Stevens, Steven Poulakos, Julie Wall, Brian Ravenet, Philipp Fechteler, Emanuele Quachio, Antonella Frisiello, Ioanis Doumanis, Lemonia Agriyou and Stefano Asioli. This was perhaps the greatest, most fun, instructive and at times most annoying part of the PhD. Without this big shared effort, much of the experimental work in this thesis would have been difficult to achieve. Last I thank the (current, past and future) group members of distributed and interactive systems for their enthousiasm and discussion on topics in multimedia communications within the group. But mostly I thank my family and my parents for supporting me throughout the years.

Amsterdam, 30 December 2015

Abstract

The Internet is used for distributed shared experiences such as video conferencing, voice calls (possibly in a group), chatting, photo sharing, online gaming and virtual reality. These technologies are changing our daily lives and the way we interact with each other. The current rapid advances in 3D depth sensing and 3D cameras are enabling acquisition of highly realistic reconstructed 3D representations. These natural scene representations are often based on 3D point clouds or 3D Meshes. Integration of these data in distributed shared experiences can have a large impact on the way we work and interact online. Such shared experiences may enable 3D Tele-immersion and Mixed reality that combine real and synthetic contents in a virtual world. However, it poses many challenges to the existing Internet infrastructure. A large part of the challenge is due to the shear volume of reconstructed 3D data. End-to-End Internet connections are bandlimited and currently cannot support real-time end-to-end transmission of uncompressed 3D point cloud or mesh scans with hundreds of thousands of points (over 15 Megabytes per frame) captured at a fast rate (over 10 frames per second). Therefore the volume of the 3D data requires development of methods for efficient compression and transmission, possibly taking application and user specific requirements into account. In addition, sessions often need to be setup between different software and devices such as browsers, desktop applications, mobile applications or server side applications. For this reason interoperability is required. This introduces the need for standardisation of data formats, compression techniques and signalling (session management). In the case of mixed reality in a social networking context, users may use different types of reconstructed and synthetic 3D content (from simple avatar commands, to highly realistic 3D reconstructions based on 3D Mesh or Point Clouds). Therefore such signalling should take into account that different types of user-setups exist, from simple to very advanced, that can each join shared sessions and interact.

This thesis develops strategies for compression and transmission of reconstructed 3D data in Internet infrastructures. It develop three different approaches for the compression of 3D meshes and a codec for time varying 3D point clouds. Further, it develops an integrated 3D streaming framework that includes session management and signalling, media synchronization and a generic API for sending streams based

on UDP/TCP based protocols. Experiments with these components in a realistic integrated mixed reality system with state of art rendering and 3D data capture investigates the specific system and user experience issues arising in the integration of these sub components.

The first mesh codec is based on taking blocks of the mesh geometry list based on local per block differential encoding and coding the connectivity based on a repetitive pattern resulting from the reconstruction system. The main advantage of this approach is simplicity and parallelizability. The codec is integrated in an initial prototype for 3D immersive communication that includes a communication protocol based on rateless coding based on LT codes and a light 3D rendering engine that includes an implementation for global illumination.

The second mesh codec is a connectivity driven approach. It codes the connectivity in a similar manner as the first codec but with entropy encoding added based on deflate/inflate (based on the popular zlib library). This addition makes the connectivity codec much more generically applicable. Subsequently it traverses the connectivity to apply differential coding of the geometry. The differences between connected vertices are then quantized using a non linear quantizer. We call this approach delayed quantization step late quantization (LQ). This approach resulted in reduced encoding complexity at only a modest degradation in R-D performance compared to the state of the art in standardized mesh compression in MPEG-4. The resulting codec performs over 10 times faster encoding in practice compared to the latter. The codec is used to achieve real-time communication in a WAN/MAN scenario in a controlled IP network configuration. This includes real-time rendering and rateless packet coding of UDP Packet data. The streaming pipeline has been optimized to run in real time with varying frame rates that often occur in 3D Tele-immersion and mixed reality. In addition it was tested in different network conditions using a LIFO (Last in First Out) approach that optimizes the pipeline. In addition, it has been integrated with highly realistic rendering and 3D capture.

The third codec is based on a geometry driven approach. In this codec the geometry is coded first in an octree fashion and then the connectivity representation is converted to a representation that indexes voxels in the octree grid. This representation introduces correlation between the indices that is exploited using a vector quantization scheme. This codec enables real-time coding at different levels of detail (LoD) and highly adaptive bit-rates. This codec is useful when the 3D immersive virtual

room is deployed in the Internet when bandwidths may fluctuate heavily and are more restricted compared to the controlled WAN/MAN scenario. In addition, it is suitable for 3D representations that can be rendered at a lower level of detail, such as participants/objects rendered at a distance in the 3D Room.

Next, the focus shifts towards 3D Point Clouds instead of 3D meshes. 3D Point Clouds are a simpler representation of the 3D reconstructions. The thesis develops a codec for time-varying point clouds. It introduces a hybrid architecture that combines an octree based intra codec with lossy inter-prediction and lossy attributes coding based on mapping attributes to a JPEG image grid. It also introduces temporal inter-prediction. The predictive frames are reduced up to 30% and the colours up to 90% in size compared to the current state of the art in real-time point cloud compression. Subjective experiments in a realistic mixed reality virtual world framework developed in the Reverie project showed no significant degradation in the resulting perceptual quality.

In the last phase of this thesis, the complete 3D tele-immersive streaming platform is further developed. Additions include signalling support for 3D streaming (session management) that supports terminal scalability for light clients (render only) up to very heavy clients. This is done by signalling the local modular configuration in advance via the XMPP Protocol. Further, the streaming platform architecture presents an API where different stream types suitable to different 3D capture/reconstruction platforms (i.e. 3D audio, 3D visual, 3D animation) can be created. As the platform includes a distributed virtual clock, mechanisms to perform inter-stream and inter-sender media synchronization can be deployed at the application layer. Therefore, synchronization of compressed 3D audio streams in an audio playout buffer was implemented in a 3D audio rendering module. We also implemented a mesh and point cloud playout buffer in the module for synchronized rendering. This mesh playout buffer enables inter-sender synchronization between different incoming visual streams. In addition, the system includes simple publish and subscribe transmission protocol for small messages based on web socket (through a real-time cloud broker). In addition publish and subscribe based on the XMPP and UDP protocols was implemented. These publish and subscribe messages are particularly suitable for 3D animation commands and AI data exchange.

All components developed throughout this thesis have been integrated with 3D capture/rendering modules and in a social networking context in the larger Reverie 3D

Tele-immersive framework. Field trials of this system in different scenarios have shown the benefits of highly realistic live captured 3D data representations. This further highlights the importance of this work. The components developed in this thesis and their integration outlines many of the significant challenges encountered in the next generation of 3D tele-presence and mixed reality systems. These insights have contributed to the development of requirements for new international standards in the consortia MPEG (Moving Picture Experts Group) and JPEG (Joint Picture Experts Group). In addition, the developed codec and quality metrics for point cloud compression have been accepted as a base reference software model for a novel standard on point cloud compression in MPEG and are available in the MPEG code repository and online on github.

Keywords

3D Tele-immersion, 3D Mixed Reality, 3D Networking, 3D Mesh Compression, 3D Point Cloud Compression, 3D Rendering

Nederlandse Samenvatting

Het Internet wordt in toenemende mate gebruikt voor gedeelde ervaringen op afstand zoals video conferenties, telefoon gesprekken (mogelijk in een groep), tekst berichten, spelletjes en virtuele werelden. Deze technologien veranderen ons dagelijks leven en de manier waarop we met elkaar interakteren. De huidige snelle ontwikkelingen in 3D camera technologie zoals time of flight (ToF) camera, Light Detection and Ranging (LiDaR), 3D opnames met mobiele apparaten en 3D reconstructie met behulp van de grafische processor (GPU) maken het mogelijk zeer realistische 3D opnamen te maken. Deze opnames zijn vaak gerepresenteerd als punten wolk of 3D mesh model. Het hergebruik van deze data in gedeelde ervaringen op afstand in het Internet kan een grote impact hebben op de manier waarop we werken en online communiceren. Zulke ervaringen op afstand omvatten 3D immersie en gemengde realiteit. Desalnietemin, biedt dit veel uitdagingen voor de bestaande Internet infrastructuur. Een groot deel van deze uitdaging komt voort uit het volume van de 3D data. Internet verbindingen hebben altijd een beperkte bandbreedte. Huidige verbindingen kunnen ware tijd verbindingen voor ongecomprimeerde 3D puntenwolken en mesh modellen met honderd duizenden punten (meer dan 15 Megabytes per frame) opgenomen met frequenties boven de 10 Hz niet goed ondersteunen. Dit vraagt om de ontwikkeling van efficiente methodes voor compressie en transmissie van dit type data. Dit is de grootste uitdaging waar ik mij op richt in dit proefschrift. Wat ook belangrijk is, is dat verbindingen opgezet moeten worden tussen verschillende software en/of apparaten zoals Internet browsers, computer toepassingen, mobiele toepassingen en toepassingen op server systemen. Dit maakt inter operabiliteit belangrijk en introduceert de behoefte voor standardisatie van data formaten en compressie technieken. In het geval van gemengde realiteit toepassingen in sociale webdiensten, is het mogelijk dat gebruikers verschillende typen 3D Data willen gebruiken (bijvoorbeeld een eenvoudige avatar of een zeer realistische 3D mesh of point cloud opname). Signalering en sessie beheer dient hier rekening mee te houden. Deze moet er dus van doordacht zijn dat verschillende typen gebruikers bestaan, van eenvoudig to zeer gavanceerd, en dat deze in gezamelijke sessies kunnen treden en interakteren. Dit is een grote uitdaging op het gebied van sessiemanagement en signalering.

In dit proefschrift ontwikkelen we strategieen voor de compressie en transmissie van 3D data in de huidige Internet infrastructuur. Ik ontwikkel drie verschillende methoden voor de compressie van 3D mesh modellen en een compressie methode voor tijds varierende punten wolken. Vervolgens ontwikkel ik een compleet en geintegreerd raamwerk voor 3D transmissie, sessie beheer en media synchronisatie. Dit platform maakt gebruik van de UDP en TCP protocollen en omvat een generieke programmeer interface (API) voor het opzetten van verbindingen. Ik gebruik dit raamwerk om te experimenteren in een realistisch 3D mixed reaility systeem met 3D reconstructie, 3D rendering en kunstmatige intelligentie. Dit stelt mij in staat om een beeld te krijgen van de specifieke gebruikers ervaring en het effect van netwerk communicatie en 3D representatie.

De eerste codec is gebaseerd op het in blokken opdelen van de vertex-list van de mesh en het uitvoeren van differentiele codering op deze geometrie. De connectiviteits codering gebaseerd op een run length coding van een herhalend patroon geintroduceerd door het reconstructie systeem. Het grote voordeel van dit coderings systeem is dat het eenvoudig is en makkelijk te paralleliseren. De codec is geintegreerd in een prototype voor 3D tele communicatie. Dit is inclusief een communicatie protocol gebaseerd op rateless codering LT codes en een raamwerk voor 3D weergave met globale verlichtings berekeningen.

De tweede 3D codec is connectiviteit gedreven. De verbindingen (driehoeken) in de mesh worden eerst gecodeerd op een soortgelijke manier als in de eerste codec. Bovendien is er entropie codering gebaseerd op het deflatie en inflatie algoritme van zlib toegevoegd voor de connectiviteit. Dit maakt de codec algemener toepasbaar. Vervolgens wordt de connectiviteit doorlopen en worden alle posities differentieel gecodeerd. Vervolgens pas ik quantisatie toe met een niet lineaire quantisator. Ik noem deze techniek late quantisatie (LQ). Het grote voordeel is dat ik op deze manier differentiele waardes sneller en meer accuraat kan coderen, en compact kan representeren zonder extra entropie codering. In vergelijking met de MPEG-4 standaard voor mesh compressie behaal ik een 10 maal hogere encoderings snelheid met een vergelijkbare bit-rate/distortie karakteristiek. Ik gebruik de codec om een ware tijd verbinding op te zetten in een MAN/WAN scenario met 3D rendering en een rateless UDP pakket fout codering schema. Ik opimaliseer de volledige pipeline met varierende input framerates door middel van een Last in First Out (LIFO) schema. Bovendien integreer ik het systeem met verbeterde 3D rendering wat resulteerde in een prototype voor interaktieve 3D communicatie.

De derde codec is gebaseerd op een geometrisch gedreven benadering. De geometrie wordt eerst gecodeerd, en later gebruikt om de connectiviteit te coderen. In deze codec worden de geometrische posities eerst in een octree (achtboom) gecodeerd. Vervolgens word de connectiviteit geconverteerd naar een representatie op basis van voxel indices in de voxel grid van deze octree. Deze representatie introduceert correlatie tussen indices die we uitbuiten met een vector quantizatie schema. Deze codec staat ons toe, door middel van simplificatie in de bladen van de octree, om veel verschillende niveaus van detail te coderen (LoD). Dit resulteert in zeer makkelijk verstelbare bit-rates. Deze codec is nuttig in een 3D mixed reality virtuele kamer in het Internet, waar de beschikbare bandbreedte fluctueert. 3D Mesh data kan nu op een slimme manier gecompromimeerd worden door rekening te houden met de beschikbare bandbreedte voor het sturen van data. Daarnaast is deze codec nuttig wanneer de 3D representatie met lager detailerings niveau weergegeven kan worden, zoals bijvoorbeeld voor 3D reconstructies/personen op grotere afstand in de 3D kamer.

Vervolgens richt ik mij op 3D punten wolken in plaats van 3D mesh modellen. Dit zijn eenvoudigere 3D reconstructies van objecten, mensen en omgevingen. Ik introduceer een codec voor in de tijd varierende punten wolken voor 3D communicatie in mixed reality. De codec volgt een hybride architectuur als bekend van video codering en combineert een octree gebaseerd intra codec met temporele inter-frame schatting. Bovendien bevat de codec een nieuw methode voor codering van de kleuren met behulp een transformatie naar een JPEG beeld raster. Experimenten in het Reverie mixed reality systeem laten zien dat de nadelige effecten van de codering niet herkenbaar zijn voor gebruikers. Dit terwijl de inter-frame schatting in 30% lagere bit-rates resulteerd en de kleur codering in rond de 90 % lagere bit rates (voor aleen de kleuren).

In de laaste fase richt ik mij op de ontwikkeling van een volledig platform voor netwerk streaming in 3D tele-immersion/mixed reality. Toevoegingen zijn onder meer een protocol voor het opzetten van verbindingen tussen heterogene eindstations. Het protocol is gebaseerd op het vrij beschikbare XMPP protocol. Dit platform stelt een programmeer interface (API) beschikbaar voor de verschillende 3D opname, reconstructie en render platforms. Het bezit ook een virtuele gedistribueerde klok, nuttig voor gedistribueerde media synchronizatie. Ik heb dit raamwerk geintegreerd met 3D Audio en 3D rendering modules. Door middel van buffer strategieen kan ik zowel synchronizatie tussen 3D audio en de 3D mesh/point

clouds bereiken. Bovendien heb ik in dit platform nog methodes voor publish en subscribe toegevoegd via websockets. Dit is met name nuttig voor 3D animatie data en uitwisseling van gegevens over de kunstmatige intelligentie.

Alle componenten in dit proefschrift zijn geintegreerd met 3D Capture/Rendering en een sociale netwerk context in het Reverie 3D Tele-immersive mixed reality systeem. Praktische gebruikers exprimenten met dit systeem hebben de toegevoegde waarde van realistische 3D reconstucties gebaseerd op punten wolken en mesh modellen in verschillende scenarios aangetoond. Dit benadrukt nogmaals de relevantie van mijn werk. Bovendien hebben de inzichten opgedaan in dit werk bijgedragen aan nieuwe en gewijzigde programma's van eisen (requirements) voor internationale standaarden voor media compressie. Puntenwolken en tijds varierende mesh compressie zijn aan het programma van eisen van MPEG toegevoegd. Bovendien heeft dit proefschrift bijgedragen aan het programma van eisen voor een nieuwe standaard binnen het JPEG consortium (Joint Picture Experts Group), voor standaard PLENO. Bovendien is de methodologie voor kwaliteits en bit-rate evaluatie voor punten wolken grotendeels overgenomen in MPEG. De code voor de puntenwolk coder ontwikkeld in dit proefschrift vormt de basis voor een nieuwe puntenwolk codec in MPEG. De broncode is beschikbaar in het MPEG versie controle systeem als test en evaluatie software en online op het github.

Sleutel Woorden

3D Tele-portatie, 3D Mixed Reality, 3D Netwerk protocollen, 3D Mesh Compressie, 3D Punten Wolk Compressie, 3D Weergave

Contents

Acknowledgements ..... **v**

Abstract ..... **viii**

Keywords ..... **xi**

Nederlandse Samenvatting ..... **xii**

Sleutel Woorden ..... **xv**

Chapter 1 Introduction **1**

1.1 Motivation 3

1.2 Challenges 4

1.3 Large Scale Tele-Immersive Systems Architecture 6

1.4 Research Questions 7

1.5 Contributions 10

1.6 Thesis Outline 12

Chapter 2 Related Work **17**

2.1 Technological Background and Motivation 17

2.1.1 3D Scene Capture 17

2.1.2 3D Rendering 22

2.1.3 3D Media Internet 27

2.1.4 3D Media Compression 36

2.2 3D Audio Visual Capture Model 40

2.3 Related Work on 3D Tele-Immersive Systems 43

Chapter 3 3D Tele-immersion with Live Captured Geometry using Block Based Mesh Compression **45**

3.1 Introduction 46

3.2 Motivation and Research Question 49
3.3 Related Work 51
3.3.1 Compression of Triangle Meshes 51
3.3.2 Transmission of Triangle Mesh Geometry 51
3.4 3D Tele-immersive Streaming Pipeline 53
3.4.1 3D Representation 54
3.4.2 Media Pipeline 54
3.5 3D Reconstruction 54
3.5.1 Capturing setup and calibration 55
3.6 3D Data Compression strategy 57
3.6.1 Qualitative Comparison Existing Mesh Compression Methods for 3D Tele-Immersion 58
3.6.2 Fast Compression Heuristic for Captured Meshes 60
3.6.3 Experimental Results 67
3.7 3D Packetisation 72
3.7.1 Rateless Coding 72
3.7.2 Implementation 73
3.7.3 Experimental Results 77
3.8 3D Tele-immersive Systems Integration and End-to-end Performance 78
3.8.1 3D Triangle Capturing and Rendering 78
3.8.2 Media Pipeline Performance 78
3.9 Conclusion and Discussion 82
Chapter 4 3D Tele-immersion with Connectivity Driven 3D Mesh Compression with Late Differential Quantization 84
4.1 Low Complexity Connectivity Driven Mesh Compression 85

4.1.1 Pattern Based Connectivity Coding....86
4.1.2 Geometry coding with delayed differential encoding....88
4.1.3 Appearance Quantization....90
4.1.4 Evaluation....90
4.1.5 Comparative Results....91
4.2 Updates on Packetisation....96
4.3 Integrated Pipeline Performance....96
4.4 Conclusion and Discussion....102
Chapter 5 Highly Adaptive Geometry Driven 3D Mesh Compression....104
5.1 Objective and Subjective Quality Assessment Methodology....106
5.2 Geometry Driven Mesh Compression....108
5.3 Objective Comparative Evaluation....112
5.4 Subjective Comparative Evaluation....116
5.5 Conclusion and Discussion....118
Chapter 6 Time Varying 3D Point Cloud Compression....120
6.1 Introduction....121
6.1.1 Contributions of this Chapter....124
6.1.2 Related Work....124
6.2 Overview of Point Cloud Codec....125
6.2.1 Requirements and use case....125
6.2.2 Schematic Overview....126
6.3 Intra Frame Coder....129
6.3.1 Bounding Box Normalization and Outlier Filter....129
6.3.2 Octree Subdivision and Occupancy Coding....131
6.3.3 Colour Coding....132

6.4 Inter-frame Coder....133
6.4.1 Predictive Algorithm....134
6.4.2 Rigid Transform Coding....138
6.4.3 Parallelization....139
6.5 Experimental Results....140
6.5.1 Datasets, Experimental setup....140
6.5.2 Objective Quality Evaluation Metric....140
6.5.3 Intra Coding Performance....141
6.5.4 Inter Predictive Coding Performance....145
6.5.5 Subjective Quality Assessment....150
6.5.6 Real-Time Performance and Parallelization....156
6.6 Conclusion and Discussion....157
Chapter 7 3D Tele-immersive Streaming Engine....159
7.1 Introduction....159
7.2 3D Streaming Engine....161
7.2.1 Modular 3D Immersive Communication Framework Architecture....161
7.2.2 Real-Time Streaming Framework....162
7.2.3 Basic Session Management Protocol and Implementation....164
7.2.4 AI and Avatar Commands messaging....166
7.3 Experimental Evaluation....167
7.3.1 Natural User Transmission....167
7.3.2 Frame Skew and Media Synchronization....171
7.3.3 Discussion and Conclusion....172
Chapter 8 Conclusion....175
8.1 Achieved results....175

8.2 Standardisation Contributions 179
8.3 Future development 182
References 184
Appendix A 3D Audio Visual Capture Model 197
Appendix B Standardisation Contributions 200

Chapter 1 Introduction

Humans enjoy communicating and sharing experiences with each other. The Internet is an excellent platform to facilitate this need. The Internet enables distributed shared experiences such as video conferencing, voice calls (possibly in a group), chatting, online gaming and virtual reality. This is changing the way we interact with each other in our daily lives. Current rapid advances in 3D depth acquisition are enabling near-instant highly realistic 3D capture of real humans and objects. The integration of these 3D representations in distributed shared experiences such as virtual reality, social networking, tele-conferencing and gaming could result in improved and novel user experiences. One example of a novel user experience would be the tele-immersive scenario, where a real users can enter a virtual world and interact with a realistic representation of himself. In this example he or she can interact with computer controlled avatars in a virtual world and with other users. Another example of an improved user experience would be interactive free viewpoint rendering of 3D captured content in a social network portal or photo sharing site.

This thesis will look at the implementation of end-to-end multimedia systems based on such highly realistic 3D representations. It will focus on real-time end-to-end conversational style communications. A typical end-to-end media communications pipeline for 3D data capture is show in Figure 1. 3D Media is captured using 3D depth sensing devices such as motion sensors, microphone arrays and 3D cameras such as Microsoft Kinect. A subsequent reconstruction stage can generate complete 3D representations such as a 3D Point Cloud or a 3D Mesh using a 3D reconstruction algorithm. In video conferencing, typically, compressed 2D video is sent over the communication link. For advanced 3D communication systems based on 3D representations meshes and point clouds have to be compressed and transmitted instead. This poses a large challenge in designing such end-end 3D communication systems. While the compression of 3D data such as point clouds and meshes has been considered in the 3D graphics literature [1], real-time compression of time-varying data for virtual reality like applications has rarely been considered. Nevertheless, this is the type of compression that is needed to enable mixing real and virtual reality in real-time distributed systems.

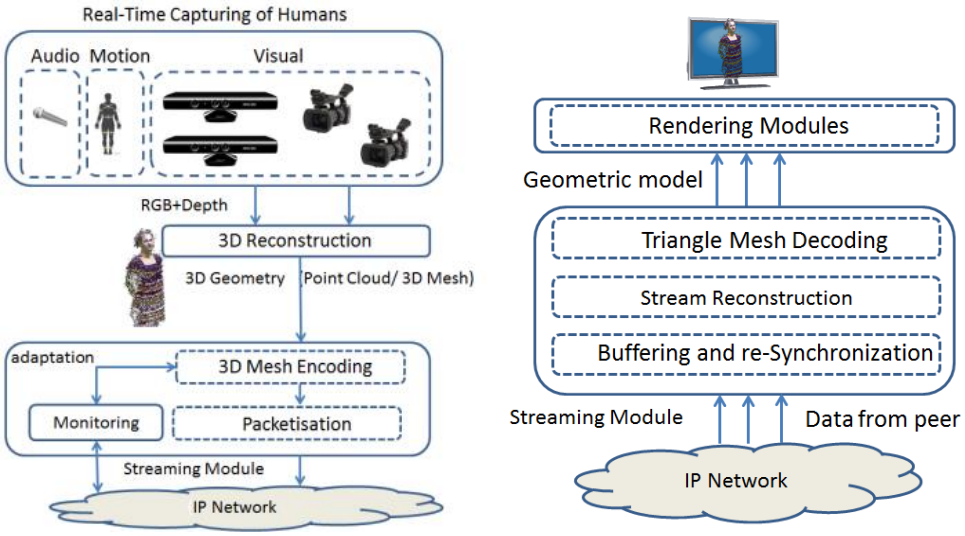


Figure 1 3D Tele-immersive media pipeline

In addition, to support video conferencing style communications technologies for transmission, packetisation, monitoring and media synchronization become important.

The compression is quite challenging due to the unstructured nature of the data and the real-time requirements. The transmission and synchronization are challenging due to the large frame sizes, varying frame rates and frame size variations. This thesis develops three different approaches for the real-time compression of 3D meshes in such a communication scenario. Further, a codec for time varying 3D point clouds is developed, an alternative useful representation for 3D objects. These codecs are integrated in a 3D streaming framework. This framework introduces session management and signalling and a media synchronization strategy suitable for this type of data. The complete framework is then used to experiment in a realistic integrated 3D immersive mixed reality communication system with state of art rendering, 3D data capture and social network services. The research work done in this thesis on end-to-end multimedia systems presents new insights in systems integration, codec design, quality assessment, user experience assessment and media standardisation.

1.1 Motivation

3D Data capture: More 3D devices are becoming available to consumers that include depth sensing and sometimes 3D reconstruction. 3D capture could be based on cameras such as Microsoft Kinect (v1, v2) [2] or based on mobile devices [3] [4]. The combination of these capabilities with computing power enable acquisition of fuller 3D Media representations such as 3D geometry with attributes based on 3D Meshes and/or Point Clouds.

Standardized *Compression Technologies* for media data types are becoming available on the market. Media compression standards often result from joint efforts of industry and academia. Popular standards include MPEG-2 video for digital television, MPEG-2 part 3 for music distribution in the internet (MP3), and the JPEG codec for image compression. For live captured 3D point cloud and mesh data an equivalent widely adopted codec does not exist yet. This motivates us to work on the development of technologies to pursue this ultimate standardisation goal of a 3D Point cloud and mesh compression standard suitable for mass usage in the digital ecosystem.

The *bandwidth and QoS* in access, core and local networks is increasing for mobile and fixed networks. These increased bandwidths will enable fast transmission of 3D data, which can enable distributed shared 3D experiences. In addition, new technologies such as software defined networking can provide per flow bandwidth, loss and jitter constrained communications useful for audio-visual data types. While a lot of work is still to be done in this area, the increasing power of the network and emergence of QoS is a key motivation for our work.

3D Rendering: Real-Time rendering of meshes and point clouds is becoming generically available via API's like Direct 3D, OpenGL and on different software and hardware platforms. The ubiquitous availability of 3D rendering of such 3D primitives makes it easy for consumer to view and interact with this data, even in web based applications.

Social networking : Applications like Facebook, Linkedin and other social network portals make it natural from people from all over the world to interact and share

experiences through video conferencing or photo sharing. Taking this existing interconnectedness of friends, family acquaintances and co-workers into account, a logical next step would be to enable 3D mixed reality in such social networking context. This could be useful in both professional settings (job interview, collaborative work) or in a social setting (social hangout, social gaming). Therefore, the interconnectedness of social networking is a key motivation for our work.

1.2 Challenges

3D Rendering and capturing combined with high bandwidth QoS enabled networking are paving the way for 3D tele-presence in the Internet. Nevertheless, the design, deployment and implementation of such systems still faces significant challenges.

3D Compression: realistic 3D mixed reality uses object based 3D video formats. These are currently not well supported in the popular media codecs such as the ones developed by the Moving Picture Experts Group (MPEG) [5] or Joint Picture Experts Group (JPEG) [6]. Reconstructed 3D Meshes and point clouds often consist of huge amounts of points and triangles consuming large amounts of data. A large part of this thesis will be devoted to efficiently compress these formats. These attempts not only aim at efficient compression of 3D frames, but also at *fast* compression to enable real-time end-to-end interactive communication.

3D Data Transmission: Compressed 3D frames resulting from 3D reconstruction are often large in size (over 1 MB per frame) and lack the data loss resilience such as present in popular image/video codecs. Therefore, a small data loss may result in a complete discard of the frame. In addition, some of our experiments have shown that end-to-end network delay can be problematic when using the TCP protocol to rapidly send large frames. Therefore, for 3D mixed reality communication it is important to study error and delay resilient transmission techniques that can handle large frames in real time. In addition to this, frames may be produced at varying framerates (5-30 fps) and frames may contain different numbers of points. These differences compared to normal video/audio require specific attention to the end-to-end pipeline. The varying frame rates and frame sizes make intra and inter-stream synchronization challenging. In addition, for mixed reality, inter-sender synchronization of frames is important to avoid inconsistence when rendering in the 3D virtual

world. This thesis deals with these different challenges in 3D data transmission and synchronization in the context of a practical system.

Signalling and Session Management: Many different types of 3D data exist such as animations, textures, meshes, point clouds and spatial audio. They need to be supported by appropriate signalling and session management. In addition, the architecture for 3D mixed reality envisioned in this thesis can also support different types of users connected to a social network. Light clients, with rendering only capabilities only up to heavy clients with multi camera reconstruction will be supported. Between users present in the system, appropriate media streams need to be setup for communication. The variety of modules, local configurations and different bit-rate codec settings make signalling and setup of the media sessions a challenging problem. This is not handled by existing media protocols. This thesis develops a signalling platform for highly heterogeneous clients using different types of 3D data formats. While a first attempt, it aims to uncover some of the challenges related to this aspect of 3D immersive communications.

System Integration: 3D mixed reality systems are complex systems. They combine 3D capturing technologies, 3D rendering, networking and Artificial intelligence. This highly interdisciplinary mix of computer software creates problems and challenges of its own. Each of the processes can run in their own threads, possibly consuming a disproportional amount of resources. In addition, they often use different libraries that possibly result in clashes (incompatibility, namespace, Operating systems issues). Last the data flow between modules needs to be carefully considered to avoid problems related to processes running in different threads and information inter dependencies.

Quality Assessment: the inter connect of different capture, rendering, compression and transmission methods for novel data types in novel applications introduces the question how to assess quality? Traditional methodology developed for image and video quality assessment cannot be applied directly to this scenario. Different applications, rendering techniques, data types make quality assessment challenging.

1.3 Large Scale Tele-Immersive Systems Architecture

Figure 2 introduces a large scale systems architecture for 3D Tele-immersive communications in a virtual world. The architecture was developed in the REVERIE FP7 project [7] and has driven both the research questions and many of the practical efforts. The architecture includes a social network that enables social network users to join 3D Tele-immersive sessions. The rest of this architecture features components for 3D Tele-immersive communications divided in 4 groups of functionalities.

The yellow *3D capture components* are used for 3D Tele-immersive data capture and reconstruction. These components deal with extracting meaningful 3D information out of visual and aural data streams. These include 3D visual reconstruction, 3D user analysis for embodying an avatar and 3D Audio capturing.

The blue *Artificial Intelligence components* deal with artificial intelligence techniques for perception, cognition and animation. They provide higher level forms of intelligence to bridge the semantic gap between the naturalistic (camera captures) and synthetic (computer animated) objects. Desired functionalities include emotion understanding and conversational capabilities of avatars.

The green components deal with networking, streaming and compression functions. As the configuration presented in this system includes 3D reconstruction, capturing, 3D Rendering, AI and social network functions, specific network support is required. Therefore, the presented architecture presents a fertile ground for research and experimentation with network protocols to support the requirements of this application. This thesis will develop the appropriate network and codec support for this architecture to tackle research challenges arising from this configuration.

The pink *rendering components* are components for 3D rendering related functions. They include spatial audio composition (mixing of multiple audio streams for efficient delivery to end users). Scene structuring and navigation (handling the composition of all objects in the 3D Room and mapping them to a single space). At the client sides, different renderers are implemented for different types of 3D data. In this architecture these renderers can operate in parallel and their output are composited, rendering multiple objects in a single scene.

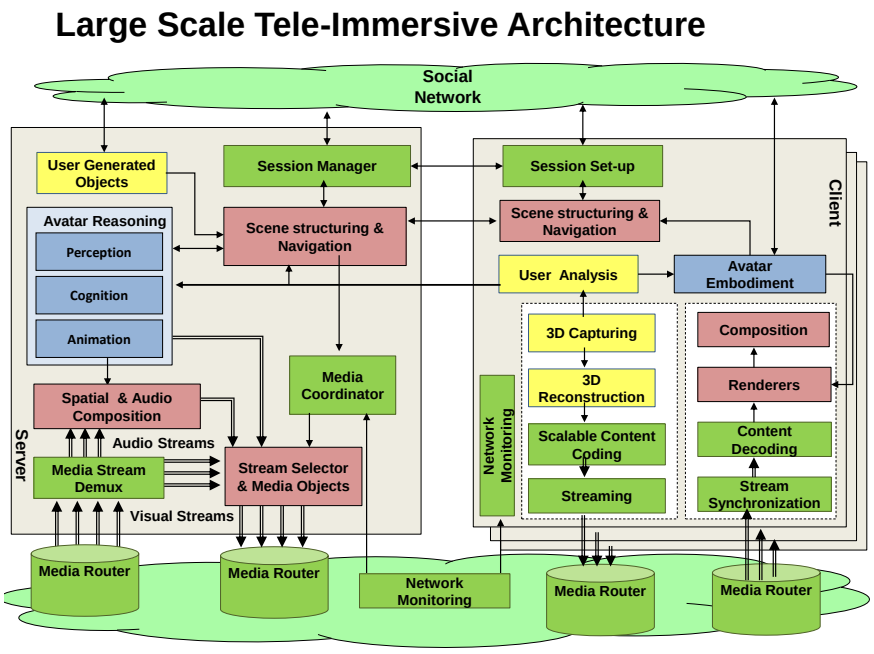


Figure 2 Large Scale Tele-immersive System Architecture

1.4 Research Questions

The architecture as shown in Figure 2 will enable 3D immersive mixed reality. In such a scenario naturalistic and synthetic contents will be mixed in a virtual world, resulting in a novel type of distributed shared experience beyond gaming and video conferencing. To support such applications from the network side, the main research question addressed in this thesis is the following:

Main Research Question: How can we support Real-Time (<300ms) end-to-end media streaming for Mixed Reality and 3D Tele-immersion be supported in the current Internet with state of art compression rates ?

This thesis addresses this question using an experimental integrated multimedia systems approach. It will build and test prototype systems that include components from capture up to rendering, thus integrating components for networked transmission and

3D compression in complete systems. The advantage of such a system centric approach is that it can shed light on integration issues and optimization strategies between components that have been previously overlooked. In addition, the prototypes can be tested with real users, resulting in explorative human centric studies on the benefits of different technical implementations. In addition, the systems and human centric approach can lead to preliminary requirements of importance for the development of the technology in academia and industry. To this aim the work in this thesis has contributed to updates in the requirements of media technology standards in MPEG [8] and JPEG PLENO [9]. The main research question is split in 5 questions that are addressed throughout this thesis.

Research Question 1: *What is the state of the art to support tele-immersive mixed reality, and how does it relate to other 3D media applications?*

The research question aims to explore related work to find out what kind of mixed reality and 3D communications systems have been developed in the past and which technologies are available for the next generation of systems. It reveals that previous work does not target the architecture that this thesis envisions and that this architecture poses many challenges. Further, we present a generic pipeline for development and study of multimedia systems and standardisation of related data types (compression formats, raw digital capture formats and network ready bit streams). This pipeline forms a reference model for standardisation of different audio visual datatypes in MPEG, and was partially based on a contribution of the author during this thesis work (see section 0).

Research Question 2: How can we achieve Real-time (below 300 milliseconds on commodity hardware) compression and transmission of highly realistic reconstructed 3D humans based on meshes with State of Art Compression rates (i.e. MPEG-4)?

Our second research question deals with the real-time streaming of highly detailed reconstructed 3D humans from multiple range sensors as 3D meshes. This is highly challenging due to the number of points (over 300,000) in the mesh and the rate of capturing (in the order of 10 to 12 frames per second) and the real-time transmission requirements. A closer look at the state of the art in mesh codecs shows that their

complexity increases linearly with the number of vertices and that especially real-time encoding is challenging and ill-considered in previous work. In addition, compressed 3D data is prone to data losses. Therefore, the aim of this research question is also to investigate a reliable transmission protocol that also works in real-time.

Research Question 3: How can we achieve highly adaptive (lossy) real-time compression and transmission of highly realistic 3D replicants based on meshes that can be used for many different bit-rates and level of detail (i.e. a geometry coding with a large range of bit-rates)?

To support real-time conferencing in the real Internet where bandwidth fluctuations happen, a codec that can support many different bit rates is beneficial. In addition, as known from computer graphics, 3D objects can often be rendered at lower levels of detail. This is especially the case if objects are further away in the scene. Based on these motivations, this thesis explores the design of a mesh codec that runs in real time and is highly adaptive. This codec represents a lossy design, i.e. not all input vertices/triangles are present in the decoded version. Such a codec results in a lower bit rate and a lower quality decoded output. This is useful for multi user/ multi-site teleconferencing using 3D mesh data and for less relevant 3D reconstructions that can be rendered at a lower quality.

Research Question 4: How do we design improved Real-Time Compression of 3D Time varying Point Clouds with improved lossy colour coding and inter-prediction with a negligible perceptual distortion compared to the current state of the art in practical real-time point cloud compression?

Point clouds are simpler to acquire than meshes and introduce less overhead as they do not store the connectivity information. While real-time point cloud compression has been studied in the past, some aspects relevant to mixed reality systems have been poorly addressed. Therefore this thesis investigates three aspects of real-time point cloud compression. First it studies real-time and lossy coding of colour attributes. Second, it will explore lossy inter predictive coding of 3D tele-immersive point clouds. Last, it will investigate quality assessment methodology based on an objective metric and subjective evaluation in a mixed reality system. The thesis combines the research results to study the complete point cloud compression framework for

mixed reality and tele-immersion and assess the performance both objectively and subjectively.

Research Question 5: how can we design a 3D media streaming engine for heterogeneous 3D tele-immersive communications that is compatible with the current internet infrastructure (i.e. existing transport and signalling protocols)*?*

This thesis will investigate the support of 3D Media streaming in mixed reality with various 3D capture and render modules. It will investigate the type of transmission schemes needed such as peer-to-peer, publish-subscribe, application layer multicast. It will develop a unified API for the external 3D modules to use the network streaming facilities. Subsequently, it investigates session management for presence and stream setup. Lastly, support for media synchronization is added and the integration into the mixed reality system is completed. The thesis will evaluate the streaming system in a realistic mixed reality system with multiple sites.

1.5 Contributions

This thesis presents the following contributions to the field of multimedia systems:

It presents work on a ***novel architecture and framework for 3D Tele-immersive mixed reality*** for use in the Internet. This framework includes advanced modules for 3D capturing, rendering and Artificial intelligence in the context of an immersive virtual room linked to the social network. This has not been considered in previous works. This is of particular relevance for enhancement of social networks for shared distributed experiences. These insights will benefit large technology providers interested in providing such services such as Facebook (Oculus), Microsoft Skype, Linkedin, Google Hangout, World of Warcraft etc.

It presents ***novel approaches for Geometry Coding suitable for 3D Tele-immersive mixed Reality***. Contrary to previous attempts in 3D graphics literature, it develops encoder/decoder frameworks that work in real time on commodity hardware. This is critical for enabling real-time end-to-end communications for 3D tele-immersive Mixed Reality. It starts with approaches that make use of the capturing algorithm/procedure that can benefit the coder, optimizing the end-to-end pipelines. Later, it develops complete and generic highly adaptive real-time mesh and point

cloud codecs. The generic geometry driven mesh codec has been integrated in the tele-immersive framework for multi-site streaming. Further, this thesis introduces a time varying point cloud codec implementation that follows a hybrid architecture and includes techniques for inter-frame coding and lossy colour coding. The point cloud codec implementation has been contributed to MPEG and currently serves as a base implementation for development of a standard for point cloud compression in MPEG.

It presents ***Approaches for Network Streaming and Signalling for 3D tele-immersive and mixed reality***. It proposes a signalling and session framework based on XMPP protocol to setup sessions between heterogeneous clients as a control plane. For 3D geometric data transmission experimented with the TCP and UDP protocols using LT codes are performed. Further, the end-to-end multithreaded pipeline for 3D geometry transmission is enhanced using a last in first out policy (LIFO) to reduce end-to-end delay. The overall streaming framework includes a generic API for stream creation and a virtual clock system that can enable media synchronization (both inter-stream and inter-sender). Last, the framework includes a fast messaging system for publish and subscribe based transport. This is useful for command and other animation messages. This framework provides core network support for a larger 3D Tele-immersive mixed reality system developed in the context of the EU project Reverie [10] and provides hints on how such network support can be provided in future tele-immersive systems.

It presents **Systems integration into large scale prototypes and preliminary user centric evaluations.** All components have been integrated in a larger 3D Tele-immersive mixed reality framework. The integration of different components for 3D capture, data transmission and rendering provides some insights of potential bottleneck issues and many practical issues. The full systems integration has enabled user testing in real world scenarios. These tests all hint towards the benefits of highly realistic 3D representations and 3D tele-immersive communication in collaborative tasks and social interaction tasks.

It contributed to the **development of requirements, benchmarking tools and codec platform implementation for future video and image coding standards.** The systems integration and codec development presented in this thesis unlocked novel

requirements for future image and video coding. Over the course of 3 years, it has contributed to the international standards for media technology (MPEG) [8]. This has resulted in updates to support time varying meshes and point clouds in the MPEG requirements [11]. In addition documents defining the requirements of 3D tele-immersive coding have been ratified in MPEG [12]. In addition, the methodology (including the quality metrics and benchmarking evaluation platform developed in this work) for point cloud compression have all been adopted by MPEG in the activity on point cloud compression [13]. The Point cloud compression and evaluation platform is used for the further development of the Point Cloud Compression in MPEG in the coming years, serving as a base exploration software. Last, during this work, the author has contributed to the draft version of the JPEG PLENO [9] [14] requirements, an emerging 3D Visual standard that will target compression of naturalistic point clouds, light fields and holographic images. These contributions focussed on the use cases in Mixed Reality 3D Tele-immersive systems area.

1.6 Thesis Outline

Chapter 2 presents related work and technical background based on the current advances in the Internet, 3D capture, rendering and social networks. Further it presents a generic 3D audio visual capture reference model for multimedia systems and data format standardisation. Last, an overview of previous work on 3D tele-immersive systems is presented. This chapter answers research sub question 1.

Chapter 3 presents a system centric optimized real-time streaming component for reconstructed 3D mesh data (capturing based on [15]). It includes a compression and transmission component. It exploits both regularities in the connectivity and in the geometric blocks and is computationally simple and easy to parallelize. In addition, the prototype includes a novel UDP based network transmission scheme based on a linear rateless code. It compares the end-to-end performance of the prototype with the MPEG-4 mesh codec and transmission based on TCP. The prototype reports reduced end-to-end delay resulting in real-time performance. A major conclusion of this work is that traditional schemes for mesh coding and transmission are not designed with the requirements of 3D tele-immersion in mind. This chapter further partially answers sub research question 2. The work in this chapter is based on the following two publications.

Mekuria, R.N. *Sanna, M. Asioli,S. Izquierdo, E. Bulterman, D.C.A. and Cesar, P. A 3D Tele-Immersion System Based on Live Captured Mesh Geometry. Proceedings of the 4th ACM Conference on Multimedia Systems (MMSys 2013), Oslo, Norway, February 27- march 1 2013 (focussing on the network-compression pipeline integration) (Best Paper Award in special session on 3D in Multimedia)*

Mekuria, R.N. *Alexiadis, D. Daras, P. and Cesar, P. "Real-time Encoding of Live Reconstructed Mesh Sequences for 3D Tele-immersion." Proceedings of the International Conference on Multimedia and Expo. International Workshop on Hot Topics in 3D (Hot3D) San Jose, CA, July 19 2013 (focussing on the capture-compression integration)*

Chapter 4 presents a codec based on connectivity driven mesh coding and an optimized streaming pipeline that integrates improved rendering. The codec bears similarity to the codec defined in the MPEG standard for mesh compression [16] as it uses the connectivity to efficiently code the geometric positions. By exploiting regularities in the connectivity using differential and entropy coding based on the deflate/inflate algorithm and repetitive patterns, the connectivity coder is made highly efficient. The solution is fast and more generically applicable compared to the codec developed in Chapter 3. Subsequent to the connectivity coding, late (staged) quantization of differentials between connected vertices to code the geometry is performed. This allows the codec to skip explicit variable length entropy encoding. Next, the streaming pipeline is optimized using a last in first out approach (LIFO) for inter thread exchange of frames. The overall system is tested in scenarios representing typical moderately managed networks such a VPN Network. The overall optimization in coding and transmission enables real-time end-to-end communication of the highly reconstructed 3D human combined with highly realistic rendering using global illumination. This chapter answers research question 2 in a more generic way. The chapter is based on the following publications.

Mekuria R.N., *Cesar P., Bulterman D.C.A. "Low Complexity Connectivity Driven Dynamic Geometry Compression for 3D Tele-Immersion" in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, (codec) (IEEE ICASSP 2014), Florence, Italy, May 4-9, 2014 (codec implementation)*

Mekuria R.N., *Sanna M., Izquierdo E., Bulterman D.C.A., Cesar P. "Enabling 3D Tele-Immersion with Live Reconstructed Mesh Geometry with Fast Mesh Compression and Linear Rate less Coding" IEEE Transactions on Multimedia 2014 Volume 16 issue 7 pp. 1809 – 1820 (codec, integration, pipeline optimization)*

Chapter 5 presents a highly adaptive geometry driven real-time mesh codec for use in 3D tele-immersive applications. Contrary to the codecs developed in chapter 3 and 4 and the codec in MPEG-4 [16], this codec enables fine grained bit-rate and quality control. It enables elimination of vertices (mesh simplification), which for 3D geometric data is a better way to control quality and bit-rate compared to vertex quantization (this is performed in MPEG codecs such as MPEG-4 TFAN). The mesh geometry is organized and simplified in an octree voxel grid. Subsequently the simplified connectivity is computed based on the octree cells and the original connectivity. Then, the new connectivity representation is converted to a novel representation consisting of the first index, and two vector integer offsets in the octree grid. Subsequently this chapter develops an approach for vector quantization to store the two offsets. The first index is coded in a run length fashion, the second two based on this vector quantization scheme. This approach achieves a compact representation of the connectivity, exploiting the grid organization of the octree. The octree structure is then compressed using serialization and a range coder. The key benefit of this codec is that it can be used for multi-site streaming and rendering less relevant objects such as those at distance at a lower level of detail. This chapter addresses research question 3 and was published in the following works.

Mekuria R.N. *Cesar P. "A Basic Geometry Driven Mesh Coding Scheme with Surface Simplification for 3DTI" IEEE Communication Society: Multimedia Communications Technical Committee E-letter, May 2014 (Codec implementation)*

Mekuria R.N. *, Cesar P, Doumanis I, and Frisiello A., "Objective and Subjective Quality Assessment of Geometry Compression of Reconstructed 3D Humans in a 3D virtual room," in Proceedings of the Applications of Digital Image Processing XXXVIII Conference, (part of the SPIE Optical Engineering + Applications, part of SPIE Optics + Photonics symposium) , San Diego, USA, August 9-13, 2015 (subjective evaluation, system integration)*

Chapter 6 presents a novel design for real-time point cloud compression for 3D tele immersive video. This chapter deals with some specific challenges for compression of point clouds for 3D tele immersive video. It starts from the classical octree based approach for point cloud compression. Then, it introduce schemes for improved lossy real time colours coding by mapping to a JPEG image grid. Subsequently, it introduces a temporal inter-prediction scheme between frames by computing motion vector translations between occupied macro voxels (blocks of octree leafs). It efficiently stores 3D rotation as a quaternion, that is compressed using a known quaternion compression scheme and quantizes the translations using 16 bits. Points that could not be predicted are coded in intra fashion (using an octree serialization and compression). This is followed by objective rate distortion evaluation followed by a subjective user study with 20 users in a realistic mixed reality system. In addition a parallelized implementation using OpenMP is provided. The results show that the inter-prediction can save up to 30% in bit rate, while the colours coding scheme results in highly reduced bit-rates (up to 90 %), without being noticeable by users. The chapter is based on the following publications.

Mekuria R.N. *Blom K., Cesar P. "Design, Implementation and Evaluation of a Point Cloud Codec for 3D Tele-immersive Video" IEEE Transactions on Circuits and Systems for Video Technology (accepted for publication, to appear sept. 2016)*

Mekuria R.N *"Point Cloud Compression" proceedings of inaugural workshop on JPEG PLENO Workshop in conjunction with MPEG/JPEG meeting in Warsaw, June 23, 2015*

Mekuria R.N. *Cesar P. MP3DG-PCC, Open Source Software Framework for Implementation and Evaluation of Point Cloud Compression.in Proceedings of the 2016 ACM on Multimedia Conference (MM '16). ACM,*

In addition, the implementation has been accepted as a first implementation for point cloud compression MPEG-4 part 16 (AFX) in the MPEG standardisation software repository [13]:

Chapter 7 presents a content-aware 3D streaming engine and the integration in the overall framework. It integrates the components developed in previous chapters, Further, the system provides the API to the overall mixed reality system for multi-site and multi-stream communication. This component supports creation of different types of media streams (visual, audio etc.), lightweight messaging (publish and subscribe) and control messaging. It uses an XMPP server for presence and stream setup, using an extension of the XMPP XEP-30 disco protocol. This presence protocol aims to demonstrate how, based on available modules, social 3D tele presence in a virtual world can be supported. The framework provides a synchronization mechanism based on a virtual clock. Synchronization schemes based on buffering have been implemented in the rendering modules. The streaming framework has been evaluated for multi-site streaming in the larger Reverie tele-immersive system. The synchronization subsystem achieves approximate inter-stream and inter-sender synchronization of 3D audio and 3D geometry streams coming from different sites. The work has been presented in the following scientific publication.

Mekuria R.N. *Frisiello A., Pasin., M, Cesar P.S. "Network Support for Social 3-D Immersive Tele-Presence with Highly Realistic Natural and Synthetic Avatar Users" in ACM workshop on Massive Multi User Virtual Environments MMVE'15, Portland USA, in conjunction with ACM MMSys'15*

Chapter 2 Related Work

This chapter presents the related work. First, it provides an overview of the technical background in 3D capturing, rendering and networking. Then it will provide some related work on 3D Tele-immersive systems development. This chapter addresses research question 1: *What is the state of the art to support tele-immersive mixed reality, and how does it relate to other 3D media applications?*

2.1 Technological Background and Motivation

2.1.1 3D Scene Capture

3D cameras detect not only pixel colours, but also the depth information of the scene. The depth information is important as it can be used for rendering arbitrary viewpoints via depth image based rendering (DIBR) which, amongst others can be used for stereoscopic rendering. For mixed reality and 3D tele-immersive communication the depth information can also be used to segment 3D objects of interest. Examples of such objects of interest are the human or participant or buildings or landscapes, or other objects such as an automotive vehicle. This segmentation often results in 3D Meshes or Point Clouds which are vertex positions and triangles in the 3D space. The segmentation can work especially well when multiple depth cameras are recording from different angles and are calibrated (both internally: depth and colour and externally between cameras). This section briefly summarizes technologies for sensing depth data, followed by camera configurations for 3D reconstruction. The chapter also summarizes an example 5 camera setup to reconstruct 3D Meshes and Point Clouds based on [15].

Depth sensing from single passive camera: 3D shape can be obtained from sequences of single camera video data. These techniques are often described as *motion from X techniques* that use shading, texture, focus or motion (for X) to obtain the 3D structure of the scene. Even though there is no complete and fully reliable solution to obtain 3D structure from a single camera stream, techniques based on structure from motion tend to be the most promising [17] in this class of techniques.

Depth sensing with multiple passive cameras often implies the usage of two (or more) configured cameras and solving some under constrained correspondence problem. A common method is *stereo matching*, that bears similarity with how the human visual system senses depth with two eyes based on binocular cues. Based on the two input images and known camera parameters and positions, disparity can be calculated between corresponding points in both scenes. This disparity information can then be used to estimate the depth at the corresponding points in the scene. However finding such matching points with a computer does not always work well. In case of occlusions and repeating textures the problem of finding corresponding points in the scene becomes ambiguous [18] and depth estimations can fail.

Depth sensing with active cameras is a more direct and straight forward way of acquiring depth information from a scene without additional computational efforts. Microsoft introduced the popular Kinect 1 sensor which is based on *structured light*. It is based on transmitting an infrared dot pattern with an infrared projector, and receiving it with an infrared Receiver. Triangulation of the received and original patterns can be used to compute the depth information. The principle is shown in Figure 3 where Zo is the distance to the object, Zr the distance to the reference plane (i.e. maximum depth), the distance between the IR receiver and projector and d the disparity between the two triangulated patterns and f the focal length. Depth can then be calculated based on formula (2.1).

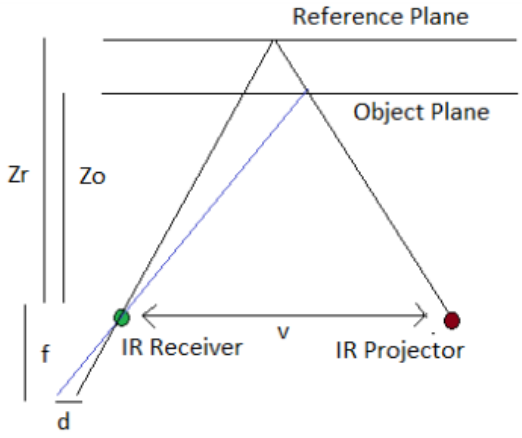


Figure 3 Triangulation of infrared pattern based on structured light [18]

$$Zo = \frac{Zr}{1+\frac{Zr}{fv}d} \qquad (2.1.)$$

Alternatively, the Microsoft Kinect for Windows v2 is based on Time of Flight (ToF) principle, which is also used in LiDaR (Laser Imaging Detection and ranging using lasers instead of infrared light). The ToF depth sensing principle is based on measuring the distance the light has travelled. This is done by modulating the light and measuring the phase shift introduced at the receiver compared to the original signal. If the light is modulated at frequency f_{mod} the depth of the object Zo can be computed by formula 2.2. In this equation c is the speed of light, f_{mod} the modulation frequency and φ_d the measured phase shift in radians. LiDars use lasers instead of light at wavelengths near the infrared spectrum and have much larger range and resolution. These active techniques provide easier and more reliable ways to sense 3D depth information with less computational overhead compared to passive methods.

$$Zo = \frac{c}{4f_{mod}}\frac{\varphi_d}{2\pi} \qquad (2.2)$$

Figure 4 Microsoft Kinect, v1 (left, based on structured light) and v2 (right, based on Time of Flight ToF) and SoftKinetic (ToF)

An alternative and generic way to capture a naturalistic 3D scene is based on the plenoptic representation. Plenoptic cameras try to capture the light fields produced by the 3D scene by capturing for each (x,y,z) the luminance radiated in different directions (θ, φ) over any range of wavelengths λ at every time t. This results in a 7 dimensional function for luminance (eq. 2.3.). Sparse Capturing of these light fields is usually done with many cameras/pinholes that are spatially co-located. Storing this type of data is highly challenging as the 7 dimensions introduce large amounts of data.

$$l(V_x, V_y, V_z, \theta, \varphi, \lambda, t) \qquad (2.3)$$

For more information on handling this type of data we refer to [19]. The rest of the thesis will look at object segmentation from colour plus depth data, instead of this type of plenoptic data. Segmented objects and a synthetic scene will allow us to implicitly and artificially generate approximate instances of $l(V_x, V_y, V_z, \theta, \varphi, \lambda, t)$ by applying rendering techniques from computer graphics. This enables using the more sparse geometry representation instead of a heavy plenoptic representation. In addition, it introduces higher flexibility for our mixed reality system and enables composite rendering with synthetic 3D graphics objects more easily.

To segment 3D objects from colour data plus depth data (either from passive or active depth sensing), multi camera setups combined with efficiently implemented 3D reconstruction algorithms can be used. Reconstruction algorithms generally remove background and other irrelevant data and stitch the point clouds from each view together and compute a single 3D Mesh or Point Cloud. Several attempts have been made with Microsoft Kinect technology [15], [20], [21] and in the near future further advances are expected. In the figures below we show some of the full 3D meshes obtained in real-time from multiple calibrated Microsoft Kinect v1 of time varying human subjects. On the left side techniques have been based on Zippering based on [15] while in right side we show the results obtained with Poisson 3D reconstruction based on [22]. The observed artefacts are due to infrared interferences between Kinect and stitching errors. With higher resolution sensors like Microsoft Kinect v2 and SoftKinetic it is expected that better 3D reconstructions will be possible. However, experimental rendering of these meshes in real time has shown that they give a photorealistic impression despite these artefacts. In addition, technologies for 3D point cloud capture are now also becoming available on mobile handheld devices [3] [4].

Figure 5 3D Reconstructions from multiple Microsoft Kinect v1 in MeshLab (zippering based on [15])

Figure 6 Reconstructions from Multiple Kinect v1 in Meshlab (Poisson based on [22])

Figure 7 Multi Kinect Camera Setup deployed in practice

Motivation 1: 3D object scanning is becoming increasingly available for multimedia applications

2.1.2 3D Rendering

Contrary to plenoptic data such as captured light fields, segmented 3D Meshes or Point clouds enable 3D rendering based on techniques supported in common computer graphics API's. Examples are OpenGL [23] and Direct3D [24] which both enable hardware independent implementation of rendering pipelines in computer software. This can be used to compute 2D images of 3D Scenes. The rendering pipeline can compute the lighting and motion blur effects based on synthetic lighting conditions. The representation of naturalistic content such as reconstructed 3D meshes and point clouds therefore opens up the possibly of composite rendering of both naturalistic 3D and synthetic 3D contents in a virtual world.

3D Rendering is the process of creating the 2D images of a 3D scene. It can be compared to taking a photo or filming a scene in real life from synthetic object. Various techniques have been developed such as ray tracing, scan line rendering and z-buffering to perform this process in real time. Most algorithms have been implemented in hardware on graphics cards and general purpose graphics processing units (GPGPU's). This facilitates rendering of a 3D scene based on input primitives such as vertices (positions + colours), triangles and material properties for reflectivity/transmissivity (i.e. how light scatters at a given point in the scene) and shading (how materials vary across the surface). The graphics pipeline generally breaks the input geometry down to many smaller graphics primitives which can be lines, points or triangles. These smaller graphics primitives can then be used to calculate the 2D image from the 3D scene in the part of the pipeline that is often referred as rasterization. Rasterization computes the contribution of each primitive to each 2D Pixel in the image. In addition, when rendering, the lighting conditions often combine synthetic lighting (light transmitted from certain sources) with ambient lighting (universal lighting of the objects to show the scenes). In most 3D graphics systems the lighting conditions can be set arbitrarily.

To further illustrate the typical 3D rendering pipeline and how it can be used to render 3D scenes, we illustrate the OpenGL 3D Graphics pipeline in Figure 8 [23]. OpenGL is one of the most often used APIs for 3D Rendering and well supported in hardware and software systems. The OpenGL pipeline contains fixed stages (rounded rectangles in Figure 8) and programmable stages (sharp rectangles in Figure 8) also called *shaders*. The first and only mandatory programmable stage is the *vertex shader*. The job of the vertex shader is to take the input vertices and transform them for further processing. This often implies a colour modification based on per vertex lighting computation and a conversion to the world space coordinates (from the model space coordinates). Other operations are also possible as long as a 1:1 vertex transformation is applied. The *tessellation stage* is optional and is controlled by two programmable shaders called the *tessellation control shader* and *the tessellation evaluation shader.* The goal of the tessellation is to break up larger primitives (also called patches) of lines/points or triangles into many smaller primitives. For example, a triangle can be divided into many smaller triangles and a line in many points or line segments. These smaller primitives make the subsequent processing

stages easier (i.e. geometry shading, rasterization). The key goal of the *tessellation control shader* is to define the right number of control points to perform the tessellation (i.e. subdivision into smaller triangles), while the evaluation shader runs for each produced vertex an evaluation routine to check the result of the tessellation. The subsequent (optional) geometry shader can be programmed to generate new 3D primitives from the primitives on data resulting from the vertex shader (or tessellation when this was enabled). A key feature of the geometry shader is that it can change the number of output primitives and the types of output primitives. Again this can be beneficial to later stages (rasterization), where the contribution of each 3D primitive to each 2D point needs to be calculated. Alternatively, it may provide higher quality approximations of primitives.

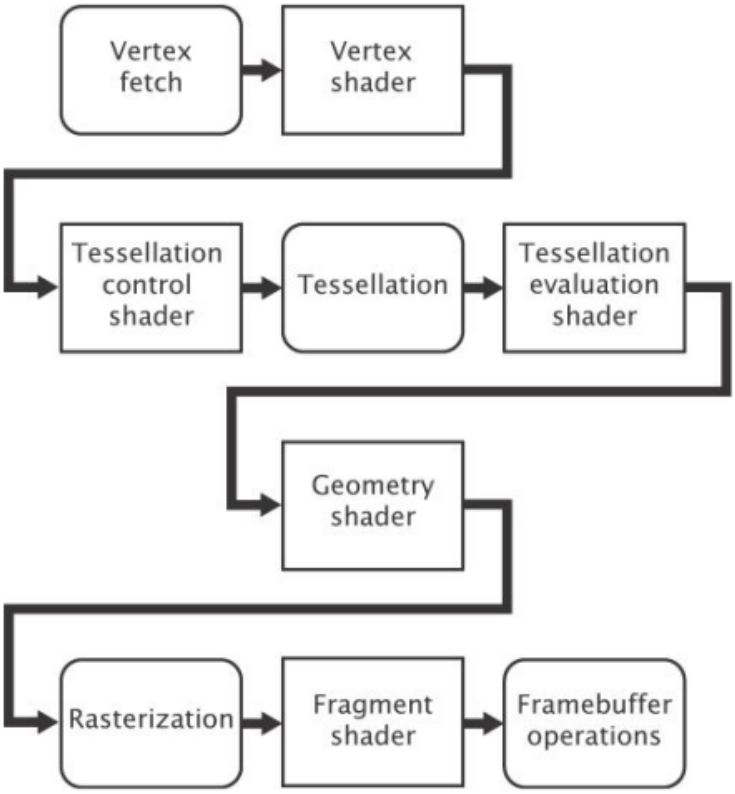


Figure 8 OpenGL pipeline for 3D rendering, sharp rectangles are programmable stages, rounded rectangles are fixed stages [23]

Figure 9 shows a 3D Avatar designed in the REVERIE project in a 3D virtual room. In addition, the API's are supported in the web by WebGL [25] and in higher level programming languages (JavaGL [26], PyOpenGL [27]).

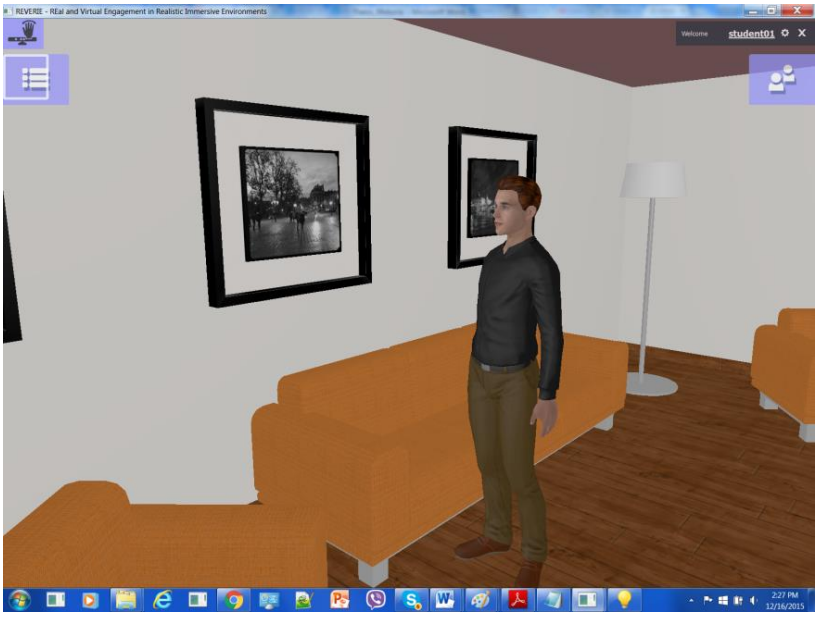


Figure 9 Avatar Rendered compositely in 3D World in fixed 3D rendering setup

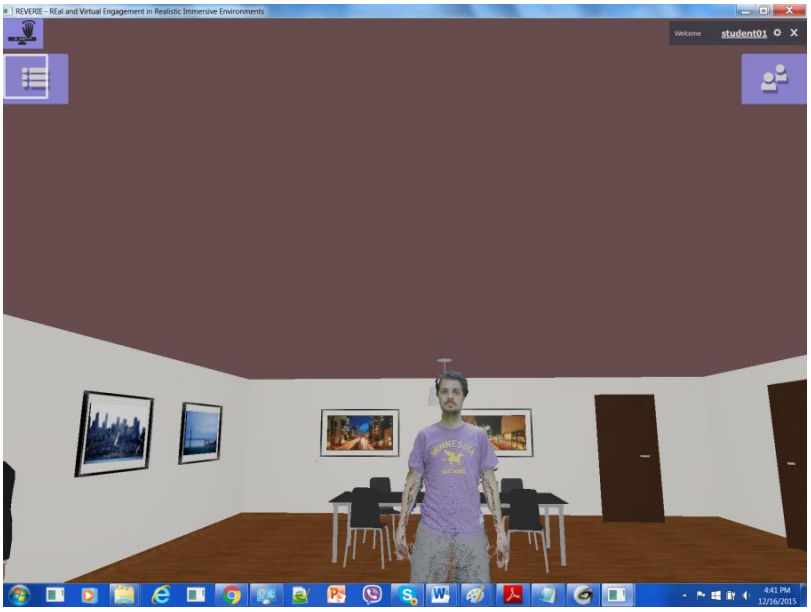


Figure 10 3D Point Cloud reconstruction rendered in 3D rendering setup

Background Motivation 2: 3D Rendering is increasingly available on any platform and can be deployed in real-time tele-immersive systems

2.1.3 3D Media Internet

The Internet has evolved massively in the last decades and is getting better at supporting 3D Media communications for distributed shared experiences. From a high level point of view, three areas of technological Internet innovation are important when considering 3D media in distributed shared experiences.

The first one is innovation in physical link technologies for access, core and local area networking. Advances include novel modulation and antenna/transmitter techniques (physical layer), error correction schemes (data link layer) and medium access control and spectrum sharing techniques (medium access control layer). These are supported by advances in electronics and enable more data to be send over existing cable infrastructures such as coax, twisted pair, fiber and the wireless spectrum. This

paves the way for higher data rates as required for realistic 3D media communications.

The second area of innovation is in the overall architecture of the Internet and support for Quality of Service (QoS). The Internet connects many different types of networks and enables robust and unified data delivery services based on an architecture of packet switching and forwarding. Resilience to component failures and topology changes and the unified data delivery support are both large achievements of the current Internet architecture. This is excellent for static data/information exchanges that the Internet was designed for. However, in case of real-time media services forwarding and switching in the Internet are often problematic. Delay, jitter and data losses caused by congestion control and queuing delays in the packet switching often degrade user quality experienced in multimedia services. Therefore Quality of Service (QoS) architectures that support delivery with hard or soft constraints on delay, jitter and data losses in the Internet have been developed to better support real-time and interactive media services. QoS will also be important to achieve 3D Tele-immersive and mixed reality systems in the Internet due to its high bandwidth and low end-to-end delay requirements.

The third area of innovation is in the design of media specific protocols. The main contribution of these application layer protocols is the setup and signalling of media session including bit-rate selection, codec configuration connection and session maintenance and tear down. In addition, quality adaptation during the media session based on network conditions is sometimes applied. Examples of such media specific protocols are Adaptive Streaming over HTTP (MPEG DASH) [28] and WebRTC [29]. These protocols play a large role in the delivery of interactive media sessions and interoperability between different devices, services and deployments.

We summarize each of these 3 layers in Table 1 and briefly describe the advances in the state of the art in the following sections. This is a brief overview of the technologies enabling the 3D Media Internet.

Table 1 Important Innovation areas supporting 3D Media Sessions in the internet

Internet Media Service	Media Specific Protocols (MPEG DASH [28], WebRTC [29], RTP/RTCP [30])
	QoS Architectures (DiffServ [31], IntServ [32])
	Physical Layer Technologies (DSL, Wifi, DocSis, IEEE 802.11xx, IEEE 802.3xx)

2.1.3.1 Physical Network Media

Many different types of wired and wireless physical media link technologies are used in IP based networks. Common wired links are fiber optic cables, twisted pair cables and coax cables. Wireless links often use parts of the radio and/or microwave spectrum. For both wired and wireless links the physical properties of the medium and the spectral bandwidth available for transmission impose a limit on the amount of information that can be sent. These limitations are based on the laws of physics and information theory. However, in practice, the electronics used (for transmission, reception and amplification modulation etc.) are often bottleneck. With advances in digital signal processing and electronics, more sophisticated modulation techniques and noise filtering techniques can be readily implemented. Therefore, for each physical medium/link type, improved transmission protocols can be implemented with improved modulation, error coding and media access control. These advances together with improved quality physical media are resulting in higher bit-rates for sending data over links. Different transmission protocols can be developed for specific settings (military, space exploration). However, standardized protocols developed by organizations such as IEEE and 3GPPP and other standardisation organizations are most deployed in practice.

Figure 11 demonstrates the evolution of supported bit rates in common wired transmission links. Figure 12 shows the same for wireless networks based on the mobile telephone system (licensed spectrum). In Figure 13 we show the evolution for wireless technologies in the ISM Bands (Industrial Scientific and Medical Bands), these frequencies that don't need licensing are often used in private wireless networks. For each of the technologies bit-rate is increasing over the years. The increased bit rate

supported by access technology often enables faster end-to-end connections in the Internet as access is sometimes bottleneck for the entire end to end connection.

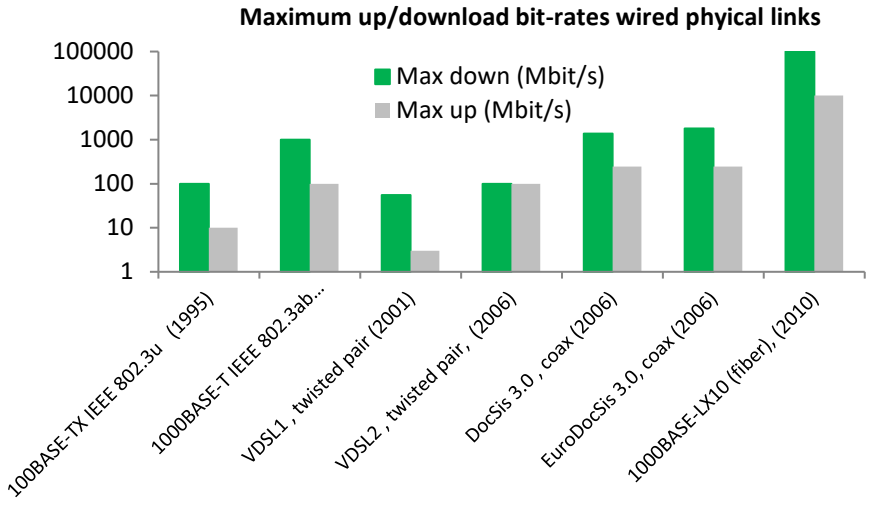


Figure 11 Maximum up/down load bit-rates in wired physical links used IP Networks

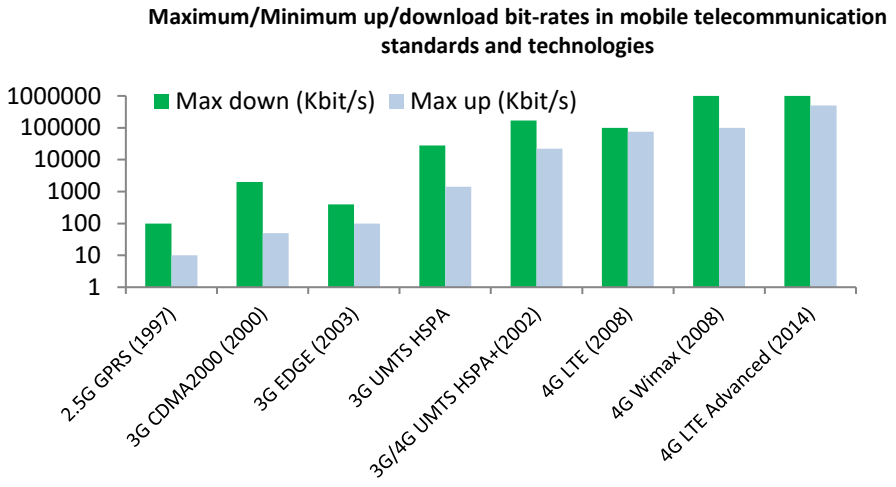


Figure 12 Maximum up and download bit-rates in wireless physical links in the mobile phone system

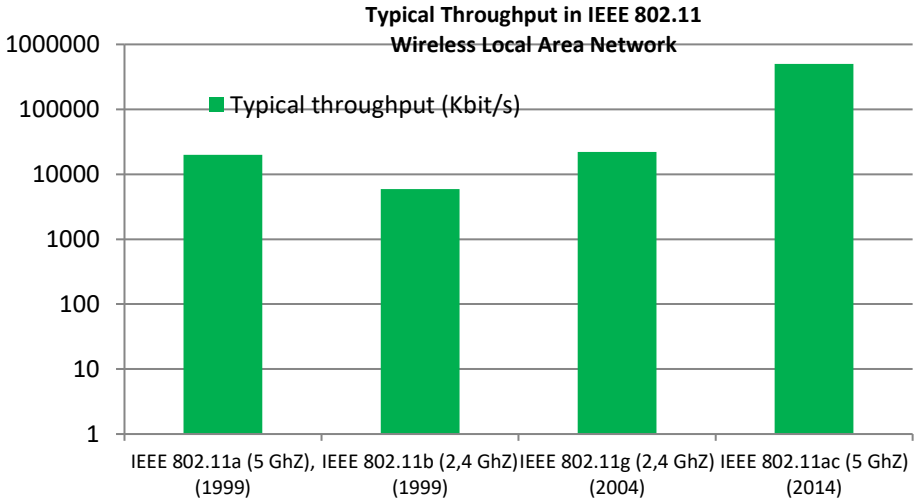


Figure 13 Typical throughput in wireless links in the ISM band often used for Wireless LAN

2.1.3.2 QoS Architectures

The original design of the Internet is based on the requirement of resilience and a unified model for data transmission. This is achieved in an architecture of store and forward packet switching. Physical links are inter connected via elements called routers that both forward small units of data (i.e. packets) towards their next destination (**forwarding**) and determine the best end-to-end path (or at least the next best outgoing link) (**routing**). Figure 14 shows an example of an IP subnet that is used for connecting a sender and receiver host through packet forwarding. The key design consideration is that the end-to-end data forwarding is accomplished by the IP subnet but that the higher level service/connection are managed by the host processes. This architectural decision implies that any type of data can be sent between processes at sending and receiving machines.

While the architecture of packet switching and forwarding enables **resilience** to topology changes and enables data transmission of **any type of data,** the architecture poses some drawbacks for real-time multimedia services.

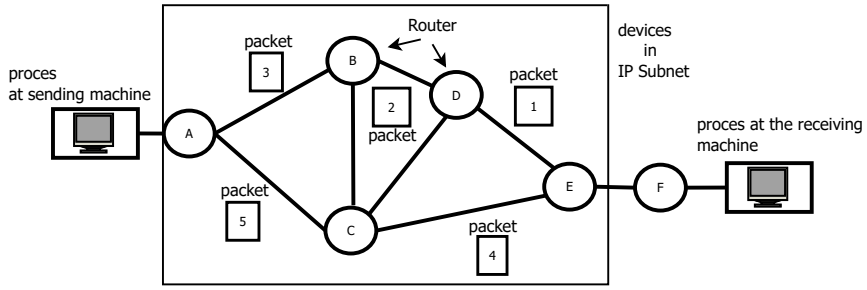


Figure 14 Hop by Hop routing IP Subnet, packets reach their destination over different paths

The following three artefacts are introduced by the packet forwarding and switching architecture:

1. Jitter, i.e. variable end-to-end delays due to queuing delays in the routers
2. Datagram loss (due to congestion (overflow of routers) and congestion avoidance control (random early drop)

3. End-to-End Delay (due to link delays, average queuing delays, retransmissions)

Table 2 Legacy mechanisms for QoS in the internet

QoS	**Description**	**Disadvantage**	**Advantages**
Overprovisioning	Increasing the network resources to avoid congestion and queuing delay	Costly to Achieve	Easy to implement, stable
Traffic policing/limiting	Limit the incoming traffic to clearly defined policies (often related to the SLA)	Not always applicable, policy violations, hard to control with many connecting users	No changes needed inside subnet
Integrated Services (IntServ) [32]	Pre allocate bandwidth on paths from sender to receiver based on resource reservation protocol (RSVP)	Hard to implement across Internet, changes needed to infrastructure	Strict QoS
DiffServ (DiffServ) [31]	Introduce per class hop behaviour, enabling different priorities for different classes	Soft QoS	Easier to implement across Internet (backward compatible)

For real-time and interactive media services these artefacts can degrade the user experience significantly. Therefore end-to-end transmission with constraints on packet loss, jitter and delay has been considered in the Internet. This is often referred to as Quality of Service (QoS). Table 2 briefly summarizes Architectures to achieve QoS in the Internet developed in past efforts.

As described in Table 2 each of these methods has significant drawbacks. The deployment of these techniques has been only a partial success as often too many changes in the infrastructure across the entire network were required. A novel and emerging technology to achieve Quality of Service in the Internet is based on software defined networking (SDN). In software defined networking, routers report their local topology information to the so called SDN Controllers. The SDN Controllers then determine the end-to-end paths and provides the routers with forwarding instructions. In other words, instead of having distributed routing protocol and algorithms running throughout the network the SDN controller becomes the "brain" of the network that can take these decisions regarding routing and forwarding priorities. OpenFlow [33] is one of the practical implementations of software defined networking, and carriers are already installing routers that support the OpenFlow protocol. Based on software defined networking, quality of services for multimedia services can be deployed such as based on approaches in [34] [35] . Software defined networking holds great promise to achieve QoS for multimedia, including 3D interactive Media enabling novel distributed shared experiences.

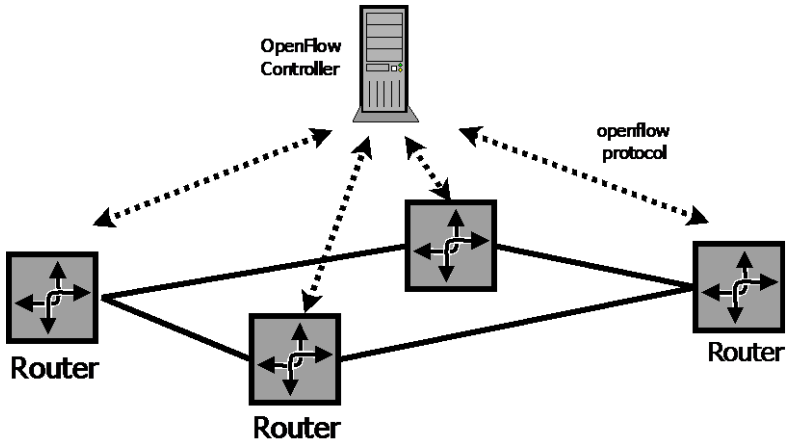


Figure 15 Software Defined Networking with OpenFlow, routers report their local topology information to the OpenFlow Controller

As some high demand content in the Internet is not changing much over time, a simple way to achieve QoS is to copy data and move it to a location as close as possible to the end user. This can be done by using network caches, creating a content delivery network (CDN) [36]. Approaches with CDN's work well for popular content that is requested by many users but less well for so called "long tail" content, i.e. content received by a very small amount of users. Nevertheless, for such type of media content CDN's are often employed to improve user experience and reduce backhaul Internet traffic.

2.1.3.3 Media Support Protocols

The internet protocol layer handles end-to-end datagram transmission. The connection and service are implemented at a higher layer. Multimedia protocols are used to setup, maintain and tear down multimedia sessions. These protocols have some knowledge about the codec configuration and bit rate and sometimes they are able to adapt to network conditions and expected user experience. Popular multimedia protocols include RTP/RTCP a dual protocol stack that uses RTP for its payload transmission and RTCP for synchronization and sender/receiver feedback. The RTP/RTCP [30] is based on application layer framing [37] that uses different profiles to different types of services (media). RTP/RTCP is used in many voice over IP

systems in conjunction with the session initiation SIP protocol. Recently, video/audio streaming over HTTP has become popular in a convergence of Apple's HTTP live streaming [38] and Microsoft smooth streaming [39] to MPEG DASH. All these media protocols use HTTP as the transport protocol and a text based Manifest file to signal the session (i.e. available bit-rates, content chunks). One disadvantage of the dependency on the media codecs is that it is harder to setup session using media that is not yet well supported in existing codecs.

Background Motivation 3: More bandwidth is becoming available in the access networks and other physical links. On top of that QoS in the Internet is potentially improving in next cycles due to Software Defined Networking

2.1.4 3D Media Compression

3D Visual data often represents huge amounts of bytes. Therefore, 3D data specific compression technology is critical for transmission over bandlimited networks and efficient storage. Two broad categories of 3D Visual data formats and their related compression technologies are of relevance. First, the coding of natural 3D visual, such as 3D video and image data is important. These formats generally represent multiple N by K (fixed resolution) images defining colour, gray scale or depth values on a regular grid. Such images are naturally acquired from digital (3D) cameras and have been used in moving picture, video surveillance, photography, digital television and many other applications. The second category of 3D Visual consists of 3D graphics/geometry based data, such as 3D point sampled data (point clouds) and 3D Meshes. Such data formats are generally represented by a number of point coordinates and triangles defined arbitrarily in the 3D space. These data do not have fixed resolution and often result from 3D authoring or 3D scanning and reconstruction. In the case of 3D scanning, data is often generated using non stationary or multiple 3D sensors or cameras combined with algorithms for data fusion and 3D reconstruction. The combined vertex + triangle data in 3D Mesh and Point Cloud data have been common in 3D Gaming, simulation, industrial and medical applications (CAD, tomography, X-Ray). In addition, they play an important role in geographic information acquisition and storage (terrain scans with 3D Lidar, SAR). Both data types can allow stereoscopic and/or free viewpoint rendering with appropriate rendering

techniques. For more information on end-to-end 3D capture to rendering and the difference between these formats, we refer to [40].

The state of the art in compression of naturalistic video data (the first category) is the most advanced. It has received heavy attention from both industry and academia in the last 25 years. This has led to some of the most widely deployed standards in information technology [41] [42] providing inter-operable compression in both hardware and software targeting compression ratios from 1:100 to 1:10000. For a more extensive state of the art in video coding and related principles we refer to chapter two of [43]. These principles have remained relatively unchanged in the last 25 years, with incremental improvements in each technique. Table 3 only highlights the basic principles of video/image coding and their (limited) applicability to 3D graphics. In video coding, redundant parts of the original signal are discarded taking the human perception system into account (eye sensitivity to lighting, high frequencies colour variation). Due to the unorganized nature of the coordinates in geometric data and the possibility of different rendering pipelines, it is harder to apply all these principles to 3D Graphics data.

The state of the art in compression of 3D Graphics data types has developed in a more academic setting [1]. It achieves compression ratios (single rate) ranging from 1:15 in practice to–1:40 (in theory, lossless). These compression ratios (lossless) are much lower when compared to the state of the art in video coding (lossy techniques), and often encoding times and memory footprints used by these codecs are huge. To decrease the data rate further, mesh and point cloud simplification can be applied. Simplification degrades the quality of the original content but when it is done in an intelligent way the effect on user perception will be limited (taking contextual specifics into account). In general compression terms, degrading the quality of the original data to achieve better compression is called lossy coding. On the contrary, lossless coding tries to keep the original signal intact, often only allowing a predefined quantization distortion to be introduced (this results from the analogous to digital conversion or floating point discretization). Lossy coding results in higher compression ratios and is most often used for video and still image coding. For 3D graphics coding, schemes that can be considered lossless are often used. This thesis will present a more extensive overview of the state of the art in mesh compression in Chapter 3 and on point cloud compression in Chapter 6.

Again, Table 3, outlines the principles of video and image coding and the potential difficulty to extend these principles to the coding of 3D graphics. We pass through each of these techniques. First, colour space conversion is often applied in video/image coding. This means converting red – green –blue samples into chrominance (colour) and luminance samples (brightness). As the eye is more sensitive to changes in brightness compared to colour, this can later be exploited by down sampling the chrominance images. However, 3D meshes and point clouds are often rendered using ambient and synthetic lighting, which alters the brightness levels. Therefore it is not clear what the exact effect is on this conversion is on the final rendering results. Next, down sampling of 3D geometry based data (decimating every other sample), is not defined as on regular grid, except for sub division meshes or quad meshes as in [44] and [45]. Chapter 5 and 6 of this thesis will experiment with down sampling/vertex decimation using octree structures instead. So this simple technique is already undefined for geometric data. Next, in video coding, rate distortion optimization plays an important role. This is often done by choosing an optimal tool for a given bit rate out of a set of tools giving the lowest resulting distortion. While this has also been done for graphics coding in [46], it has received much less attention in the literature compared to video coding. This makes it harder to compare lossy coding schemes for geometric content.

Next, inter-prediction is often used in video coders. Motion is estimated between subsequent blocks between frames. Motion compensation is used to improve the result. After this only residuals need to be coded that often turn out to be of lower entropy. While such techniques can be applied (even rather easily) in time varying meshes with consistent connectivity (same triangles and number of vertices in each frame), it is much harder to extend these principles to time varying meshes and point clouds without a consistent connectivity and varying number of points per frame. The search space for motion is enlarged in 3D compared to 2D and the points are sparser and often don't correspond between frames. This makes inter-predictive coding of time varying point clouds and meshes a challenging task, and a hard research problem that will be explored in this thesis.

Quantisation of coefficients and pixels is often performed in image and video coding. This can be extended to 3D geometric data, but from our experience, coarse quantization can result in quite large artefacts when coding 3D Geometric data. This is

because in a video frame a single defect pixel does not have a lot of effect. However, single incorrect points in a 3D mesh for example can affect the entire neighbourhood in the surface. Last, for 3D graphics a plethora of proprietary input formats exist such as Collada, XLT, ply, Obj, etc.. For video coding a few standardized input formats have been developed with common subsampling schemes makes working with these data types much easier.

Table 3 Principles of Video Compression and their limited applicability 3D Graphics data

Video/Image Coding principle	**Description**	**Problem applying to 3D Mesh/Point Cloud**
colour space conversion	Convert to a colour space of chrominance (colour) and luminance (brightness)	The influence of the 3D rendering pipeline on the final result is not taken into account.
(down) down sampling	Decimating values subsampling, representing chroma with lower resolution	Down sampling operation not defined (except for sub division meshes and quad meshes, see for example [45] and [44])
Rate Distortion Optimization	Choosing codec setting from a range of setting that gives the lowest distortion for a given bit rate	This has been tried [46], but further research is needed to account subsequent rendering pipelines and the effect of attribute coding (colour, normal etc.).
Intra prediction	DCT or wavelet based transforms, DC coefficient prediction based on neighbouring	Often only 1or 2 way prediction is used only. , Exception are found in spectral methods for meshes, but again they

	blocks. Exploits correlation between points grouped in blocks	pose restrictions on the connectivity (regularity).
Inter-prediction	Motion estimation and compensation between corresponding blocks	No fixed inter-frame pixel correspondence (except in time consistent mesh geometry) this makes inter-prediction more difficult
Quantisation	Some colour components or transform coefficients can be coarsely quantized	Coarse quantization of geometry positions can large big artefacts when rendering the 3D data
Standardisation of input format	Several well defined raw input formats	Many proprietary file formats (Collada, XLT, .ply .obj etc.).

Background Motivation 4: Compression of Geometric Content can enable mixed reality and 3D tele-immersive shared experiences. However, techniques for lossy and real-time coding of geometric content are much less developed compared to coding of video and images. The logical next step is better compression techniques for geometric content in mixed reality and 3D tele-immersive shared experiences

2.2 3D Audio Visual Capture Model

Technologies for acquisition, compression, transmission and presentation of audio visual media are often combined in 3D multimedia systems. It is in this area of interconnected systems that this thesis tries to make a contribution. While traditionally multimedia systems focus on audio and video capture, 3D audio and visual data formats introduce new challenges. Firstly, there are many 3D presentation types that use different camera types, different data formats and different rendering technologies. The combination of all these technologies can provide an improved user experience. However, the multitude of different technologies from capture up to rendering introduces the need for new strategies for interoperable compression and networking.

In the design of end-to-end 3D Media systems, the choice of 3D representation impacts the end-to-end pipeline of compression, networking and rendering [40]. To limit this impact, storage and transmission of media data often follow well defined data formats between processing changes (capture, compression, networking, rendering). These data formats are often precisely defined in documents ratified by international standardisation organizations or open source communities such as ITU, MPEG or JPEG etc. The advantage of this approach is that it achieves internationally recognized inter-operability. Different software, hardware and networks possibly from different vendors can access data by following the specifications. In addition, storage and transmission systems can be re-used by different acquisition systems more easily. This is a key benefit of standardized data formats.

To provide an exact overview of the problem space and interoperability points, we classify the data types in the end-to-end 3D Media pipeline generically. As for 3D media the amount of data types from new devices is exploding, it is important to have this exact classification to align technologies and classify systems. Figure 16 classifies the data types for interconnecting each stage in the pipeline as a type. It shows the block diagram that generically represents the 3D audio visual pipeline labelling each of the stages 1-8 [47]. Further details on the generic 3D audio visual representation can be found in Appendix A based on [48], which is an output of the MPEG organization for 3D Audio/Visual[1]. We detail this step by step. The first stage is the physical input signal (light/sound etc). This signal is sensed with sensors such as camera, microphone and usually results in digitized by sampling the measurement. For digitized data parameters like sample rate, sample frequency, quantization bits need to be chosen, and are typically standardized (16 bit PCM, 44.1 KHz sample rate for CD Audio etc). These digital media formats can be further processed, think for example of algorithms from signal processing/computer vision. In the diagram this is described as sensed data converter, which results in a type 3 format that is fed to the encoder stage. The type 4 data is then the compressed data format for networked

[1] The input contribution that led the MPEG standardisation organization to redifine this requirement was contributed in: m37050 **Mekuria R.N.** Lafruit G. Chiarglione L. 3D Visual Capture Inventory *MPEG 113 Geneva, Oct 2015*

storage and transmission. The type 5, 6 and 7 correspond to type 3, 2 and 1 respectively. Last in 8 commands and interactions can signal interactive operations and context over the entire pipeline. For 3D Media, these could be view changes (visual) head rotation or changes in the context of the virtual world.

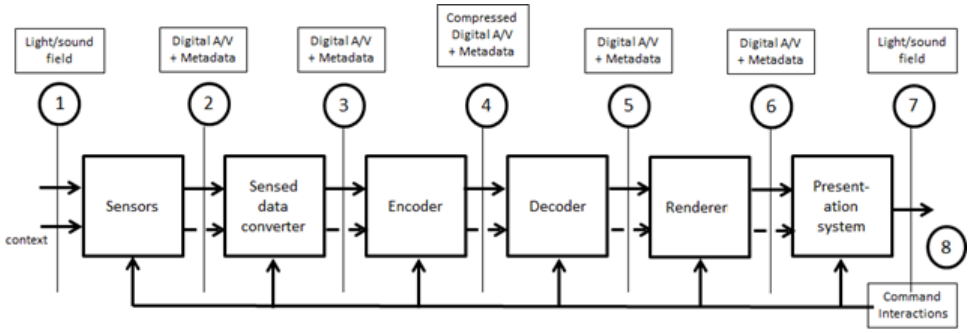


Figure 16 End-to-end Pipeline for 3D Audio/Visual source: MPEG Requirements [48]

For 3D visual/aural data the sensed data converter component is important and has often been overlooked in past efforts. Many Encoder and Decoder systems only target sensor output directly (images, sampled audio, video etc). The sensed data converter can convert this data in a more meaningful representation useful for the application such as meshes/point clouds for visual or sparse audio wave fields for audio. By recognizing more different type 3 formats novel encoder/decoder technology is needed to support the sensed data conversion operation. In this work we will investigate type 3 media based on point clouds and 3D Meshes. We refer to Appendix A and [48] for typically specified types for MPEG 3D Audio/Visual (type 3 / type 4 media data). In the next section we will look at the practical experimentation with real time end to end 3D communications, in so called 3D Tele-immersive systems.

2.3 Related Work on 3D Tele-Immersive Systems

A 3D tele-immersion application presents great challenges in capturing, networking and rendering technologies. 3D tele-immersive research dates back to the beginning of this millennium. The National Tele-Immersive initiative NTII, a consortium of American universities and industries, demonstrated a first 3D immersive system that captured 3D point cloud representations and stereo video rendered using stereoscopic displays [49]. This system presented immersive interaction between sites but was not optimized for efficient transmission. The focus of this work was on rendering and reconstruction. The system used the Internet2 and TCP/IP to send the point cloud representations and the stereo video uncompressed. In this system, video streams were sent from different machines to a cluster. As these different TCP/IP streams often utilize a common link, the TCP bandwidth congestion mechanism made such streams to compete, degrading the overall quality. Ott and Patel [50] developed a coordination protocol that allowed, in the gateway, coordination between the streams to alleviate this problem. At the University of Illinois, Yang et al. looked at the case of streaming multiple live-captured 3D videos in an overlay with 4-9 sites on the Internet2, leading to a large amount video traffic [51]. By adapting the forwarding mechanisms, in the overlay network, to the view of the specific recipient user, they achieved bandwidth gains by timely dropping irrelevant streams. Huang et al. looked at a similar problem, also taking into account the synchronization and skew level between the streams. They developed a scheme, called sync-cast [52], that allows video streams, in an overlay with multiple immersive sites, to be forwarded based on bandwidth, synchronization and latency requirements. Vasuvedan et al. proposed a 3D tele-immersion system for capturing and rendering. This system can in real-time reconstruct humans in the scene with high level of detail [53]. They used twelve clusters of four DragonFly cameras that reproduce 3D colour plus depth images of a human. On this depth image, triangular meshing is applied. The main advantages of this technique is that by interpolating points a more efficient representation of the depth image becomes possible. In addition, by changing the size of the triangles it is possible to adapt the level of detail. Wu et al. developed a streaming engine that exploits the aspect of this representation, [54], where the representation is referred to as colour plus depth and level of detail (CZLoD). In this study they first investigated human perception, by testing a set of samples generated with a stimulus

engine. The authors found the just noticeable degradation and just acceptable degradation levels, and subsequently used these for the dynamic adaptation of the CZLoD, depending on network and user conditions. With this engine, the real-time CZLoD can be adapted to match user perception for the given available network bandwidth and user conditions. This system represents the current state of the art in 3D tele-immersive systems. Note that the CZLoD representation is different from the full 3D polygonal mesh or point cloud based 3D reconstruction presented in this chapter. The CZLoD representation is a triangulation on a depth image, while a polygonal mesh is a triangulation in a full 3D space and the point cloud are points in a full 3D space.

While these works pave the way for 3D Tele-immersive communications, this thesis proposes a 3D Tele-immersive system architecture for mixing real and virtual content that that can be linked to social networks and deployed to massive audiences. As for such systems interoperability between components becomes a key factor, standardisation becomes an important theme. Therefore, based on this thesis several standardisation contributions have been made.

Chapter 3 3D Tele-immersion with Live Captured Geometry using Block Based Mesh Compression

This chapter presents an initial prototype for real-time end-to-end tele-immersive communication based on live reconstructed geometry. It is an end-to-end system integration of 3D reconstruction, compression, network streaming and rendering. This initial prototype reveals the gap between the state of the art of geometry compression and streaming and the requirements for 3D tele-immersive systems. By intelligently integrating and developing novel technologies for compression and data transport this prototype achieves real-time end-to-end communication. In this way it is out-performing the state of the art in geometry compression and transmission. In addition, this chapter discusses some of the benefits of geometry based data formats for free viewpoint rendering and mixing real and synthetic 3D graphics content. The 3D reconstruction component has been provided by CERTH/ITI, and the network coding mechanism by Queen Mary University of London, both will be presented briefly for the sake of completeness. The thesis contribution lies in the development of the real-time compression algorithm, the end-to-end integration and optimization and the reflection towards the state of the art 3D compression, 3D reconstruction, 3D tele-immersion systems development and networked transmission. This work addresses the second research question:

Research Question 2: How can we achieve real-time (under 300 ms motion to photon) compression and transmission of highly realistic reconstructed 3D humans based on meshes with State of Art Compression rates (such as in MPEG-4 TFAN)?

The work in this chapter is based on the following two publications:

Mekuria, R.N. Alexiadis, D. Daras, P. and Cesar, P. "Real-time Encoding of Live Reconstructed Mesh Sequences for 3D Tele-immersion." Proceedings of the Inter-

national Conference on Multimedia and Expo. International Workshop on Hot Topics in 3D (Hot3D) San Jose, CA, July 19 2013 (focusing on co-design of 3D capture and compression)

Mekuria, R.N. Sanna, M. Asioli,S. Izquierdo, E. Bulterman, D.C.A. and Cesar, P. A 3D Tele-Immersion System Based on Live Captured Mesh Geometry. Proceedings of the 4th ACM Conference on Multimedia Systems (MMSys 2013), Oslo, Norway, February 27- march 1 2013. (25.8% acceptance rate) (**Best Paper in special session on 3D MultiMedia**) (Focusing on end-to-end and compression/network streaming integration)

In addition, the compression and MMSys paper have been contributed to MPEG 113, April 2013 Incheon S. Korea (on invitation of the MPEG chair on 3D graphics). This has led to an exploration activity and core experiment in MPEG. In this context, results reported on compression in this chapter have been validated by a third party (Hanyang University in South Korea).

3.1 Introduction

3D Tele-immersion provides a common virtual space, where distributed participants can naturally interact. Advances on 3D reconstruction and rendering – and the success of the Microsoft's Kinect – enable, in real-time, the creation of highly realistic representations of the participants as triangle mesh models (see Figure 17). Efficient real-time transmission of these representations opens up possibilities for 3D tele-immersion and mixing real and virtual environments. Nevertheless, existing video codecs and packetisation schemes do not support such representations well (neither do geometry streaming mechanisms intended for downloading and interacting with remotely stored geometry-based objects). Therefore the first step towards this direction, would be an initial prototype that is capable of real-time efficient transmission of live-captured mesh objects between remote locations.

Figure 17 A 3D Mesh merged in a virtual world with an avatar representation

3D tele-immersion has been studied in the past for a variety of application areas such as creative dancing, cyber-archaeology, medicine, and gaming [55] [56] [57] [58]. While high-resolution video conferencing systems may facilitate basic interaction between distributed sites, they generally fail in providing users a natural way for eye contact (gaze), when more than two sites are involved, and they do not truly immerse users in the same virtual environment. Over the years, some attempts have been made to address these issues by using larger displays, multiple cameras, gaze correction mechanisms, and expensive fully furnished environments. Instead real-time transmission of live-captured 3D representations such as a triangle meshes can enable 3D tele-immersion of participants including their body and facial expressions for realistic immersion and interaction.

Capturing highly realistic representations of people in real-time was generally only possible at professional media studios using arrays of expensive stereo cameras and hardware for real-time depth estimation based on stereo correspondence. The recent commercial success of Microsoft's Kinect, which provides reasonable depth estimates, has brought inexpensive range cameras to the market (for less than $200). By deploying multiple Kinects from different angles, reasonable 360-degree geometric representations of people can be reconstructed (Figure 17). Moreover, the wide availability of general-purpose GPU and parallel computing, allows a speed optimization of the 3D reconstruction process to real-time [15] [53]. Finally, modern graphics

cards and displays allow multi-view and auto-stereoscopic rendering of such 3D reconstructions.

Looking ahead in the future, when current challenges regarding calibration and synchronization of the different Kinects are solved, 3D tele-immersion will become integrated into social network experiences (similar to current Google's Hangout video conferencing). Still, in order to enable real-time transmission of such geometric representations over the Internet, efficient compression and streaming mechanisms that can operate in a realistic environment will be needed. For traditional video, such mechanisms that can operate in real-time are already available: H.263+ and H.264 compression and network streaming profiles based on RTP/RTSP. However, if the captured human representation for 3D tele-immersion is geometry-based (e.g., triangle mesh), existing codecs and packetisation schemes cannot be re-used, as they do not support this format. Therefore, to support 3D Tele-immersion, there is a need for a compression and streaming for live-captured geometries.

Over the years, various schemes for geometry-based compression have been developed, mostly aimed at efficient rendering. Transmission schemes and middleware solutions for real-time streaming over lossy networks have been developed as well, generally dealing with remote downloading of stored objects [59] [60] [61]. None of these approaches handle the critical real-time requirement as imposed by live-captured triangle mesh for 3D Tele-immersion.

The rest of the chapter is structured as follows. Sections 3.3 overviews the related work regarding existing mesh geometry compression algorithms, and mesh geometry transmission solutions. Next, 3.4 introduces the streaming pipeline of my 3D tele-immersion system. Section 3.5 introduces the 3D reconstruction module introduced by CERTH/ITI. Section 3.6 presents a specific compression method designed for captured meshes, comparing it with some existing mesh compression schemes. Section 3.7 describes the proposed transmission scheme based on a rateless code, reporting results that highlight the benefits of this approach over TCP transmission. Section 3.8 presents the 3D Tele-immersion system, highlighting the performance of the overall system. Section 3.9 discusses the implications of the work. The next section motivates this work and discusses the chapter contributions.

3.2 Motivation and Research Question

Real-time streaming of live-captured video is common in video conferencing systems, but streaming of captured triangle geometry objects has rarely been considered in the past. There are various reasons why efficient real-time transmission of live-captured triangle geometry essential for the next-generation of 3D tele-immersion systems. First, modern graphics cards can take advantage of advanced rendering methods, such as multiple views for stereo and multi-stereoscopic or free-viewpoint rendering. Second, it allows for easy integration with virtual worlds, where triangle mesh representations are common. Finally, novel application areas can emerge such as natural interactions between people in real and virtual worlds. Geometry Streaming solutions can be also beneficial for applications like camera or terrain surveillance, where geometric data is live-captured and needs to be available in real-time to observers or automatic engines. This chapter is concerned with the 3D tele-immersion application for interactive purpose.

In particular, this chapter aims to provide a first answer the following research question:

Research Question 2: How can we achieve Real-time (< 300 ms on commodity hardware) compression and transmission of highly realistic reconstructed 3D humans based on meshes with State of Art Compression rates (i.e. MPEG-4)?

The main contribution is the design and development of a streaming engine that takes into account the specific requirements for streaming such high resolution captured 3D meshes. The requirements are the following:

Support a full 3D triangle mesh representation: the engine should support streaming of the common 3D triangle mesh representation. That is, a list of points with properties (coordinate, normal and colour) and a list of faces indexing these points, resulting into a surface in the 3D space.

Low end-to-end Latency is generally considered the most important factor in 3D Tele-immersion, as the pipeline consists of bandwidth and computation savvy operations. For this chapter we aim to provide an end-to-end transmission latency below 300 ms, based on video conferencing requirements.

Flexible I/O representation: the data should efficiently flow from the capturing and reconstruction blocks, via the streaming engine, to the renderers without blocking. To allow for minimum pipeline delay and possible synchronization between different streams a flexible I/O scheme is needed

Robustness to packet loss as it occurs in congested networks is desired. Some quality degradation may happen, but it should be possible to reconstruct the triangle mesh at the receiver in case of packet loss.

Bandwidth: the triangle mesh stream should not consume too much bandwidth; some form of compression is desired. Target should be compression rates achieved by state of art geometry compression standards such as MPEG-4 TFAN.

No a priori information of the geometric properties of the objects can be assumed. The application should be able to stream any triangle mesh that is currently captured. Similar to video conferencing, any object that is captured should be streamed. This means that no pre-stored avatars or models can be transmitted as placeholders.

Real time Live-captured triangle meshes should be supported. Contrary to live captured video, where frames generally consist of a fixed number of points (320x240, 640x480), captured triangle mesh frames can have different numbers of points i.e. no such point (pixel) correspondence between frames can be made. This means that the triangle meshes have to be transmitted as a sequence of static meshes. To my knowledge, a mechanism to estimate this correspondence between captured points in real time does not exist.

Contribution: this chapter presents a component that can efficiently stream triangle mesh geometry that is live captured and reconstructed in real-time. Two main subcomponents are presented: encoding and transmission. It introduces an encoding mechanism that reduces the size of the reconstructed mesh to values similar to those obtained with a state of the art TFAN MPEG encoder, but over 10 times faster in section 3.6. This meets the requirements on latency and bandwidth. Second, it details a transmission scheme that uses a rateless code, meeting the requirements of robustness, latency and adaptability to changing network conditions. The methods are integrated with state of are 3D capture and rendering to test the real-time end-to-end performance.

3.3 Related Work

3.3.1 Compression of Triangle Meshes

Peng [1] provides a survey of a number of compression methods up to 2005. Some common terms in mesh compression are single rate encoding, which refers to coding at one quality level; progressive encoding, which allows lower quality reconstructions if part of the data is received. Mesh coding can be connectivity driven (encodes indices first) or geometry driven (encodes vertex data first). Generally, geometry encoding can be lossy, but connectivity coding should be lossless to maintain the original topology. The single rate encoder that achieves the highest compression rate is the Touma and Gotsman encoder [62]. Compressed progressive mesh [63] is a compression scheme that reconstructs a mesh by successive vertex splits and allows both more coarse and more detailed views. Streaming compression, as proposed by Isenburg [64], can work on small parts of the mesh, making it more memory efficient and faster (large meshes may be too large to fit the main memory). Currently, standardized compression of 3D graphics such as geometry meshes is available in the Motion Picture Experts Group (MPEG) MPEG-4: Scalable complexity 3D Mesh Coding (SC3DMC). The standard provides 3 types of encoders with various settings in complexity to allow a trade off between decoding speed and compression ratio [16] [65]. The most sophisticated coder, the triangle fan encoder, is reported to achieve a compression rate close to the Touma and Gotsman encoder.

Many other mesh compression methods that exploit various topological properties have been developed. They offer the compression performance needed to reduce the large data streaming rate required in 3D tele-immersion. Unfortunately these encoding mechanisms have not been explicitly designed for real-time encoding, transmission and robustness to packet losses needed for 3D Tele-immersion. A loss of connectivity data, for example, could destroy the entire topology.

3.3.2 Transmission of Triangle Mesh Geometry

Robust real-time transmission of compressed mesh geometry over lossy networks is challenging. Regib and Altunbasak [60] propose 3TP, a streaming protocol that sends parts of the compressed data over TCP and other parts over UDP. Given an

allowed acceptable degradation of the mesh, 3TP selects a combination of packets to be sent over TCP and UDP. This selection is then optimized for reducing the transmission delay given the current level of packet loss. 3TP gives the user a lower download time, but extensive offline pre-processing of the object is required to do the selection. This makes it unsuitable for the interactive streaming case, where geometry is captured, encoded and transmitted live. Another somewhat similar but more generic approach, by Li et al, is a middleware that allows streaming of various different compressed mesh representations [61]. The essence of this approach is that subsets of the representation can be sent reliably or unreliably based on the type of data (i.e. geometry/connectivity), the loss rate and the type of user environment (renderer, terminal type). This approach works for any type of mesh representation, as values are calculated offline using the general distortion measure for mesh geometry Hausdorff distance, described in [66]. This makes it more generic, but less applicable for live captured data due to the complex pre-processing step. Cheng et al. studied dependencies between data, when streaming compressed progressive mesh (CPM) based representations [59]. They found that packet loss in the initial phase, when reconstructing from a coarse mesh with vertex split operations, blocks the decoding process. When a vertex split packet is lost, many other packets cannot be decoded until the packet is retransmitted and successfully received. Cheng et al. do not solve this problem, but model the effect mathematically. This mathematical model of the dependencies in the data (using a graph representation) is used to find an optimal packet scheduling strategy (order of sending packets). Experimental evaluation shows that they are able to reduce the delays by minimizing long term dependencies between packets.

3.4 3D Tele-immersive Streaming Pipeline

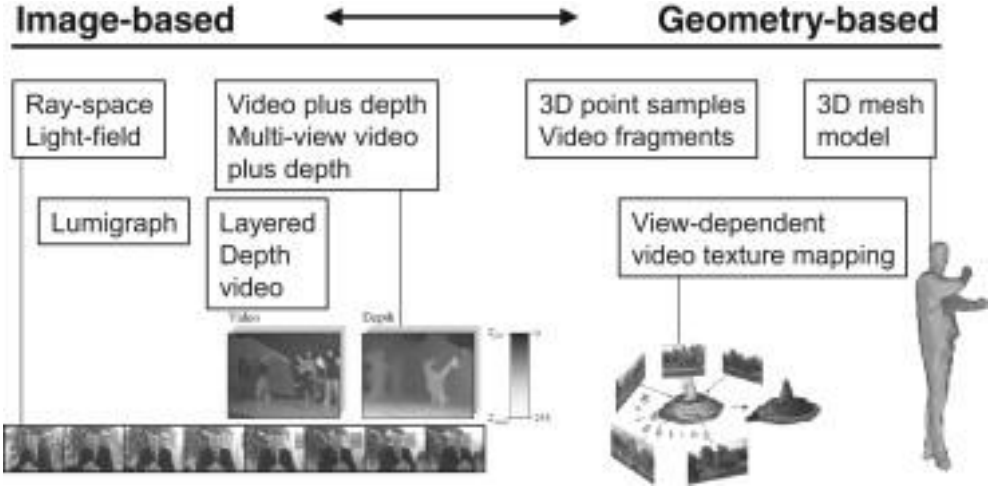


Figure 18 3D representations, from image based to geometry based

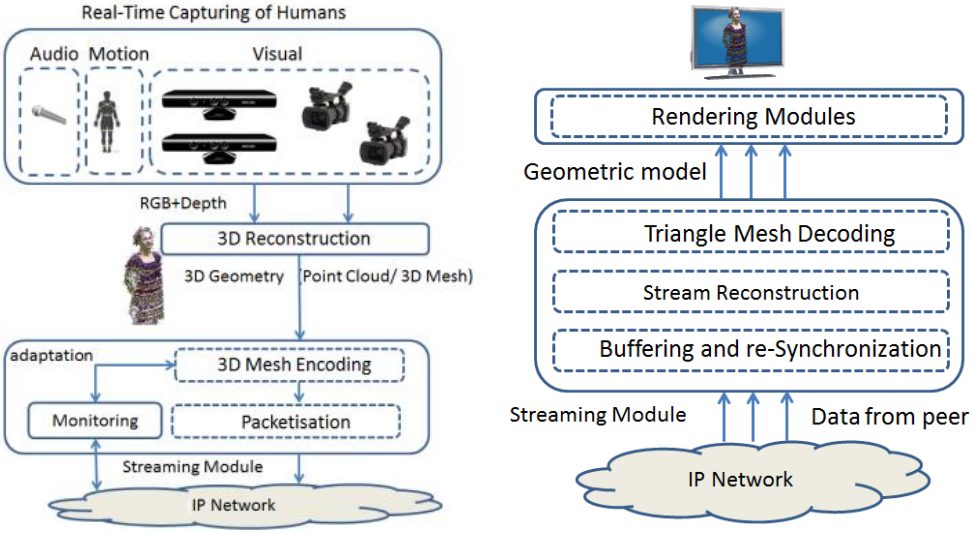


Figure 19 3D Tele-immersive media pipeline

3.4.1 3D Representation

3D video can be interpreted in different ways, such as the 3D stereo video in the cinema with an artificial depth perception by rendering a left and right image, or the free-view video that allows viewpoint navigation. The work in [40], which is particularly useful, categorizes different representations on a spectrum between image-based methods and geometry-based methods (Figure 18). Examples of geometry-based representations include triangle meshes and point clouds. Image-Based methods, on the other hand, are similar to traditional video, since they use multiple separate (possibly) interpolated views. Traditionally, geometry-based methods have been restricted to games, computer aided design and virtual worlds, while image based has been used for television, broadcast and video.

3.4.2 Media Pipeline

Figure 19 shows the media pipeline at the sender site. Further details about the individual components of the system, and reports of comparison to existing mechanisms, are provided in the next two sections. First, humans are captured in real-time. Different media are captured, audio, motion and visual. This chapter focusses on the visual pipeline for a mesh geometry representation of a human. As humans interact in real-time, low delay is required. After capturing, the representation has to be encoded using an efficient compression method. Subsequently, streams and packets are sent over the network. The right side of Figure 19 shows the streaming module at the receiver site. The received packets are first buffered and synchronized. Subsequently, the re-construction of the stream takes place, and rendering can be done.

3.5 3D Reconstruction

This section shortly presents the employed real-time frame-by-frame 3D reconstruction method, from multiple consumer depth cameras. The method shares similarities with [15] and is based on the notion of Step Discontinuity Constrained Triangulation (SDCT), i.e. terrain triangulation on the depth 2D image plane. This module is provided by CERTH/ITI in the context of the REVERIE FP7 project and presented here only for the sake of completeness based on our joint work in [67].

3.5.1 Capturing setup and calibration

A calibrated capturing setup with five RGB-Depth sensors (Kinects) was implemented. The sensors are connected on a single host PC with high processing power, as well as a CUDA enabled GPU. One sensor is placed horizontally at a height of 1.30m, to capture the front upper body, while the remaining four are placed vertically, at a height of approximately 1.80m to capture the whole human body. The sensors are positioned on a circle of diameter 3.60m, all pointing to the centre of the working volume, introducing a circular "active" region of diameter approx. 2.40m

In order to fully calibrate each single Kinect, the method of [68] was used, which simultaneously estimates the depth and the RGB camera intrinsic parameters, the relative pose between them, as well as a depth (disparity) distortion model. With respect to the external calibration of the multiple Kinects network, a custom-made calibration object with three intersecting large planar surfaces was used, which is moved inside the working volume and captured simultaneously by all sensors. For each captured frame, the planar surfaces are detected in the depth images. Then, a fast coarse pairwise calibration is realized, based on the normal vectors of the detected 3D planes, followed by minimization of the mean squared distance of the reconstructed points on the three planes with the corresponding planes in a reference camera;. Finally, a global (all-to-all cameras and for all frames) ICP optimization procedure is realized.

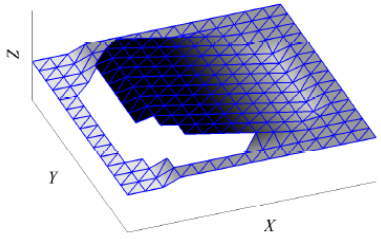


Figure 20 the idea of SDCT. Each 2 × 2 square neighbourhood of a depth map is bisected into two triangles, unless the absolute difference of the neighbour depth values is greater than a predefined threshold (= 2cm for the presented results).

The overall reconstruction approach can be summarized as follows:

- **Pre-processing:** 1) A weak 2D bilateral filter is applied to the depth maps to reduce Kinect measurements noise. 2) Additionally, a binary "human silhouette" map is generated for each depth maps, by segmenting out the foreground object of interest (human). Background subtraction is realized by checking the absolute difference of the current frame from a "background" depth image, accumulated in multiple frames.
- **Triangulation:** From each depth map, we construct a mesh via SDCT, considering the depth values only inside the "human silhouette" map. The idea in SDCT is that depth measurements that are adjacent in the 2D depth image plane are assumed to be connected, unless their Euclidean 3D distance is higher than a predefined threshold (= 2cm in this case). The idea is illustrated in Figure 20. Since the spatial sampling step in *X,Y* is very small compared to this threshold, it is adequate to consider only the depth (Z) difference of the adjacent measurements, This saves execution time. SDCT is formally summarized as follows: i) For each pixel $Z = D(u,v)$ of the depth map, consider its adjacent pixels at the right, right-bottom and bottom, $Z_r = D(u+1,v)$, $Z_b = D(u,v+1)$, $Z_{rb} = D(u+1,v+1)$, respectively; ii) If all depth distances $|Z - Z_r|$, $|Z - Z_b|$ and $|Z_r - Z_b|$ are below the threshold, generate a triangle by connecting the corresponding 3D points; iii) Similarly, if all $|Z_r - Z_b|$, $|Z_r - Z_{rb}|$ and $|Z_{rb} - Z_b|$ are below the threshold, generate a second triangle.
- **Post geometry smoothing:** A fast mesh-smoothing operation is applied to the reconstructed mesh, which calculates the average position of each vertex with its connected neighbour vertices. Two iterations are used.
- **Weighted colour mapping:** The RGB colour of each vertex is obtained as the weighted average of the pixel colours in all visible cameras that the vertex projects to. Visibility is inferred by comparing the actual Z distance of a vertex to a camera and the corresponding depth value observed by the camera. The angle (inner product) between the vertex normal and the line connecting the vertex with the camera is used for weighting the colour observations. Practice showed the use of such a weighted colour mapping scheme can significantly improve the visual quality.

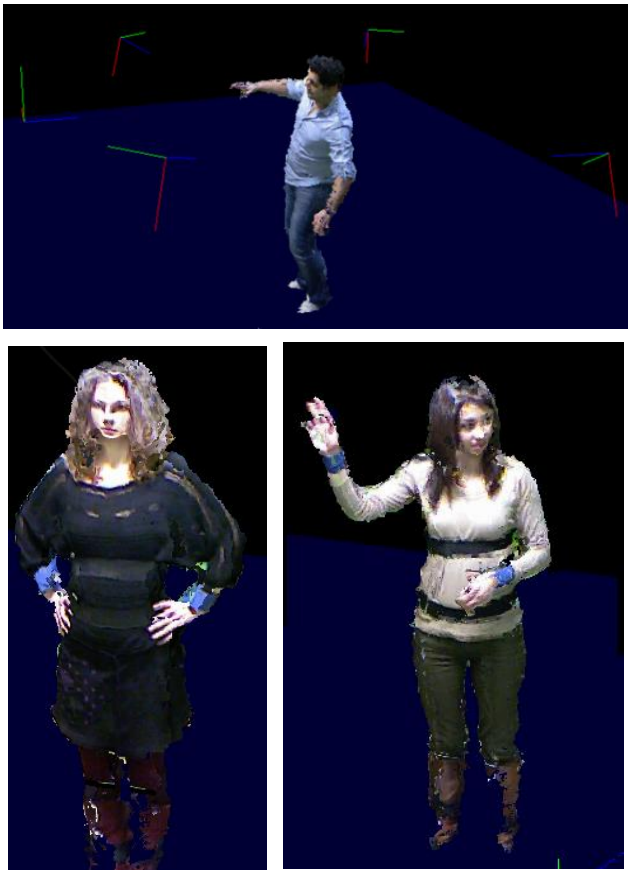


Figure 21 Examples of real-time reconstructions. In the upper row, the coloured lines indicate the positions and orientations of the cameras [15]. (CERTH/ITI [2])

3.6 3D Data Compression strategy

Compression is one of the key methods to relieve the high bandwidth demands. Captured meshes can have a different number of independent points and faces (triangles). Due to the lack of such a direct relationship between the frames, it becomes necessary to investigate static mesh codecs to encode each mesh independently, without inter-frame coding.

Table 4 qualitatively compares the different static mesh compression mechanism identified in the literature for their properties and applicability. Then, the solution to the problem, specifically addressing the requirements for enabling a 3D tele-immersive system will be presented. The performance of the method will be compared with the MPEG SC3DMC Codecs.

3.6.1 Qualitative Comparison Existing Mesh Compression Methods for 3D Tele-Immersion

This section qualitatively compares compression methods based on the following criteria: **C** *Compression Rate,* **E** *Encoding speed,* **D** *decoding speed,* ***SC*** *scalable complexity,* ***L*** *tolerance to loss* and **P** *progressive* transmission, **T** *shape topology independence (*if the compression method is geared to a specific triangle mesh topology*)*. They are rated from 1 (bad) to 5 (very good). These criteria are used to make a selection of mesh encoders that are suitable for real-time streaming.

Touma and Gotsman propose an encoder that encodes at high rates, intended for semi-regular meshes. In the literature, this codec is considered the best in terms of compression rate. This encoder is not optimized for losses, progressive transmission, or real-time encoding. Gumhold and Strasser propose a fast (real-time) compression and decompression technique that codes at high compression rates [69]. Looking at this approach further, a specific disadvantage can be identified. The data structure that is fed to the algorithm (that makes it run in linear time) is not a typical list of points and faces, but a data structure that allows random access to a vertex indexed based on an edge. Creating such a data structure from the captured mesh representation would minimize the reported speedup advantages. Attempts to obtain such data structure by using algorithms available in the C++ standard library, showed that this was not trivial. Specifically, we created a map with pairs of points (edges) to index the other vertices in a typical captured mesh. This already resulted in delays of over 200 ms on an Intel i7 machine compiled with a 64-bit Visual Studio Compiler in release mode. The three SC3DMC encoders standardized in MPEG-4 SC3DMC (2009) have been developed for fast decoding and have many coding options that provide scalable complexity.

Table 4 Qualitative Comparison of Triangle Mesh Compression methods

Encoder Name	C	E	D	SC	L	P	T
Touma and Gotsman [62]	5	1	3	3	1	1	2
Gumhold and Strasser [69]	5	3	5	2	2	1	3
TFAN (SC3DMC) [70]	5	2	4	3	2	1	2
SVA (3DMC) [65]	3	4	4	4	3	2	2
QBCR (3DMC) [65]	2	5	5	3	3	2	4
Khodakovsky et al. [71]	5	2	4	5	4	5	1
Pajarola and Rossignac [63]	4	3	5	4	1	5	2
Taubin and Rossignac (3DMC) [72]	4	2	2	1	1	1	2

The QBCR and SVA [65] are faster, but simpler codecs. The TFAN [70] is a more complex single rate encoder, which allows fast decompression for rendering and achieves compression rates near to the Touma and Gotsman encoder. Up to now, MPEG or IETF have not published any scheme for transmission using these codecs over the Internet. However, their speed may make them useful for 3D tele-immersion application. Khodakovsky et al. [44] developed a progressive wavelet-based coder. The advantage of this approach is that reconstructions are possible, even with partially received data. A disadvantage is that it works only for semi-regular meshes. The current implementation is available, but currently works with ASCII-based files as an input, introducing extra delays in the encoding. Compressed progressive Mesh (CPM) presented in [63] was used in many studies on transmission of geometries. It

offers progressive transmission, but losses of packets can delay the decoding process as studied in [19]. Apart from that, this codec is not specifically optimized for encoding speed. The approach of Topological surgery was standardized in MPEG 3D Mesh coding (MPEG3DMC) [72], but in practice it has not been used much mainly due to its slow encoding speed and on its dependency on MPEG BIFS. Also, we compressed some meshes with an open source compression mechanism, OpenCTM[2]. This software employs entropy coding and quantization to compress meshes, instead of state of the art compression techniques geared to 3D meshes. This codec introduces a delay of over 1 second when applied to the modules captured in the system and did not achieve rates comparable to other methods that are described in the scientific literature.

The three MPEG-4 SC3DMC encoders with different complexities and options provide good compression rates and fast decoding. We chose to integrate them with the system for further evaluation. The software, available on mymultimediaworld.com also works with ASCII (text based) formats. The classes provided by this codecs can also be used to directly encode the binary incoming indexed face set, the common representation of polygon a Mesh. To achieve this, an extra C++ class to interface this codec was written, integrating it with the 3D tele-immersive system.

3.6.2 Fast Compression Heuristic for Captured Meshes

None of the current solutions for 3D triangle mesh compression are specifically geared to the envisioned 3D tele-immersion use case. In order to address this issue, this section develops a fast local compression method that is capable of real-time encoding and decoding and that meets the requirements. Unlike generic mesh compression methods, this algorithm takes advantage of the properties, present in meshes reconstructed from multiple depth images. The two most important observations exploited are:

1. Subsequent coordinates in the list of vertices are co-located, we exploit this with differential coding and local quantization.

[2] www.openctm.org

2. The face indices resulting from triangulation on multiple depth images are highly structured and show repetitions that can be exploited.

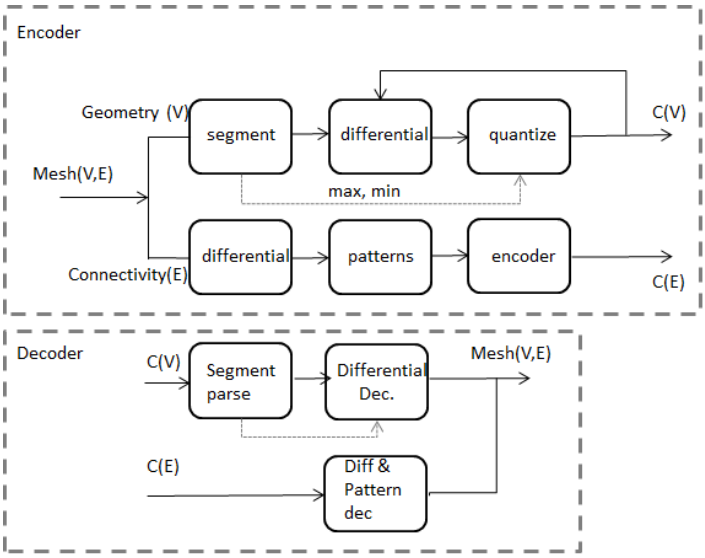


Figure 22 Overview scheme of 3D designed compression

Figure 22 presents an overview of the developed compression algorithm. It includes a part for compression of the geometry by local differential encoding and a connectivity coder based on coding specific patterns in the mesh connectivity.

The algorithm for compressing the geometry (points) is presented as Algorithm 1, the connectivity compression method in Algorithm 2. The code is provided as pseudo-code in Tables 2, 3 and 4. Algorithm 1 takes the mesh geometry (a list of points) from the capturing component and processes it piece by piece. The algorithm allocates storage for the expected number of compressed blocks stored in coded_data. In the *while* iteration the geometric data points are processed into compressed blocks of an approximately constant size of about 1436 bytes. Algorithm 1b illustrates how the individual blocks are processed. For each local block of data, first the maximum difference in its range is computed with the method *compute max diffs per vertex value()*. Based on the local maximum difference per subvalue quantization vectors are computed. These vectors are used to quantize the rest of the differences

between the specific subvalues in the block. A non-linear base quantizer was adopted with a higher resolution in the lower values, as experimentation showed that this kept more specific details in the mesh. In line 3 in algorithm 1b, based on the computed maxima, the number of vertices for each block is computed. By default each value is assigned 4 bits, but 0 bits will be assigned if certain subvalues do not change. As the blocks maintain about equal size, the number of vertices in the packet differs, based on this allocation (this is calculated by computing n_vertices for required datablock_size (P)). Subsequently, in the *for* loop in Algorithm 1a, the differences are encoded by quantizing them with the local quantizer (that was computed from the maxima). The values are stored in the block coded_data. The starting coordinates, local maxima and the number of vertices are also stored in the compressed block (data structure c_block), as they are needed in the decoding process. The index value is currently also added, in the future this could allow more flexible processing such as parallel or on the fly execution using GPU's. Currently the blocks are encoded and decoded in linear fashion.

Table 5 Algorithm 1a geometry compression

ALGORITHM 1a	**Geometry Compression**
INPUT: P	Block of floating point geometric data: nV vertices with w floats of data per vertex
OUTPUT: c_data, N	N blocks of piecewise compressed data (c_data)

```
COMPRESS_GEOMETRY P, nV, w:
1 c_blocks  = c_block[max_number_of_blocks]
2 nr_vertices_processed = 0
3 nr_blocks = 0
4 pos = 0
5 While  (nr_points_processed < nV)
6       compress_geometry_block(P.next()  c_block.next())
7        nr_points_processed += c_block->nr_vertices_in_block
8        nr_blocks++
9 End
```

```
10 return   c_data, nr_blocks
```

ALGORITHM 1b
COMPRESS_GEOMETRY_BLOCK P, &block :

INPUT: P, block	P: block of floating point values representing local points of the mesh , Block: an empty struct c_block representing a compressed block of data (to be filled)
OUTPUT: n_vertices, block	The number of vertices encoded in the block, the filled block of compressed data

```
1 Vector max_diffs = compute_max_diffs_per_vertex_value()
2 q_vec = compute_quantization_steps(max_diffs,  w_vec)
3 nr_vertices=compute_n_vertices_for_required_datablock_size(P)
4 coded_data[w][n_vertices]
5 For(j=1….w)
6   prev[j] =P[j][0]
7   For(i=1…..nV){
8     diff =  P[j][i]- prev[j]
9      cindex == qvec.find_index_closest_to( diff)
10    coded_data[j][i]  = cindex
11    prev[j] =    qvec[cindex] +  prev[j];
12  End For
13 End For
14 Block->start_coords[0…w] = P[0…w][0]
15 Block-> max_v[0…w] = max_diffs[0…w]
16 Block-> nr_vertices = nr_vertices
17 Return n_vertices
```

DATASTRUCTURE C_BLOCK

```
Int start_index,  int nr_vertices;
vector max_v    ,vector start_coords
```

byte[] coded_data

Algorithm 2 handles the connectivity compression. The mesh reconstruction process introduces repetition patterns in the connectivity data. The patterns were repeating differences [+a,+a ….] and alternating differences [+a,+b,+a,+b] or [+a,+a,+b,+b.]. As such, patterns can occur many times, actively searching for them and encoding them as a coded sequence which we call a run obtains a large reduction in connectivity data size. Algorithm 2 starts by initializing three vectors to store the differences between points of subsequent faces. For example if face 1 <a,b,c> and face 2 <d,e,f>, then these differences will be d-e, e-b and f-c. In the beginning of Algorithm 2, we run over the entire list of faces to find these patterns (*find_run_pattern*) and store them in pattern_runs. Subsequently, in the *for* loop in Algorithm 2, either a difference between values in consecutive faces is stored in T_Coded, or, if a pattern_run was previously found, this pattern is added to T_Coded. Specifically, this is done by adding an escape value to the T_coded vector and the sequence of values representing the pattern_run (see datastructure pattern_run). The loop index *i* is then incremented with the length represented in the pattern run. In this way, the indices are either stored as 16 bit differences, or encoded in a pattern run. Specific entropy coding such as Huffman and Entropy encoding are avoided, so no extra latency is introduced. In practice, over 90% of the connectivity information is stored in runs.

Algorithm 2b represents the pattern search algorithm. In this specific case, it stores repeating differences, and when a pattern is broken that has been repeated more than a threshold (32 times was chosen), Ithe pattern is stored as a pattern run. These patterns are then assessed in Algorithm 2.

Table 6 Algorithm 2

ALGORITHM 2	**Connectivity Compression**
INPUT: T , nT	Array of 3 by nT representing the Triangles
OUTPUT: T_coded	3 vectors with coded geometry data

```
COMPRESS_CONNECTIVITY T, nT
1 T_coded[3][]
2 pattern_runs[]  = find_run_patterns(T,nT)
3 For( j=0….3)
4   For(int i=0…nT)
5     d_[j] =  T[j][i] – T[j][i-1];
6     T_coded[j] .append(d_[j] );
7     If( pattern_run.colum == j AND pattern_run.start ==i)
8        T_coded [j].insert_pattern_run(pattern_run)
9        pattern_run = pattern_run.next()
10       i.increment(pattern_run.length)
11     End if
12   End For
13 End For
14 return   c_data, nr_blocks
```

ALGORITHM 2a FIND_RUN_PATTERNS

INPUT: T[w][nT] , nT	Array of 3 by nT representing the Triangles
OUTPUT: run_d[3]	3 vectors with pattern run data structures

```
run_d[3][]
diffs[4]
run_counter[nr_modes]
first_triangle[3] = T[0…2][0]
For(i=2; ……nT )
  diffs = compute_local_diffs
  if(pattern = find_pattern())
    run_counter.increment()
    if(pattern broken and run_counter > thresh)
      patterns.add(pattern);
    End
  End
End
```

DATASTRUCTURE PATTERN_RUN

```
Int mode, length, diff1,diff2,start,value;
```

The decompression algorithm is provided as pseudo-code. The blocks of compressed geometry data, represented as *c_block* datastructures are all subsequently processed by algorithm 3a. Algorithm 3a re-computes the local quantization vector of the differences based on the *max_v field*. Then, based on start_coords the differential decoding of the geometric data is performed. The second decompression step involves the decoding of the connectivity data. The first values of T_Coded represent the first face, then based on this face differential reconstruction of the different face columns is performed. When an escape value is found in T_Coded for a run, the run is decoded into respective column of the faces. By processing all the values in T_Coded, the complete connectivity is reconstructed.

Table 7 Decompression Algorithm

ALGORITHM 3	**De-Compression**
INPUT: c_data[N] T_coded	N coded data c_blocks from algorithm 1 and the vectors T_coded obtained from algorithm 2
OUTPUT: P[w][nV], T[3][], nT	The geometry data: nV vertices of w floats each, the number of faces nT

```
DECOMPRESS_MESH c_data, T
P[][]
For(block in c_data)
  P.append(decode_c_data_block(block))
  For(j=0…3)
    T[j][0] = T_coded[j][0]
    For(i=1…… nT, k=1…..nT){
      If(T_coded[j][k] == run_start )
        T[j][i…i+runlength] = decode_run(T_coded[j][i])
        i.increment(runlength);
        j.incremenent( 6);
      Else
        T[j][i] =  T[j][i-1] + T_coded[j][i];
```

```
    End If
  End For
End For
```

ALGORITHM 3a	**Decode c_data_block**
INPUT: c_block	A coded block of geometric data (a struct of c_data)
OUTPUT: P[w][]	Block of points from the decoded mesh of w floats per vertex

```
Prev[0…w] = c_block-> start_coords[0….w]
Compute_local_quantization_bounds( c_block->max_v)
For(i=0…..w)
  prev = c_block->start_coords[i]
  For(j=0……c_block->n_vert)
    P[i][j] = prev + q_diff( c_block->coded_data[i][j])
    Prev = P[i][j]
  End
End
```

3.6.3 Experimental Results

This section evaluates the performance of the method with live captured data. Figure 22 to Figure 29 show the results in terms of compression size, coding latency and distortion.

The scheme was implemented in C++ using Visual studio and compiled using a 64 bit compiler. The tests ran on an Asus laptop (PRO64J) with a first generation (1.6 GHz) mobile Intel i7 processor with 4GB of ram and Windows 7 home edition. The MPEG SC3DMC Codecs were compiled from source code with the same compiler and ran in the same environment. The tests were run offline with previously captured data stored in files. All files are first completely loaded into memory before the compression routine is started. The running times are recorded with operating system wall clock times in boost C++ that provides a wall clock time in Windows 7 with a resolution around 366 ns.

The first set of experiments (Figure 23 to Figure 25) show the performance when the capturing device is a single Kinect and the capturing mechanism is tuned to capture objects within a 130 cm range. This represents a situation where a user is behind a pc with a Kinect on it. We captured high-quality representations for this dataset with on average 72,855 vertices per frame and 143,302 faces. Each vertex points contains 9 floating point values, 3 for coordinates, normal and colours each. The raw frames are therefore about 4.3 MB each. As shown in Figure 4, the size is reduced by more than a factor 10, close to the performance of triangle mesh Encoder TFAN (tuned with 8 bit quantization and differential encoding). Figure 5 shows a qualitative comparison between the reconstructed frames from the different coding mechanisms. This qualitative comparison is based on the Haussdorf distance (rms) and measured with a tool developed in [66]. The values measured represent the root mean square distance between the original and the reconstructed surface. The models decoded with the scheme have slightly less distortion and in theory are slightly better reconstructions. Note that the distortion differences (Haussdorf distance rms) between the models of 0.0004 are not significant (the reconstructions are of comparable quality). We chose the quantization values such that they allow a fair comparison between different algorithms (operating at similar quality). As the method has lower distortion, it is fair to compare speed and size (assuming the quality is at least as good from human perception). The main gain of the heuristic is in the speedup. The proposed method can encode the high quality representation in about 70 ms, and decode it consistently below 10 ms.

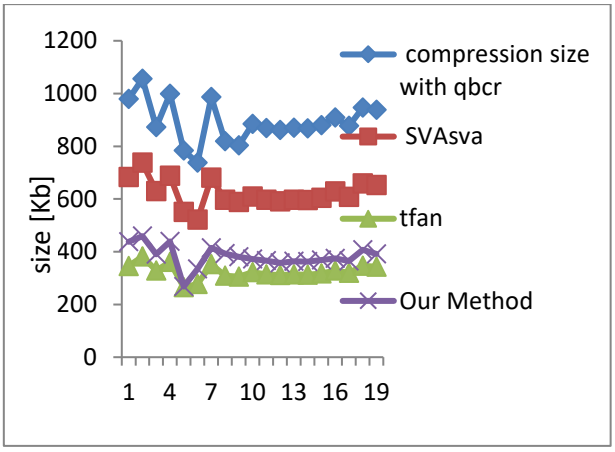


Figure 23 Compressed size (130 cm 1 Kinect) (Kb)

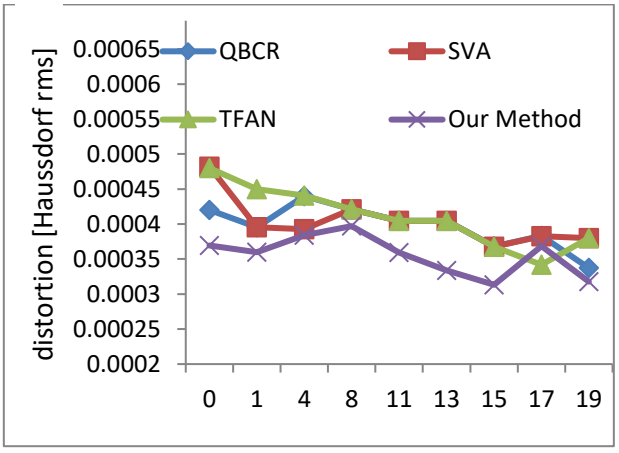


Figure 24 Distortion (130 cm 1 Kinect)

This result implies a speedup of over 100% compared to the state of the art MPEG 3DGraphics encoder at only a slightly lower compression gain. Most gain is achieved in compressing the connectivity data, of which in most cases over 90% is encoded in runs. Figure 27 to Figure 29 compare the different possible setups and encoding solutions. In this case comparing setups with 5 Kinects and 1 Kinect at both high and low quality at a bit longer distance (300 cm), representing a more console like or

living room like experience. The high quality 5 Kinect representation is the most challenging, as the frames consist of about 253,000 vertices and 487,500 faces on average. My heuristic is able to process such a frame into a 1 MB block in on average 160ms. Further parallelization would allow transportation of such highly realistic captured realistic representations in real-time. The scheme heavily outperforms other methods on the speed requirement that is critical for the application.

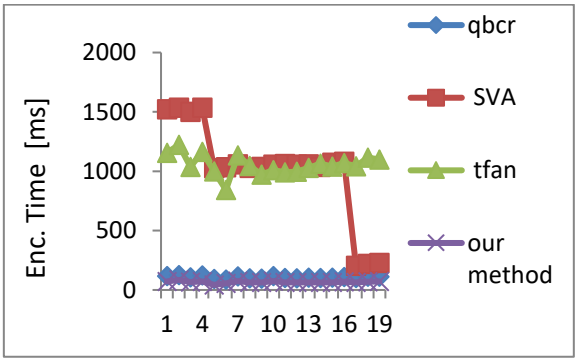


Figure 25 Encoding time with different methods [ms] (130 cm one kinect)

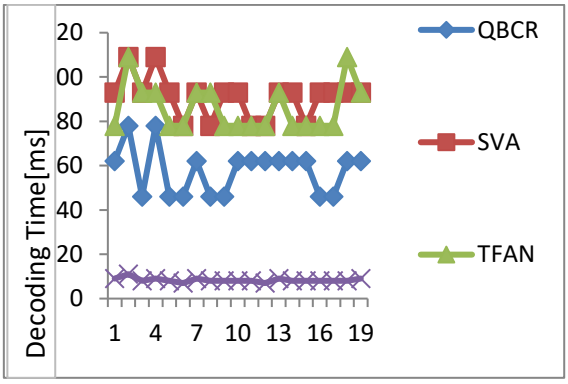


Figure 26 Decoding time with different methods [ms] (130 cm one Kinect)

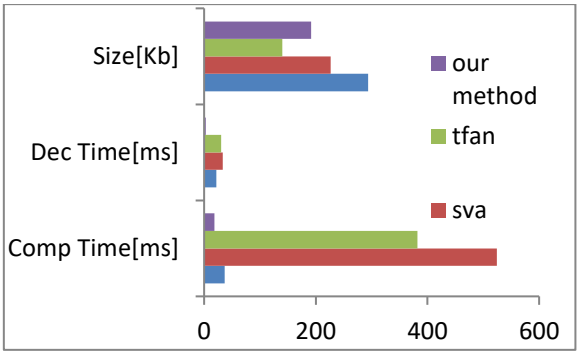


Figure 27 Low Res (~253K vertices) Data with 5 Kinects (300 cm)

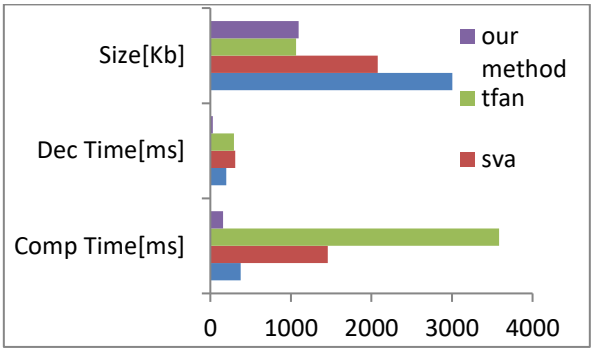


Figure 28 High Res (~72K vertices) Data with 5 Kinects (300 cm) (size in kb time in ms)

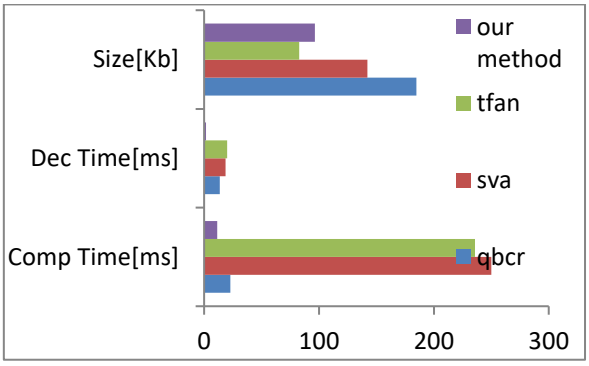


Figure 29 Low Res Data (17K vertices) with One Kinect (300cm)

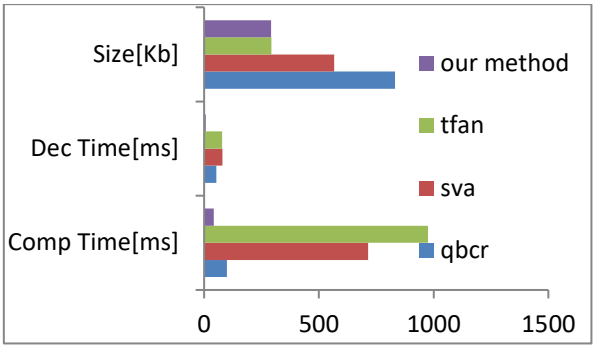


Figure 30 High Res (~70K vertices) data with 1 Kinect (300 cm)

3.7 3D Packetisation

Generally, interactive communication over lossy networks has been tackled with the possibility of omitting information at the receiver. Modern video codecs implement the possibility of decoding at reduced resolution or frame rate, should not all the information arrive at destination within a target end-to-end delay. Units that can be dropped are generally small and have poor impact on the continuity of the service. Triangle mesh compression has not been designed to be resilient to information losses. Thus the loss of a packet can waste a lot of resources since:

1. The packet needs to be retransmitted to make sure the mesh can be decoded, yielding to uncontrollable delays (e.g., TCP)
2. If the frame is skipped, bandwidth is wasted for information that is not decoded.

In the tele-immersive system, a rateless code was integrated, to achieve minimal end-to-end delay and protection against packet losses. In this section the concept of rateless coding is introduced and compared to resilient transmission via TCP, based on a number of real experiments. It shows its favourable properties for geometry transmission and 3D tele-immersion. This module was contributed by Queen Mary University of London, and it is presented here briefly on a joint work in [73].

3.7.1 Rateless Coding

Random Linear Coding aims to achieve packet loss protection with near optimal rate and quick adaptation to the network conditions. The idea of rateless and fountain

codes is that any amount of packets can be generated at the sender. The first practical Random linear codes were first proposed in [74]. Currently, codes like Raptor [75] and RaptorQ [76] have been proposed as standards by IETF. The benefits include linear encoding and decoding time of the data (compared to quadratic time in Reed Solomon codes). The codes are called rateless, as the amount of data generated is not fixed, in case of increased packet loss in the network, the data generated can be increased for extra protection. This constitutes one of the main advantages compared to traditional fixed rate FEC codes such as Reed Solomon codes. Additionally rateless codes also reduce the end-to-end delay, because they do not need retransmission of information. The receiver then only has to receive a set of packets to make sure the reconstruction of the frame is possible. A symbol based version of rateless codes, more similar to our proposed technique, has been also adopted in the field of network coding [77] [78] to allow receivers to decode from packets recursively encoded by different nodes.

3.7.2 Implementation

The data stream is divided in units that are encoded together (similar to NAL units in the H.264/AVC standard), e.g., frames containing the triangular mesh at a certain instant. The implementation further divides these segments in generations and data blocks (See Figure 31). Further, it adopts packet-based random linear coding on a Galois Field (GF). Blocks belonging to a generation are always meant to be coded with blocks from the same generation.

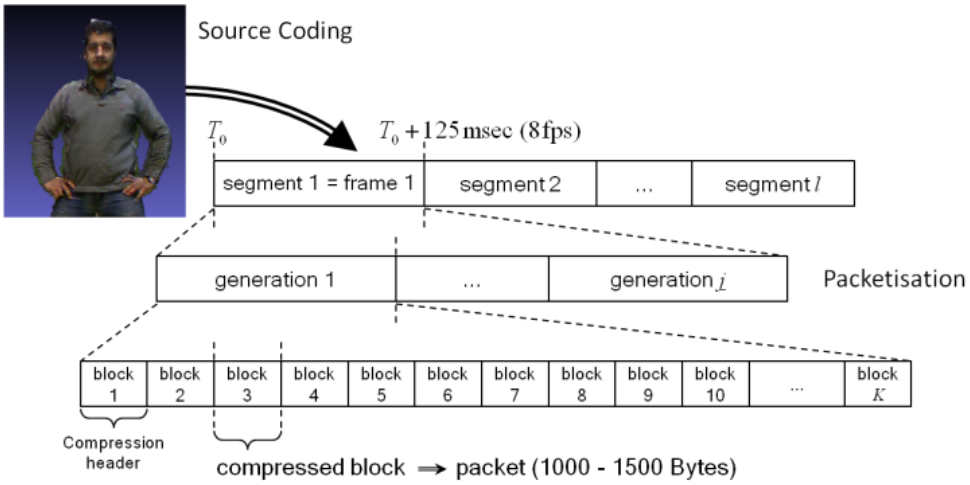


Figure 31: Arrangement of blocks for rateless coding

Each block is a sequence of code words, each code word made of m bits each (typically 8 bits, or a multiple of 8 bits), so that encoding and decoding operations are performed in an algebra over a Galois Field (GF) of size 2^m. A new packet is generated by linearly combining the K source blocks of the current generation with random coefficients$c_1, c_2, \dots, c_K$. A codeword $b_{new,j}$ from a new coded block $\boldsymbol{b}_{new}^{(g)} = b_{new,1}, \dots, b_{new,\ Nw}$, of generation g can be expressed as:

$$b_{new,j} = \sum_{i=1}^{K} c_i \, b_{i,j}^{(g)}, \qquad j = 1, 2, \dots, N_w \tag{3.1}$$

where $b_{i,j}^{(g)}$ is the j-th codeword of the i-th block in generation g.

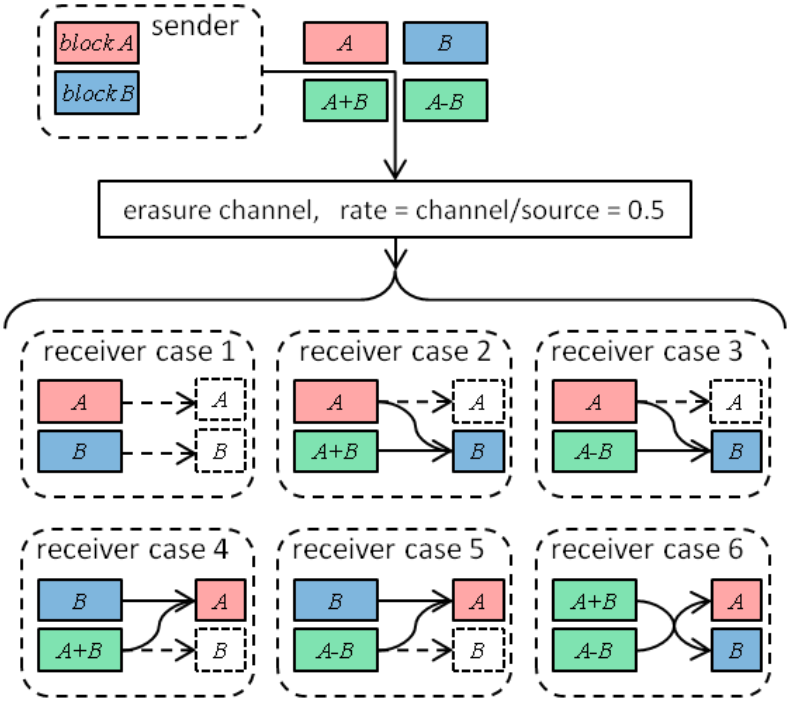


Figure 32: Example of rateless coding and decoding from linearly independent subsets of packets

The coefficients of the linear combination are embedded in the packet header, to make sure the receiver knows which specific linear combination has been received. Decoding operations are also performed between blocks labelled with the same generation. As soon as enough packets are received for a generation, the coefficients are used to build a $K \times K$ linear system that allows decoding and recovering the data. Figure 32 shows a simple example of how packets randomly-coded from 2 two source blocks are decoded from any linearly independent subset of blocks. In order to reduce the decoding computational load, the method constructs a composite matrix of data, and coefficients of the incoming packets and perform Gaussian elimination each time a new packet is received. This spreads the computational cost over time and drastically reduces the decoding complexity. Such complexity can be still reduced by reducing the dimension of the coding space. These factors need to be properly considered:

1. Loss protection (Larger coding diversity).
2. Decoding complexity (Smaller linear system).

Properly balancing the coding space between big linear systems (more coding diversity, more decoding complexity) and small systems (less coding diversity, less complexity) allows achieving optimal delay performance and the required resilience against packet losses.

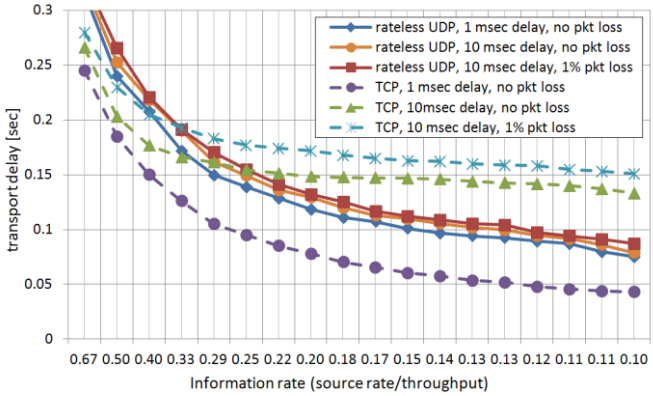


Figure 33: Transmission delay of TCP and rateless coding varying depending on the available channel rate.

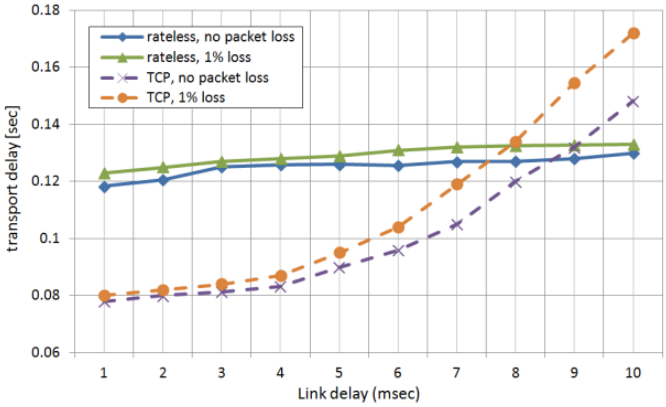


Figure 34: Transmission delay of TCP and rateless coding in seconds, due to network delays.

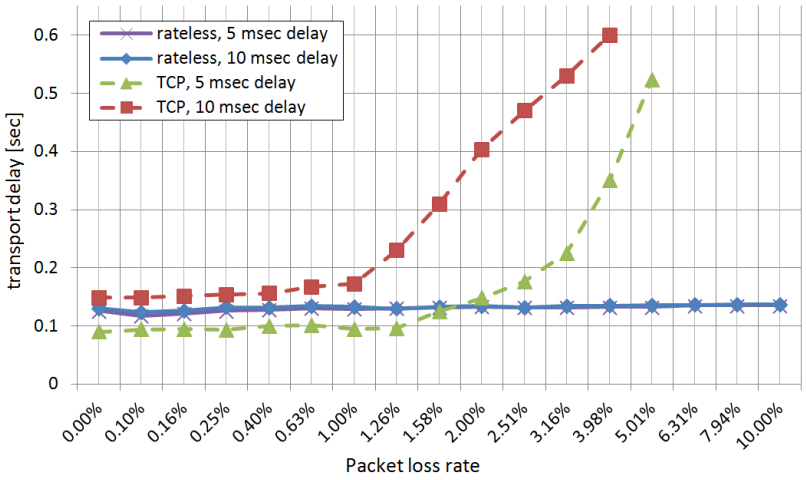


Figure 35: Transmission delay of TCP and rateless coding in seconds, due to packet losses.

3.7.3 Experimental Results

In order to assess the delay performance of the rateless coding system some experiment on an experimental setup composed by two Intel Core i5 machines (3.10 GHz) have been performed: one in charge of capturing, encoding and transmission and one receiving from the network and decoding the source data; a network emulator that reproduces a large variety of network conditions in terms of delay, bandwidth and packet loss rate is run in the receiving machine. Our rateless transmission system makes use of UDP packets and is compared with a standard self-managed and reliable TCP connection. The factor influencing the efficiency of our rateless transmission is the ratio between throughput and source rate, given a certain packet loss rate. This should always be able to sustain the source information rate. It has been shown that the delay performance relative to a set of experiments with limited bandwidth, variable delay, and packet loss rate. Delays and packet losses in the network affect only linearly the delay performance of the rateless decoding, as opposed to TCP that needs to engage mechanisms of recovery every time a packet is lost. The mechanisms of recovery are further affected by the link delay. We analyse how the delay introduced by the transmission and channel coding and decoding and the way this is affected by the network conditions. Figure 33 shows the introduced delay and the

influence of the available bandwidth, with different packet loss and delay conditions. Rateless coding works best when extra bandwidth is available, in order to cope with some additional overhead introduced by the extra packets. Figure 34 shows the delay performance depending on the latency of the network, whereas Figure 35 shows again the transmission delay and its sensitivity to packet losses. Although in some ideal conditions TCP reaches optimal transmission performance, it suffers large transmission delay in the case of network delays and packet losses, making it unsuitable in realistic networking conditions. In the rateless system, the influence of network impairments is linear and controllable. The rateless code achieves good delay performance in different network conditions.

3.8 3D Tele-immersive Systems Integration and End-to-end Performance

3.8.1 3D Triangle Capturing and Rendering

The capturing component was provided by the Centre for Research and Technology Hellas. It creates reconstructions in real-time (8-10fps) of humans using range images (RGB plus depth) captured with multiple Kinects. The system is based on merging tessellated depth images using either zippering or volumetric method, which are the most common methods to reconstruct a triangle mesh from multiple depth images. The render component renders meshes in real time with shading based on global illumination effect using the normal data in the vertices. This component was implemented with OpenGL and QT and provided by Institut Telecom, Paris. These components have been linked together to construct the tele-immersive prototype.

3.8.2 Media Pipeline Performance

We tested the computational performance of the media pipeline in two different ways. Table 8 shows the results when running the sender and receiver on one machine (a 1.6 GHz Intel i7 laptop, 4GB Ram). First, captured meshes with one depth camera, reconstructed and sent over the local interface back and rendered. In the process, we recorded the time taken by capturing, encoding, decoding and the rate at

which frames were sent and received. The update frequency (refresh rate) of the renderer to the screen was also tested. This illustrates achievable frame rate in the pipeline.

Table 8 Achieved results in tele-imersive pipeline

Sub-part	Average [ms]	Std [ms]
Capturing	94 ms	5,3 ms
Encoding	52 ms	3,5 ms
Decoding	11 ms	1,1 ms
Rendering	46 ms	4 ms
Send-Rate	5-8 fps	
Recv-Rate	5-8 fps	

The graphs in Figure 37 show the global end-to-end delay of the system from capturing to rendering, using both the traditional transmission over TCP and our rateless coding system. In this case two Intel Core i5 machines (3.10 GHz) PC's are connected (sender and receiver) and the network impairment simulator is used to generate the networking conditions between the machines (connected in the LAN). The delay over the link was 10ms in all experiments, while data could be medium or high resolution. The medium resolution frames contained around 16-18k vertices (90-100 Kbytes per compressed frame). The high resolution contained around 50k vertices, (280-300 Kbytes compressed per frame). We performed different tests, with 0% and 1% packet loss. Figure 17 and 18 show that the system still enables real-time communication in the case of packet loss/delay. The system also allows reconstruction and reliable transmission without retransmission. By employing rateless coding over UDP and fast mesh compression we avoid exceeding the target end-to-end delay for interactive applications of about 300 milliseconds including live capturing and 3D rendering.

Figure 36 Some screenshots of the rendered output

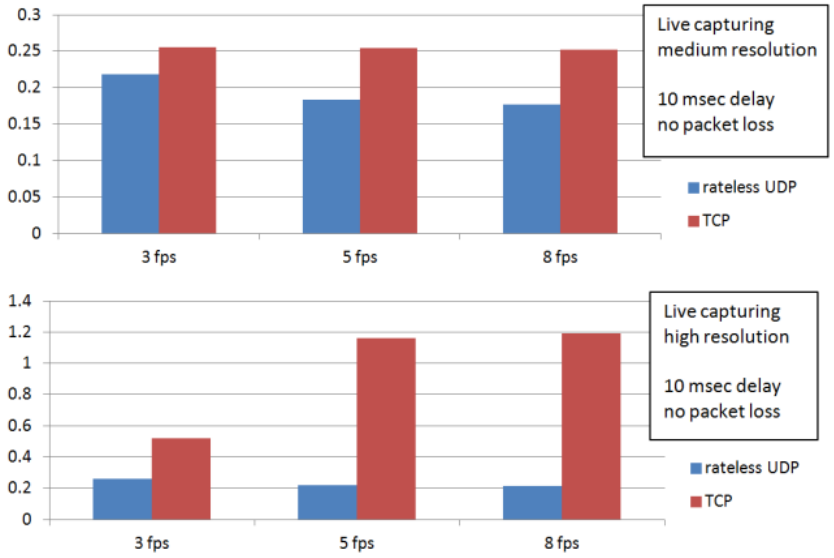


Figure 37 End-to-end delay with no packet loss

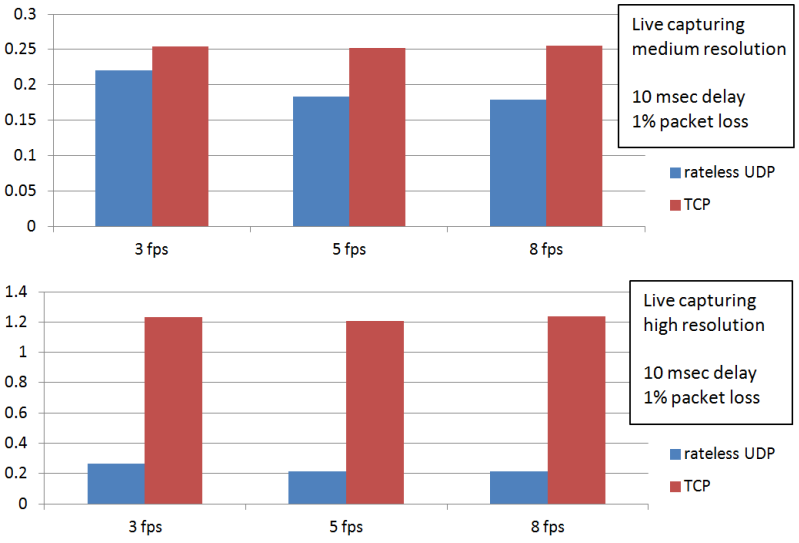


Figure 38 End-to-end delay with 1% packet loss

3.9 Conclusion and Discussion

This chapter presented a prototype implementation that enables conferencing between remote participants, focusing on the compression and the transmission components. The novelty is that our 3D tele-immersion system is based on triangle mesh representations, unlike previous solutions. The triangle mesh representation has traditionally been supported by computer graphics in virtual worlds and games. Real-Time streaming of captured mesh data, therefore, will enable novel applications that can integrate real and virtual worlds. The prototype addressed some of the significant challenges that triangle mesh representation poses to the media streaming pipeline in terms of latency, data-volume and robustness to losses.

The contributions can be summarized as follows:

*Encoding/Decoding***:** triangle mesh codecs have not been designed with the interactive scenario in mind. The encoding time is simply often too long. We developed a local method that takes advantage of specific properties resulting in a speed increase of ten times when compared to the TFAN encoder from MPEG, at comparable quality/rate. The method works well with the meshes produced by our capturing system. It takes advantage of the coherence property of the captured mesh. This property was also found to be present in reconstructed and captured meshes. Also, the relatively simple operations of this method and block separation allow for fast parallel or hardware implementations.

Streaming: systems that efficiently transmit geometry in real-time, generally dealt with stored objects instead of captured reconstructions. Such middleware solutions, and application-layer protocols, facilitate efficient real-time downloading of a mesh. This is achieved by an offline optimization step, which cannot be performed in geometry-based 3D tele-immersion systems. We do not develop a specific middleware or protocol but introduce an implementation of a rateless code (inter-packet FEC) that protects each mesh frame against packet losses. Rateless code allows any given number of extra packets to be generated, depending on the amount of protection that is needed. Rateless codes, like Raptor and its successor RaptorQ, have both been proposed as standard by IETF [76] [75] in 2007 and 2012, but have seldom been

tried for real-time streaming of geometry-based data. Our experiments show the favourable properties of the rateless code, such as low end-to-end delay compared to TCP in case of packet loss of over 2%. Moreover, the distributed implementation of the packet decoder introduced a modest delay of 50 ms. In addition, the streaming method is generic, video audio and other data can also be efficiently transported in real-time based on this code.

3D Tele-immersion: this chapter presents a prototype of a triangle mesh based 3D tele-immersion system. This prototype is integrated with state of the art capturing and rendering components. None of the previously presented 3DTI systems can capture and stream full 3D reconstructions in real-time. On top of that, state of the art rendering techniques can be applied such as global illumination. The focus in this chapter was on the visual media pipeline, but further integration with spatial audio techniques such as binaural hearing are planned in the near future. A next integration step will include merging the received 3D mesh live into a virtual world adding spatial audio.

Nevertheless, this thesis still wants to achieve higher quality results which are hard to achieve with the current compression method. In the next chapter, we will therefore focus on developing a codec with a higher precision of the coordinates and attributes to enable a higher quality transmission compared to Figure 36 with a connectivity driven approach.

Chapter 4 3D Tele-immersion with Connectivity Driven 3D Mesh Compression with Late Differential Quantization

The previous chapter presented an initial prototype for 3D tele-immersion with support for compression and transmission of highly realistic 3D human meshes. This chapter continues to work on the same line (research question 2). This chapter aims to reduce some of the drawbacks of the previously presented approach. The two main drawbacks are: low precision (precision compared to 8-bit quantization was achieved which could result in blocky appearance) and strong dependence on the capturing system. Therefore, this section introduces a novel approach for connectivity driven mesh coding that gives better precision and is more generically applicable. In addition it introduces further improvements in the end-to-end pipeline. It integrates progressive packet decoding to the FEC scheme and optimizes the inter thread data exchanges based on Last in First Out (LIFO) scheduling policy to deal with varying frame rates and sizes. These advances are tested in network conditions corresponding to MAN/WAN like scenarios such as those that can be used in a Virtual private network distributed between locations (VPN). In addition, a more advanced rendering method with composite rendering has been integrated, resulting in highly realistic composite rendering with 3D graphics content. This work addresses the second research question:

Research Question 2: How can we achieve Real-time (sub 300 ms motion to photon) compression and transmission of highly realistic reconstructed 3D humans based on meshes with State of Art Compression rates (i.e. MPEG-4)?

In a more generic way compared to the approach presented in chapter 3.

The work in this chapter is based on the following publications:

Mekuria R.N., Sanna M., Izquierdo E., Bulterman D.C.A., Cesar P. "Enabling 3D Tele-Immersion with Live Reconstructed Mesh Geometry with Fast Mesh Compression and Linear Rateless Coding" in *IEEE Transactions on Multimedia (16,7) pp 1809 -1820* (codec, rateless packet decoder, rendering integration)

Mekuria R.N., Cesar P., Bulterman D.C.A. "Low Complexity Connectivity Driven Dynamic Geometry Compression for 3D Tele-Immersion" in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* , (IEEE ICASSP 2014), Florence, Italy, May 4-9, 2014 (codec implementation)

As this chapter follows the same research question as the previous, it immediately present the novel connectivity driven codec in the next section.

4.1 Low Complexity Connectivity Driven Mesh Compression

This chapter considers 3D mesh sequences reconstructed on-the-fly, as a temporally incoherent mesh sequence MS. MS = $M^i = (V^i, F^i), i = 0...n$ is a reconstructed mesh sequence and the number of vertices in mesh I is $|V^i|$ and the number of faces is $|F^i|$. Both are not constant over i. The codec aims to compress this data with a rate-distortion comparable to available methods in MPEG-4 [16], but with a lower computational complexity resulting in real-time encoding.

Figure 39 outlines the compression system. It introduces a fast connectivity traversal method combined with connectivity-driven (offline) optimized DPCM coding and layered quantization. Further, it explains the rationale and operations for compressing the connectivity $C(E)$ in the next sub-section. The differential encoding of the geometry $G(V)$ followed by non-linear quantization is presented in 4.1.2. Section 4.1.3 discusses how the appearance (normals(V) and colours(V)) are handled, we call this appearance quantization.

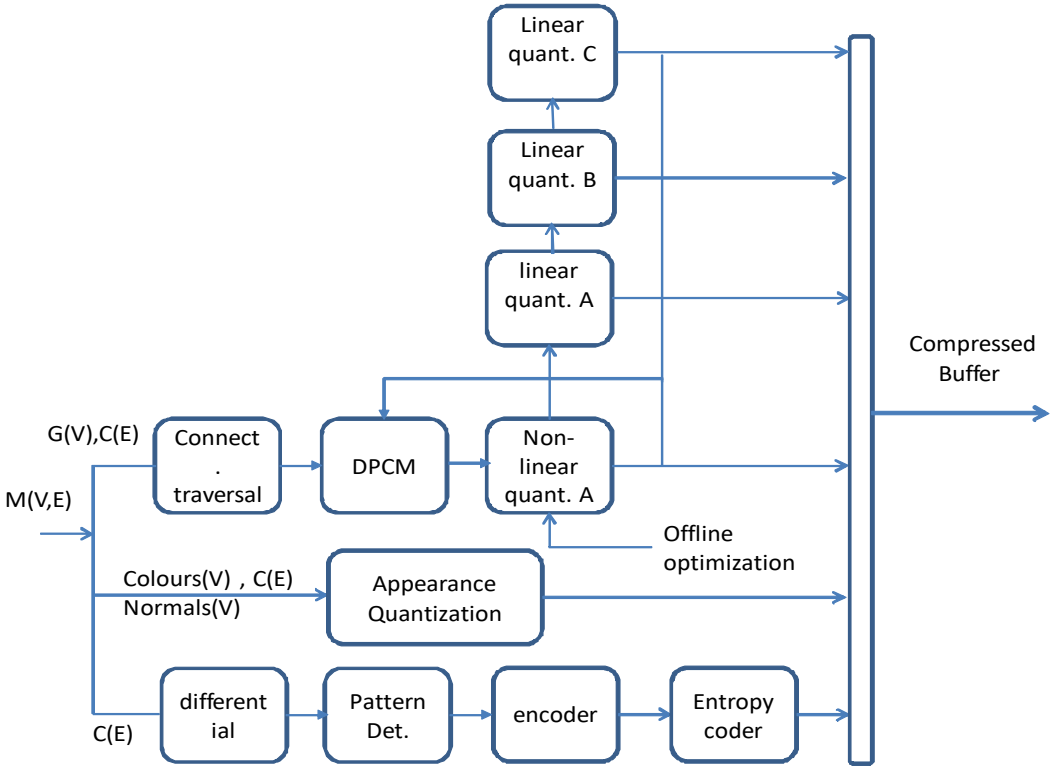


Figure 39 Schematics of proposed dynamic mesh codec

4.1.1 Pattern Based Connectivity Coding

The idea behind the connectivity coding approach presented is that 3D reconstruction can introduce specific regularities in the connectivity information that can be exploited for compression purposes. For example, in the zippering method in [79] multiple range images are tessellated first into range surfaces that are subsequently zippered (stitched) together (after redundant triangles are removed). If these range surfaces are tessellated in a consistent order, this can introduce a more regular and predictable connectivity structure. Such patterns were also found in the connectivity of the reconstruction data in [15]. As such, patterns occur many times in a row, the method actively searches for them and uses them for efficient encoding. An example

of how following differences occur is shown in Table 9, where each next index is an increment of 1. While such patterns might differ per reconstruction method and need to be found before they can be exploited, it is very beneficial to enable low-complexity encoding of large meshes.

Figure 40 illustrates the connectivity compression scheme. First, the entire connectivity information is searched for repeated regularities which are counted and stored in the data-structure pattern run shown in Table 9. The mode field represents the type of pattern, the diff field the two differences that occur and the count field signals the number of repetitions. The start field is used to reference the position of the first index in the connectivity list. This value is used to detect the start of a run in the next encoding step and in the decoder. Next, all indices are iterated again. If an index is the start of a run, the pattern run (indicated by the start field), is stored and the connectivity index iterator is increased with the count field. If an index is not stored in a run, the difference with the index in the previous face is stored instead. Storing differences instead of absolute values yields mostly small numbers skewed around 0 and 1 and therefore allows efficient entropy coding. The resulting data vector is entropy encoded via the zlib library [80].

Table 9 repetitive pattern in mesh connectivity

1632 1520 1633
1520 1521 1633
1633 1521 1634
1521 1522 1634

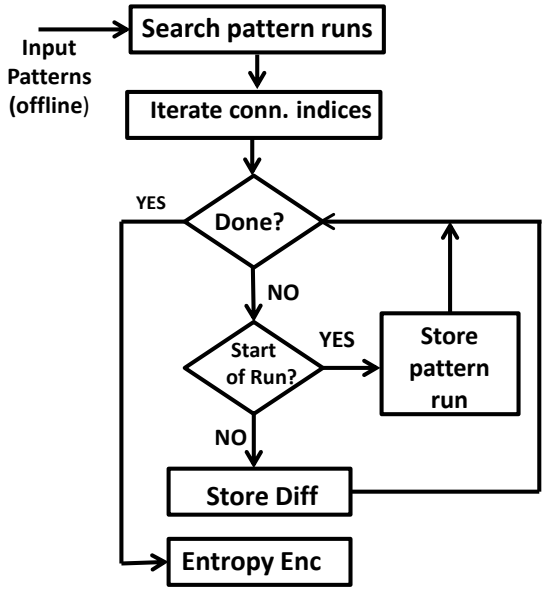


Figure 40 Pattern Based Connectivity Coding

Table 10 Structure Pattern Run

Mode	Diff1	Diff2	Start	Count

4.1.2 Geometry coding with delayed differential encoding

Uniform quantization is used for geometry compression in MPEG-4 SC3DMC [16] and other methods, which have been designed with high quality graphical models with s large number of quantization bits (10-20). For the low-bit rate requirements for streaming live reconstructed 3D geometry that are densely sampled this method however poses several disadvantages.

Firstly, the dynamic range of the vertices that are quantized in a 3D space is large compared to the details in the 3D surface that becomes visible to the viewer. This makes details in the surface vulnerable to quantization distortion that increases with coarse quantization parameters. When coarse quantization is used for geometry compression, the lack of detail in the mesh surface immediately becomes visible and

vertices may share positions resulting in degenerate triangles (triangles with a zero surface) and blocky appearance of the surface arises.

Therefore, instead we propose quantization after the transform (DPCM) with a variable number of bits. In this proces a layered structure is proposed: most vertices are quantized with 4 bits, but values outside the 4-bits range are quantized with 8 bits, 16-bits and 32-bits respectively when needed, i.e., if the differences become large. This way, large errors in the geometry are avoided, as even for a small amount of vertices large quantization errors can become clearly visible when the mesh is rendered.

Next, we code the differences between vertices that are connected to each other. In this case the differences are even more fine-grained as by the nature of densely sampled 3D reconstruction, connected vertices are closely co-located. The primary quantization vector codes over 95% of the values and is trained for the given datasets and set as follows:

$$\begin{bmatrix} -0.011 - 0.0063 - 0.0042 - 0.00315, -0.00236 \\ -0.0012 - 0.0004\ 0\ \ 0.0004\ 0.0012\ 0.00236 \\ 0.00315\ \ 0.0042\ 0.0063\ 0.011 \end{bmatrix}$$

This configuration for the 4 bits non-linear quantization vector was computed offline. Typically this vector can be dependent on the reconstruction method and we envision that such information can be exchanged in a 3D Tele-immersion system out of band (in our 3D Tele-immersion system a custom session management system is used that can exchange such information). Subsequently, the method utilizes a layered structure with 8-bits for the range from 0.11 to 0.1, and 16 bits between 0 and 2 to cover the entire dynamic range found in the training data (the value of the primary quantization vector is used for sign indication).

To perform connectivity driven differential encoding, a traversal method that does not change the order of the vertices in the list was designed and implemented. Each of the faces is traversed, and the first connected vertex that was previously stored (set) is used for differential prediction. If none of the connected vertices have been previously traversed and set, we set the first vertex index in the face by adding this

value to the reserve_vals list. The other connected vertices in the face are then differentially encoded. At the decoder, the method traverses the connectivity in a similar manner, unset vertices are loaded from the reserve_vals list and the other vertices are recovered via inverse differential prediction. This method works well as the reconstructed meshes are relatively coherent (connected vertices are also closely located in the indexed face list). In practice less than 0.1% of the vertices are stored in the reserve list in some of the reconstruction systems we have tested. Values in the range $[-0.011\ ,\ 0.011]$ are stored and appended in the 4-bits vector, values between $[-0.1 - 0.1]$ are quantized with 8 bits and appended to the eight bits queue vector. Larger difference values are stored using the 16-bits or 32 bits quantizer but occur very rarely but need to be stored accurately as they introduce large visual distortion when decoded incorrectly.

4.1.3 Appearance Quantization

In the case of surface mesh geometry, the perceived appearance depends not only on the colour but also on the geometry coordinates and surface normal data. We deploy the same system of late differential quantization for the appearance data (normal and colours), however the layers are assigned differently. We quantize normal components in the range from -0.25 – 0.25 with 4 bits and the rest with eight bits (8 bits plus the sign bit retrieved from the 4 bits quantization vector). The method codes colour differences between -25 and 25 with 4 bits and the rest with 8 bits (precisely 8bits plus the sign bit retrieved from the 4 bits quantization value covering the entire range$[-255,255]$. Again, such information can be communicated out of band prior to the media streaming session to configure the encoder and decoder properly.

4.1.4 Evaluation

For evaluation of the compression method we use datasets of meshes reconstructed with five depth cameras that are currently used in the 3D Graphics group of Moving Picture Experts Group (MPEG) for evaluation of 3D Tele-immersive Mesh coding technologies. They have been created by the Center for Research and Technology Hellas (CERTH) based on the method reported in [81] and contain participants performing different activities. The datasets are publicly available at the website currently hosted at [82]. These datasets represent the case where a user is in a room

captured by multiple depth cameras at a distance of 300 cm. We have also performed several test sets with data reconstructed with one depth-camera representing a user that is sitting in front of his computer (at around 1 meter).

Two metrics are used to assess the geometric quality of the mesh the *symmetrical Haussdorf distance* and similarly the *symmetrical root mean squared distance,* computed between the original and decoded surface. The symmetrical Haussdorf distance is defined in equation (6) as:

$$d_{sym}(\mathrm{M}, \mathrm{M}') = \max[d(M, M'), d(M', M)] \qquad (4.1)$$

Where the M is the original and M' the decoded mesh and d(M,M') is the Haussdorf distance between the two surfaces. Similarly the symmetrical root mean square error (RMS) defined where instead d(M,M') is the root mean square distance between the two surfaces. To facilitate the computation of these metrics the tool developed in [83] was used (see also for additional details on the used metrics). Alternatively, the quality of the colours and normal (appearance) and the normal with root mean square error on a per vertex basis were compared.

$$d_{app} = \sqrt{\frac{1}{|M|} \sum_{p \in M, p' \in M'} \|p - p'\|_2^2} \qquad (4.2)$$

Where p and p' denote the 3 coordinate colour/normal in original mesh M and decoded mesh M' respectively. As the MPEG TFAN codec does not preserve the order of the vertices, we compared to the SVA codec at the same quantization parameter instead. Also, we compare against the quantization error introduced by quantization of a uniform source with a uniform quantizer in 3D which is given by

$$d_{uniform\_quantization\_uniform\_source\_rms} = \sqrt{3}\Delta/\sqrt{12} \qquad (4.3)$$

Where Δ is the quantization step size. Last, the encoding and decoding times in the proposed codec will be compared and we will provide screenshots of the original and decoded meshes.

4.1.5 Comparative Results

The scheme was implemented in C++ and the test machine was a desktop machine with Intel i7 CPU and 8,00 GB or RAM and a Geforce GTX 760 video card. The

MPEG SC3DMC Codecs were compiled from source code with the same compiler as available on [84]. To avoid overhead that deals with loading the supported text formats in MPEG part-25 [85], we interfaced directly the class SC3DMCEncoder and SC3DMCDecoder in the reference software with an MPEG indexed face set structure directly loaded from the input mesh. All files are first loaded into memory before the compression routine is started. The run times are recorded with CPU wall clock times provided by the boost C++ library with a resolution around 366 Nano seconds.

We ran all methods on N=158 Meshes with 5 depth cameras with an average of 302K Vertices. Figure 4 shows that a large speedup is achieved compared to other methods. In terms of compression size, it achieved an average a mesh size of 1,460 KiloBytes compared to 1266 KiloBytes with TFAN MPEG with 10 bits for quantization of coordinates (per direction) and 6 bits for colours and normal (per component). This is a 15% lager file size compared to TFAN, however a speedup of almost 10 times was achieved. The result is similar in case the meshes are reconstructed with 1 depth camera (average of 72K vertices), but the proposed method encodes meshes in as little as 35 ms on average.

Table 11 Encoder parameter settings for evaluation

TFAN-8-6-6	MPEG-4 TFAN with 8 bit per coordinate component and 6 bit per normal/colour component
TFAN-10-6-6	MPEG-4 TFAN with 10 bit per coordinate component and 6 bit per normal/colour component
SVA-8-6-6	MPEG-4 SVA with 8 bit per coordinate component and 6 bit per normal/colour component
SVA 10-6-6	MPEG-4 SVA with 10 bit per coordinate component and 6 bit per normal/colour component
Our method	The method developed in this chapter with the pre-trained quantization vectors

Evaluation of the symmetric quality metrics shows that comparable quality of geometry is achieved (Figure 5). The jump in the right side of the graphs in Figure 5 represents the fact that more densely sampled meshes are tested there (more vertices and visual detail). The results with our method are still closely clustered around 0,0002 and the quality is approximately equal to 10 bits quantized MPEG compression (a bit worse with sparser meshes and a bit better with denser meshes). The method achieves higher quality decoded models when compared to 8 bits quantization.

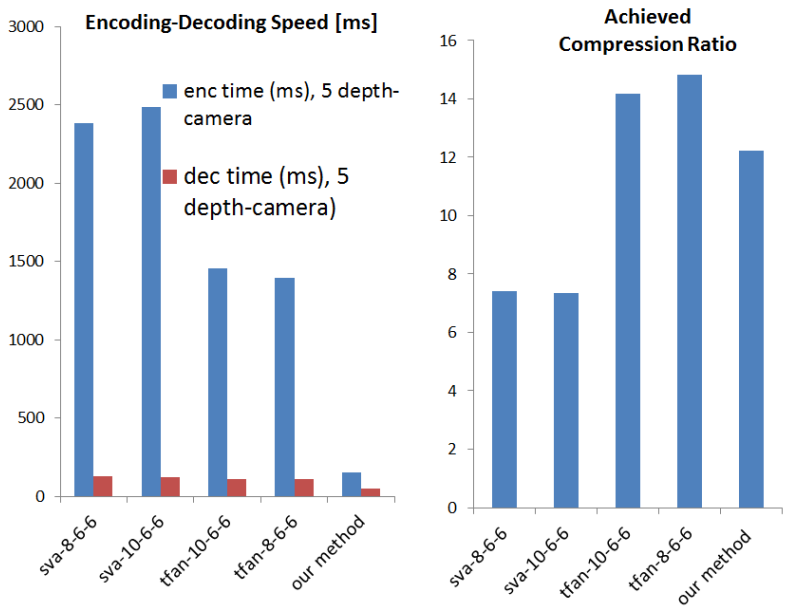


Figure 41 Encoding and decoding results with 1 camera

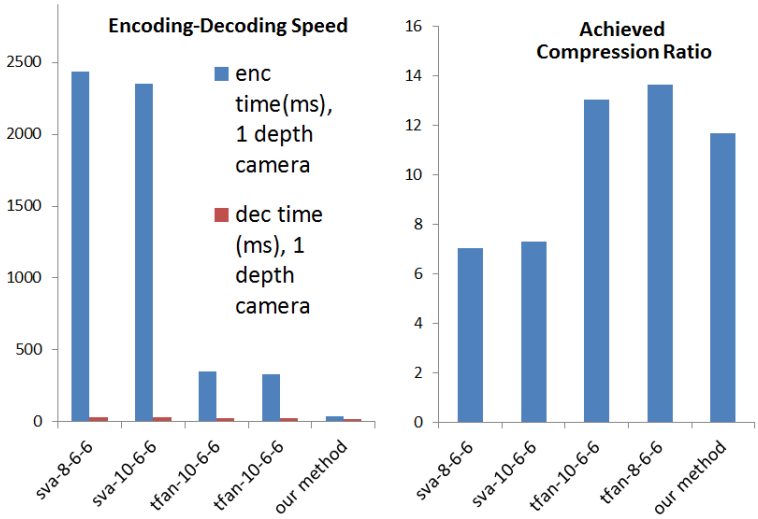


Figure 42 Mean symmetric Haussdorf distance between the original and decoded model

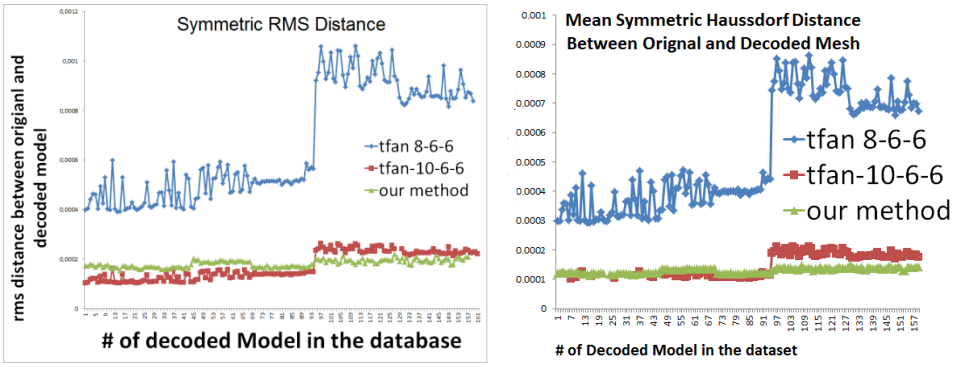


Figure 43 RMS error of colour and normal between original and decoded reconstructed meshes

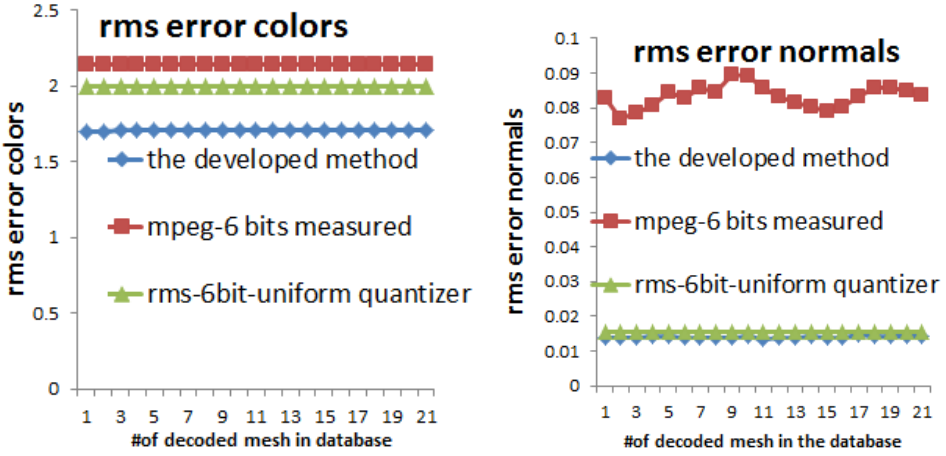


Figure 44 RMS distortion colour and normal attributes

Figure 45 Decoded meshes, the original (left), mpeg (tfan) with QP10 (middle), our method (right)

We compared the quality of the colour and normal data with the root mean square difference on a per vertex basis. As TFAN re-orders vertices we compared only with SVA in Figure 44 and $\sqrt{3}\Delta/\sqrt{12}$ which is the root mean square quantization error introduced when quantizing a uniformly distributed source uniformly in 3D with step

size Δ.The results show that our method achieves slightly lower distortion of the normal data and the colours compared to the theoretical and the measured values. Figure 45 shows snapshots of the decoded meshes, most artefacts are related to the 3D reconstruction method and not the compression method.

4.2 Updates on Packetisation

Some updates to the packetisation scheme have been contributed by Queen Mary University of London. The details can be found in section V of our joint work in [86]. The main advantages of the updates is that the packet coding and decoding can happen progressively, i.e. form partially received information, already scrambled packets can be generated. This improves the end-to-end pipeline performance and makes the packet coding less of a bottle neck. The impact of this advance will be come clear in the end-to-end evaluation.

4.3 Integrated Pipeline Performance

The developed compression and transmission components have been integrated with a 3D reconstruction system and a modular rendering engine to test the complete end-to-end system performance. Two separate modules were developed: the acquisition module for the acquisition, compression and transmission and the reception module for packet reception, decoding and real-time rendering (see Figure 46). The rendering engine provides the illumination and shading techniques and manages the composition into the scene. The end-to-end streaming performance is tested in a lab environment that consists of two computers. The first running the acquisition module, with an Intel i7 2.8 GHz CPU and 8 GB of RAM plus Microsoft Kinect depth camera. The second, with an Intel i7 3.4 GHz CPU with 16 GB of RAM, is running the reception module. At the side of the reception module, a network emulator [87] is setup that introduces random network impairments such as delay, jitter and packet loss. To measure overall performance, we deployed a monitoring system integrated in the software that measures synchronized timestamps in each step of the transmission pipeline. It monitors each event (i.e. the time a frame was captured, when a frame was compressed, scheduled, rendered etc.). This way we assess the real-time streaming performance of the entire mesh based 3DTI System pipeline from acquisition to rendering. We matched the network impairment introduced by the emulator to the levels presented in ITU G.1050E [88] which are values widely considered in industry. Table 12 shows the parameters for well-managed IP connections (class A based as based on leased line) in the first and second column. In the third and fourth

column we show established characteristics for partially managed networks (Class B) (this is the class of network that provides QoS based on separate router queues only like DiffServ). For our evaluation, our attention focusses on latency and jitter values. Table 12 shows that in case of TCP large frame sizes cause large delays. We measured the end-to-end frame delay from the capture instant to the instant when the mesh is remotely rendered. The frame capture and reconstruction time varies from around 30 milliseconds to a maximum 50 milliseconds per frame depending on the number of vertices captured. The reconstruction and compression introduces bit-rates as shown in Table 13. As the rateless coding and UDP transmission (outgoing data rate control) each run in their own thread, it may not keep up with the rate introduced by the capture and compression threads. In such cases a frame will be skipped in the inter-thread exchange of frames. Only the newest frame is exchanged in each stage, and earlier frames are dropped. This avoids delay to build up in the pipeline that can happen for example when the transmission component cannot keep up with the incoming data rate. This policy is implemented in both the acquisition and the reception modules to minimize the user level end-to-end delay. Table 14 presents the time intervals that were assessed. The overall network delay caused by the transmission is the network time (the time the acquisition module uses to send out packets (sender thread) plus data reconstruction time, which is the additional time the reception module uses to incrementally reconstruct the original data from the packets via progressive decoding. The other time intervals are all computational latencies and depend on the system hardware configuration (reconstruction, decoding are all pure computational operations independent of the network). We have synchronized virtual clocks between the machines enabling the application to monitor the time intervals to do the measurements. Figure 47 shows how TCP handles the large mesh frames (this is without rateless encoding) in the represented network conditions. Almost all time (4-10s) is spent waiting on receiving the frame data and as a result, very low throughput and frame-rate are achieved at the receiver. This makes TCP unsuitable for real-time transmission of large media frames that require low delay. Figure 48 shows that the end-to-end delay with the rateless channel coder with progressive decoding is much lower and more predictable. The information rate is set to 75 % (75 % information with 25 percent redundancy, i.e. 33 % overhead) and the source encoding is based on MPEG TFAN in Fig. 16. In this case over 50 % of the total end-to end latency is caused by the TFAN encoder which has become the performance bottleneck in the system. Figure 49 shows the delay performance of the optimized transmission system integrated with both our proposed compression method and the progressive rateless coding scheme. In this case the compression is no longer the bottleneck. Instead the outgoing UDP socket system calls (i.e. network time) sometimes cannot keep up with the incoming data rate from the compression

module introducing an extra network ready wait time which is 13,3 ms on average. In this case the end-to-end delay stays below 300 ms for the large majority of the frames. The frame rates achieved at the remote site are much higher in this configuration. Compared to TFAN as shown in Figure 50, we achieve higher frame rates of approximately 10 fps on average in the 10 ms and 25 ms scenarios and between 8 and 9 fps in the 50 and 100 ms scenarios. On the other hand, the current implementation causes low frame-rates at the receiver site of around 3fps. To summarize we achieved a 3 times increase in the frame rate.

Table 12 Industry accepted network impairments

delay	Regional (A)	IC (A)	Regional (B)	IC (B)
latency(ms)	20-100	90-300 ms	20-100	90-400
jitter(ms)	0-50	0-50	0-150	0-500
packet losses (%)	0-0.05%	0-0.05%	0 to 2%	0 to 2%

Figure 46 Screenshot of rendered 3D person

Table 13 Average bit-rate requirements resulting from compression of 3D frames

	5 fps	8fps	10 fps	12 fps
1 Camera	14.8 Mbit	23 Mbit	29.6 Mbit	35.52 Mbit
5 Camera	58 Mbit	93 Mbit	116 Mbit	134 Mbit

Table 14 Explanation of time interval measurements of the streaming performance

compression time	time to compress the frame
packetisation time	time to generate coded packets
network ready wait	time until network interface is available
network time	time writing UDP datagrams (i.e. sendto calls)
data reconstruction time	time to receive and progressively decode data
mesh decoding time	time for source decoding the mesh
mesh render time	time spend before rendering the mesh

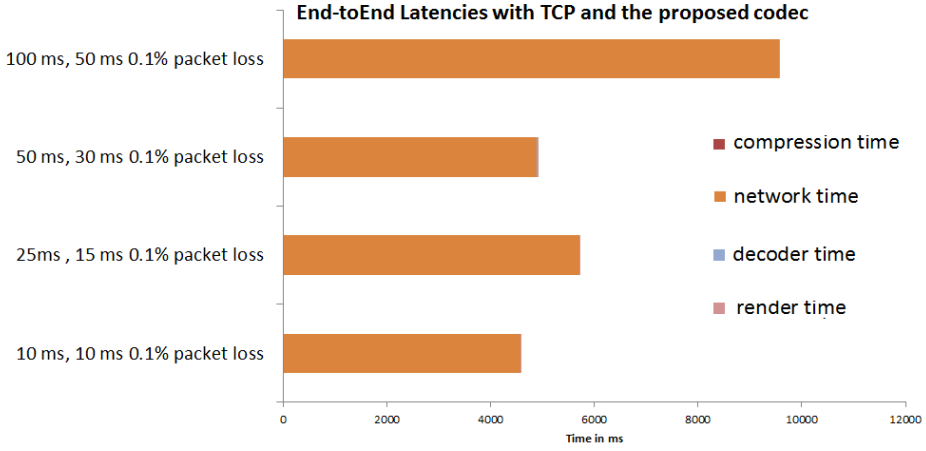


Figure 47 End-to-end delay with TFAN and TCP (capture to render)

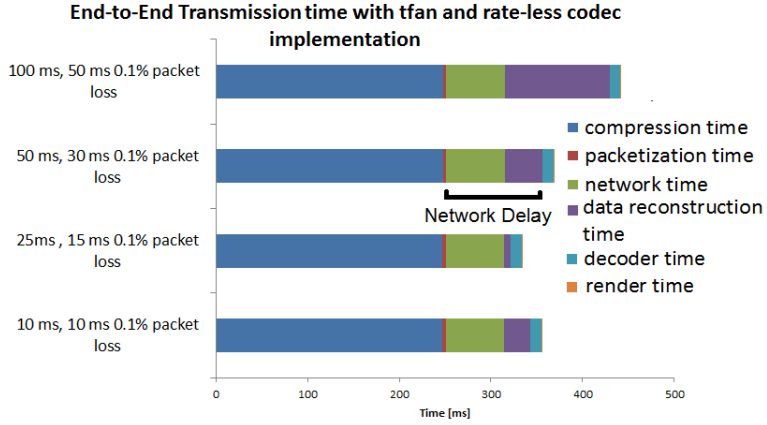


Figure 48 Average end-to-end delay with novel packetisation and MPEG TFAN

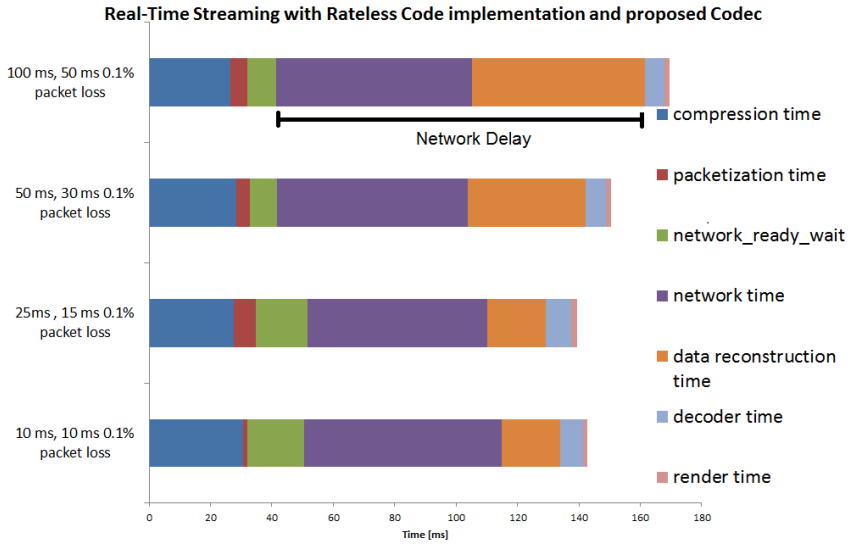


Figure 49 End-to-end delays with new codec and packetisation, emulated network delays in local network

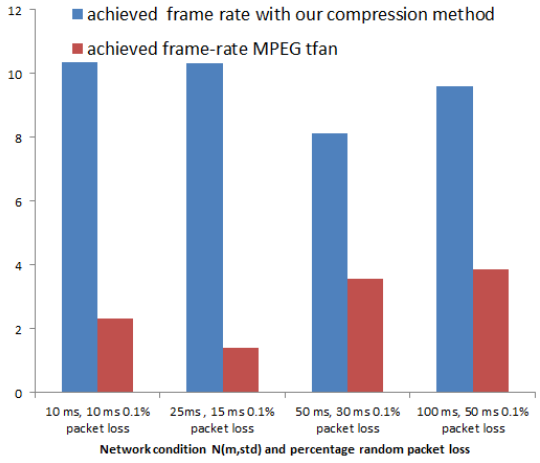


Figure 50 Resulting Frame-rate at the receiver when the sender is capturing at a target rate of 15 in various network conditions, our method gives up to 3x higher frame rates

4.4 Conclusion and Discussion

The improvements for real-time highly realistic coding of 3D frames in this chapter has enabled real-time end-to-end 3D Communication in a working prototype. In this way, my work has outperformed the state of the art in this area by providing a speedup in encoding at comparable bit-rates. This work enabled higher resolution and precision compared to the system developed in the previous chapter. This is all based on techniques for differential connectivity coding plus entropy coding and late differential quantization with pre-trained quantization vectors. In addition, the streaming pipeline has been optimized to handle varying frame rates and frame sizes with a LIFO scheme (last in first out). This way stalls in the end-to-end media pipeline are avoided. The components presented in this chapter have been integrated in a larger 3D tele-immersive system and have been presented to the European Union Commission on different occasions for project reviews. Updates in the 3D capturing platform have shown that the codec is more generic and can work well for different setups. It has enabled realistic composite rendering in a 3D virtual world as shown in Figure 51.

Figure 51 Highly realistic composite rendering of decoded 3D data

Nevertheless, this type of 3D compression is still in infancy, and the bit-rate requirements for this type of communication is still high. Therefore, we point out important directions for future research.

Firstly, in essence the presented coding scheme is *Lossless* as only a quantization step is applied, which bounds geometric distortion. Because of this, compression ratios around 12 times are achieved which is much lower compared to for example lossy image or video coding. Therefore, it would be interesting to explore lossy coding approaches. Interesting direction would be based on simplification (We will explore this in the next chapter), and by exploiting transform coding on irregular grids such as applying theory from the emerging field of signal processing on graphs [89] . In addition, techniques for *inter-frame* coding exploiting redundancy between frames could help reduce the bit-rate requirements.

In the next chapter we explore lossy coding using simplification. We will show that this can result in highly adaptive geometry coding using different bit-rates. Chapter 6 will explore inter-frame coding and exploitation of correlation temporally adjacent frames.

Chapter 5 Highly Adaptive Geometry Driven 3D Mesh Compression

The approach for geometry compression and transmission presented in the previous chapters enables high quality real-time 3D end-to-end communications. However, as the approach is based on near lossless techniques, the compression ratios are limited and bit-rate requirements are still high. This chapter experiments with lossy coding of live reconstructed geometry using mesh simplification. In this approach, the final user experience in the 3D tele-immersive virtual room is taken into account. This user experience is tested in a real 3D tele-immersive system, with rendering of different decoded 3D Meshes. The codec design presented in this chapter is geometry driven, i.e. first the geometry is coded followed by coding the connectivity by explicitly taking the geometry into account. The results show that this approach can be used for low bit-rate coding of reconstructed geometry in a highly adaptive manner. This codec is recommended for low bit-rate coding for participants at a distance in the room, or for those that are not actively involved in a conversation or discussion. These participants can be rendered instead with a lower level of detail. This codec complements the designs presented in the previous chapters for streaming 3D reconstructed data in the virtual room with a highly adaptive and lossy coding approach. In the presentation of this work, we carefully take all quality aspects (both related to the user experience and to the overall systems design) into account. In addition, this codec can enable geometry coding with parameter based rate control, allowing the output bit-rate to be controlled. This codec design is novel as it combines the octree with fast and efficient connectivity coding which was a combination not addressed in the previous literature. This chapter answers research question 3:

Research Question 3: How can we achieve highly adaptive lossy real-time compression and transmission of highly realistic 3D replicants based on meshes that can be used for many different bit-rates and levels of detail ?

The work presented in this chapter is based on the following two publications:

Mekuria R.N. Cesar P. "A Basic Geometry Driven Mesh Coding Scheme with Surface Simplification for 3DTI" *IEEE Communication Society: Multimedia Communications Technical Committee E-letter, May 2014*

Mekuria R.N. , Cesar P, Doumanis I, and Frisiello A., "Objective and Subjective Quality Assessment of Geometry Compression of Reconstructed 3D Humans in a 3D virtual room," in *Proceedings of the Applications of Digital Image Processing XXXVIII Conference, (part of the SPIE Optical Engineering + Applications, part of SPIE Optics + Photonics symposium)* , San Diego, USA, August 9-13, 2015

An important challenge in designing lossy codecs for this type of 3D data is that subjective and objective quality assessment is not well understood. Subjective and objective quality assessment is always needed to validate lossy coding schemes. Therefore, section 5.1 gives an overview of subjective and objective quality assessment targeting the 3D virtual immersive room. Section 5.2 presents the geometry driven mesh codec that was designed. Section 5.3 and 5.4 perform the subjective/objective quality assessment comparing this approach with the coding method presented in the previous chapter.

5.1 Objective and Subjective Quality Assessment Methodology

Fig. 52 Reverie Immersive virtual room, naturalistic user geometry is compressed and transmitted directly into the synthetic 3D scene

Fig. 52 shows the Reverie 3D immersive virtual room based on the REVERIE framework. In this application, reconstructed (scanned) 3D content is acquired and transmitted in real-time. It can be used as a realistic remote user representation. It shows that the acquired 3D user data is rendered remotely and compositely with the synthetic 3D scene. The remote transmission of such 3D geometry poses a large challenge in compression. The aim of this chapter will be to evaluate the quality of reconstruction and compression of 3D Humans in a 3D virtual room, both objectively and subjectively. This is of particular importance to compare the performance of the codec designed in the previous chapter with the lossy geometry driven codec designed later in this chapter. For lossy coding approaches, the quality assessment becomes a critical aspect that requires better understanding. The qualitative and quantitative objective quality metrics have been identified as follows:

Support a full 3D triangle mesh or point cloud representation: Does the codec support the compression of triangle mesh or point clouds that are reconstructed on the fly from 3D scanners?

Low encoder and decoder frame delay: Is the structural algorithmic and computational delay below a certain threshold such that real-time end-to-end communications is possible?

Flexible/Progressive I/O representation: Does the compressed format have a "progressive" format, such that from partial bit-streams a coarser geometry can be decoded?

Colour/normal: Does the codec support efficient, i.e. low-bit rate coding of colour and perhaps normal attributes?

Inter-Predictive Coding: Does the codec efficiently exploit redundancy between subsequent frames?

Adaptability: Is the codec able to change coding parameters to adapt to different network, system and user conditions? i.e. does it have fine grained control of bit-rate, complexity, redundancy etc.?

Robustness to information loss: Does the codec provide resilience to data losses?

Bandwidth: what is the resulting bit-rate and resulting distortion performance (R-D performance) of the codec?

Systems support and integration in overall framework. Can the codec be integrated easily in a mixed reality system, i.e. does it use generic API and programming interfaces?

Secondly, we will address the following 2 subjective quality aspects.

Subjective degradation compared to the original reconstructed mesh: what is the perceived perceptual degradation compared to the originally reconstructed mesh?

Subjective experience compared to Computer based avatar: what is the user experience of having a natural reconstructed user, instead of a computer based avatar, i.e. such as those used in traditional virtual environments?

This goes a step beyond traditional quality evaluation of geometry compression, which generally only considers algorithmic complexity, rate-distortion and the memory footprint introduced by the codec. Instead, this chapter presents a full comparative quality evaluation of the mesh codecs designed in this thesis and integrated in the REVERIE immersive virtual world. We believe the REVERIE virtual world can be a representative of future 3D immersive virtual worlds that use geometry based video for user representation.

5.2 Geometry Driven Mesh Compression

Fig. 53 illustrates a geometry driven mesh coding scheme for the immersive virtual room designed in this chapter based on a lossy coding approach. Contrary to the approach presented in the previous chapter, this algorithm is based on first coding the geometry information (V) using an octree composition scheme, and only after that code the connectivity information. The octree composition results in a voxel space composition, denoted $v_{octree}\{x,y,z\}$, where x,y,z is a key that can be used as an index in the voxel grid. The octree composition is illustrated in Fig. 54.

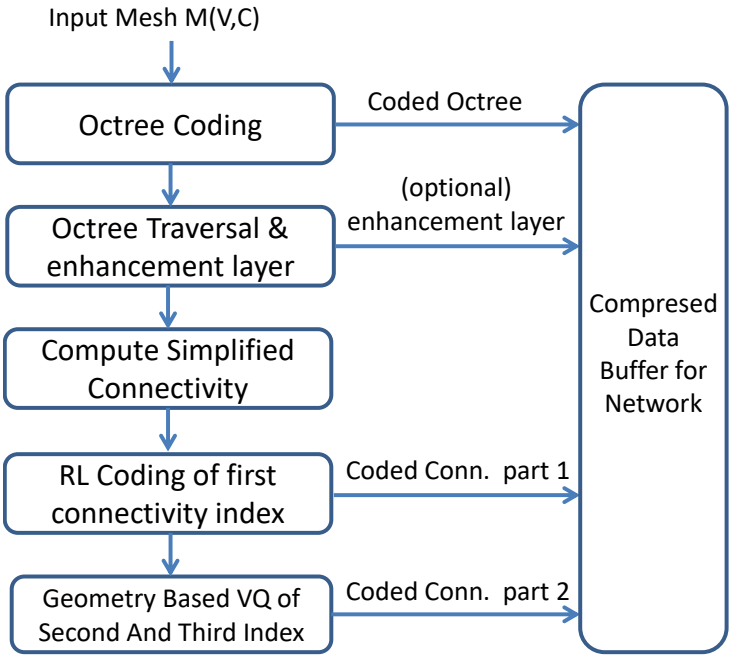


Fig. 53: Geometry Driven Mesh Codec based on [90]

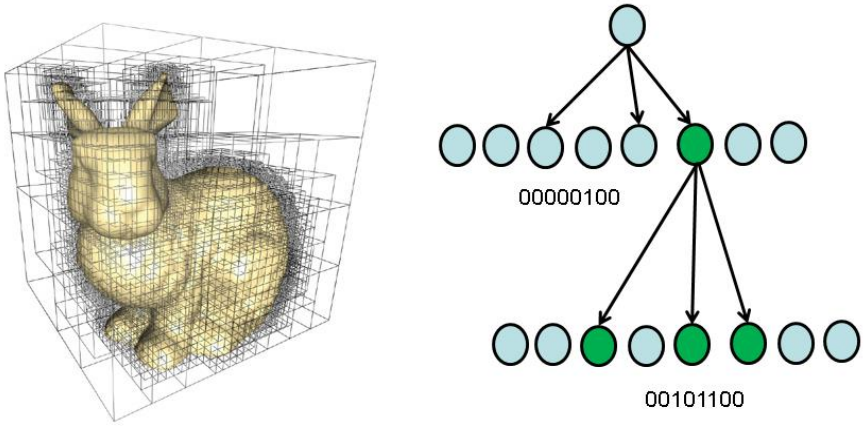


Fig. 54 Octree subdivision in space

In the second stage, the method traverses the octree grid v_{octree} storing the indices I_{simp} which are single integer indices to voxels. I_{simp} contains all vertex indices for the new simplified connectivity $C_{simplified}$, which is derived from the original connectivity $\boldsymbol{C}_{\text{original}}$ of the mesh. During this traversal, we optionally compute an enhancement layer that refines the position of the voxel center based on the positions of enclosed vertices based on a weighted average. This layer can be used to increase the decoded mesh quality without further octree subdivision. This traversal gives two mappings that are useful for coding connectivity information. First, the mapping m from the original vertex indices I_{or} to the simplified leaf voxel indices I_{simp} and, second the mapping l between I_{simp} and the 3D voxel indices $v_{octree}\{x,y,z\}$ in the grid V_{octree}.

$$m : I_{or} \rightarrow I_{simplified}\{i_{or} \in I_{or}, m(i_{or}) \in I_{simplified}\} \tag{5.1}$$

$$l : I_{simplified} \rightarrow V_{octree}\{i_{simp} \in I_{simplified}, l(i_{simp}) \in V_{octree}\} \tag{5.2}$$

Then, the original connectivity $C_{original}$ is presented as a collection $T_{un\_ordered}$ of triples of unordered indices and the new simplified connectivity as $C_{simplified.}$ as a collection $T_{ordered}$ of index triples in ascending order. $T_{ordered}$ and $T_{un\_ordered}$ represent collection of index triples as given by relations (5.3) and (5.4) below. $C_{simplified}$ is then the collection $T_{ordered}$ that satisfies relation (5.5).

$$T_{unordered} = \{i_1, i_2, i_3\} : \{i_1, i_2, i_3 \in I_{original}, i_1, \neq i_2 \neq i_3\} \tag{5.3}$$

$$T_{ordered} = \{i_1, i_2, i_3\} : \{i_1, i_2, i_3 \in I_{simplified}, i_1, < i_2 < i_3\} \tag{5.4}$$

$$\begin{aligned} C_{simplified} = \{t_{ordered}(i_1, i_2, i_3) \mid \exists tunord(i_{or1}, i_{or2}, i_{or3}) : \\ t_{unord} \in C_{original}, m(i_{or1}) = i_1, m(i_{or2}) = i_2, m(i_{or3}) = i_3\} \end{aligned} \tag{5.5}$$

Ordering of the vertex indices i_{simp} as $t_{ordered}$ in $C_{simplified}$ helps to preserve more of the spatial correlation introduced by the structured octree traversal and detect duplicate triples. Next, the set $C_{simplified}$ is ordered in ascending order of i_1 and all i_1 are coded via a run length coding scheme, that codes the number of repetitions and increments in i_1.This part is the run length (RL) coding method for the first connectivity index in Fig. 53. Next, qw construct geometry based representations t_{VQ} of each triangle $t_{ordered}$ in $C_{simplified}$ to code the second and third indices i_2, i_3. Every t_{VQ} represents a $t_{ordered}$ where i_2 , i_3 are represented by a 3D offset in the octree voxel grid V_{octree} away from connected voxels $v_1=l(i_1)$ and $v_2=l(i_2)$. Hence the elements of the set T_{VQ} satisfy relation (6):

$$T_{VQ} = \{i_1, z_2, z_3\} : \{i_1, \in I_{simplified}, z_2, \in \mathrm{Z}^3, z_3, \in \mathrm{Z}^3\} \tag{5.6}$$

Z_2 and z_3 are signed discrete 3D vectors representing a shift in the octree voxel grid V_{octree}. The T_{vq} representations are obtained from $T_{\text{ordered.}}$ we describe this by the mapping F_{vq} given in (7):

$$F_{vq} : T_{ordered} \rightarrow T_{VQ} = \{i_1, l(i_2) - l(i_1), l(i_3) - l(i_2)\} \tag{5.7}$$

In each t_{vq} in $T_{vq,}$ i_1 was already coded by the RL coding of the first index and therefore, only z_2 and z_3 need to be encoded. By structuring the connectivity information in $T_{ordered}$ and F_{vq} achieved a structure where z_2 and z_3 can be coded very efficiently via vector quantization. The prototype vectors *[-1,0,0]*, *[0,-1,0]* and *[0,0,-1]* occur over 75% of the cases in our training data for z_2 and z_3 while around 15% of the other cases are covered by the vectors *[-1,1,0]*, *[0,-1,1]*, *[1,-1,0]*. The remaining signed binary vectors represent the last 7% of the cases in our training data. We developed an efficient variable length coding (VLC) scheme to code these vectors with 2, 4 and 8-bit code words. In the exceptional other cases, we store all components of z_2 and/or z_3. The decoder executes inverse operations, i.e. the octree voxel structure is decoded, and instead of only l we also compute the inverse l^{-1} that relates the 3D octree voxel index v_{octree} to $I_{simplified}$, i.e.:

$$l^{-1} : V_{octree} \rightarrow I_{simplified} \{v_{octree} \in V_{octree}, l^{-1}(v_{octree}) \in I_{simplified}\} \tag{5.8}$$

The mapping l^{-1} is used to decode all $t_{ordered}$ recovering $C_{simplified}$ from all t_{vq} using the mapping $F^{-1}{}_{vq}$:

$$F^{-1}vq : T_{VQ} \rightarrow T_{ordered} = \{i_1, l^{-1}(l(i_1) + z_2), l^{-1}(l(i_2) + z_3)\} \tag{5.9}$$

Where i_1 was already recovered from the run length encoded data and therefore, by recovering i_2 followed by i_3 for each t_{vq}, $C_{simplified}$ is recovered. By having the geometry and both connectivity, the mesh is fully decoded. Therefore, we have presented a complete lossy geometry compression scheme that allows very fine grained control of the output resolution (number of output points) by changing the octree depth. In addition, the codec enables real-time encoding and decoding, useful for real-time end-to-end communications.

5.3 Objective Comparative Evaluation

Table 15 Comparative Evaluation of properties of codecs in MPEG-4, chapter 4 and chapter 5

Codec	Mesh	Point Cloud	Low Delay	Flexible I/O	Colour	Normal	Inter-Predictive	Adaptability	Rate / Distortion
MPEG-4 TFAN	Y	N	+	+	Y	Y	N	+	++
Conn Driven (chapter 4)	Y	N	++	+	Y	Y	N	--	+
Geometry Driven (chapter 5)	Y	Y	+	++	Y	N	N	++	+

Table 15 compares the codecs integrated into the virtual world environment, they have been discussed in the current and previous chapter. All codecs are able to directly compress the 3D mesh geometry resulting from 3D surface reconstruction. The geometry driven codec can also compress point clouds (i.e. point sampled data without the connectivity information). With connectivity driven approaches such as MPEG-4 TFAN or the approach form chapter 4 this is not possible. All codecs have low delay properties, but in different ways. The MPEG-4 TFAN Codec has a linear encoding and decoding time and can decode in real-time. For dense meshes with over 300K points (as present the test and training data) however, our tests were not able to run the encoder in real-time on our hardware/software setup (i7, 3.400 GhZ desktop machines, 16GB Ram, Windows 7, VS 2010). However, for meshes with vertices in the range of 10K points, the encoder can also run in real-time (<200ms delay) in our tested hardware/software setup. This is useful for coarse meshes that are rendered with textures that are separately coded (using JPEG image codec etc.). The connectivity driven codec is the fastest codec, especially when used with dense meshes, as it is mostly based on computing differentials and quantization only. All

codecs provide coding of the colours and MPEG-4 TFAN and the connectivity driven codec can also encode normal information. We are not sure if it is preferable to encode the normals or instead re-compute them at the receiver. This is a design choice, and good results can be achieved in both cases. The codecs currently do not support inter-predictive coding. Inter predictive coding of time varying point clouds and 3D mesh geometry has not been studied well in the literature. This deserves attention and will be explored in the next chapter with point cloud representations. In terms of adaptability, i.e. the degree in which one is able to tune the coding complexity, i.e. gain, complexity etc., the geometry driven codec is best. The geometry driven codec can be used to decrease the number of output vertices compared to the input, which enables fine grained quality control and low bit-rate encoding. This feature is not present in MPEG-4 TFAN and the connectivity driven encoder. The MPEG-4 TFAN Codec only allows quantization bits for the geometry, normal and colour data components to be specified. This enables some tuning of the bit-rate, but for real time encoding of dense meshes at low bit rates and in variable network conditions this is quite limited. In practice, reducing the amount of points in the geometry driven codec is a better way to achieve a lower quality representation than direct quantization without vertex decimation. The connectivity driven codec has no support to change the bit-rate settings on the fly, which is its key disadvantage. Last, both the MPEG-4 TFAN and the connectivity driven codec do not provide resilience to data/information loss. In current networks, such information losses are often corrected at lower layers, so it tends not to be a big problem. The geometry driven codec does not provide full error resilience, but instead provides a progressive format. This implies that from a partial correct stream a lower resolution object can be obtained. This facilitates context adaptive forwarding of data and application layer routing based on available bandwidth characteristics and application needs. In terms of rate-distortion, the MPEG-4 TFAN codec provides the best rate-distortion characteristics, while the geometry and connectivity driven codecs are slightly worse in this respect. On the other hand the last two codecs provide better support for real-time and low bit –rate encoding.

Figure 55 shows the achieved geometric quality versus the byte size per encoded frame on data reconstructed with the system in [15]. The graph plots bit-rate on the horizontal versus distortion on the vertical axis. In such a comparison, the lower the

curve, the better the codec is in the rate-distortion sense. The purple line represents the geometry driven codec at different settings that was designed in the previous section. For MPEG-4 TFAN we presented the rate-distortion for two settings (8 bits and 10 bits per geometry component, and with 6 bits per colour/normal component), in the red and green dots. The connectivity driven codec is represented in the pre-configured setting (the blue dot). The distortion was measured on the vertical axis using a symmetric root mean square (rms) distance measure between original and decoded models. What can be seen is that the connectivity driven codec and MPEG-4 TFAN are better in the rate distortion sense, but that the geometry driven codec can provides more configurations for low bit-rate encoding. This is very useful in the virtual room environment as these different properties complement each other, highlighting the benefit of having multiple codecs in the framework. We can use the geometry driven codec when a low bit-rate and low quality is needed, and gradually increase the quality. For high quality representations we can then use the connectivity driven coder for dense geometry or MPEG-4 TFAN in case meshes are sparse. In all these cases, real-time encoding and decoding can be maintained, enabling adaptive and real-time end-to-end communications in the multi-site tele-immersive framework.

Figure 56 assess the real-time performance of the different codecs running in the integrated immersive communications framework. The numbered entries represent different settings of the geometry driven encoder. The settings for the connectivity driven and MPEG-4 TFAN correspond to those in the previous section. The red value corresponds to the decoder time, while the blue line represents the encoder time. We can see that the encoder/decoder time increases exponentially for higher quality settings the geometry driven codec. For higher qualities one should therefore switch to the connectivity driven codec to still enable real-time encoding in the immersive framework. The MPEG-4 TFAN was mainly designed with offline encoding in mind, which is observed. However, MPEG-4 TFAN can be used for encoding lower resolution meshes complemented with textures (we have experimented with this in the tele-immersive framework and this is a suitable alternative approach). This comparison illustrates the complementary nature of the different codecs designed in this Thesis. In the Reverie tele-immersive framework, the geometry driven codec

will be used for low bit-rate real-time encoding, the connectivity driven codec for high bit-rate real-time and TFAN for low res input meshes with textures.

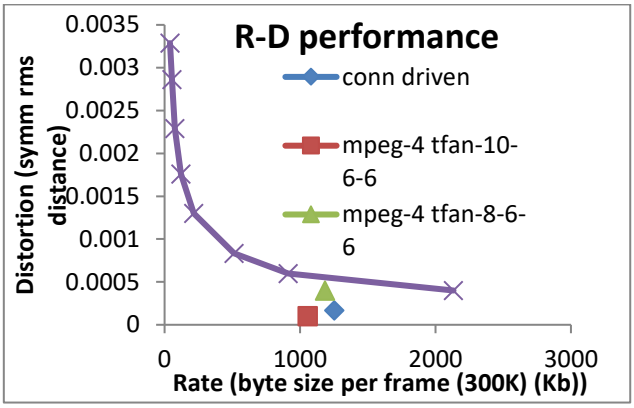


Figure 55 Rate-distortion performance codec in the immersive virtual room

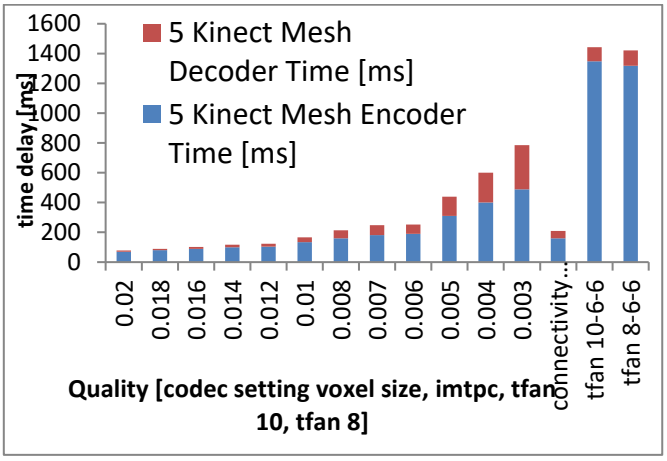


Figure 56 Real-Time performance on reconstructed mesh data with over 300K vertices

5.4 Subjective Comparative Evaluation

This section presents the subjective results of user studies carried out based on a set of test stimuli generated based on the reverie framework and the codecs presented in this thesis.

A laboratory test has been performed according the procedures suggested in ITU Standards [91]. 16 volunteer representative users have been recruited. The participant sample population was gender balanced, the age range from 25 to 38 years old (mean 30, 8), with a high educational level (degree, master or PhD), mainly in engineering branches. The users provided their informed consent to participate to the experiment when invited.

A within test has been designed: in order to estimate how individual behaviour changes when the stimuli and variables change, each subject has been asked to assess all the configurations to be tested [92]. Two iterations have been performed with different subjects. The stimuli to evaluate were a sequence of randomized videos, presenting the same subject, a 3D human moving at 8 fps in a virtual environment created within the REVERIE system (hangout virtual room). It includes a high quality 3D rendering of 3D Meshes and degradation introduced by the compression method. The independent variables observed were: The compression methods: original, connectivity driven (chapter 4) and geometry driven (this chapter) and enhanced geometry driven (this chapter)

The virtual distance of the 3D reconstructed user (far vs close to the virtual camera in the 3D space) was also tested.

The subjects have been welcomed in the laboratory, and before starting the test received the introduction to the experimental goals, highlighting the focus on the 3D human and the context of the study. The possibility to interrupt the test if needed was provided. Before evaluating the videos, each subject had the possibility to interact with the real system to become more familiar with the context of realistic users in a 3D virtual room. Then, they were asked to watch pre-recorded stimuli videos one at a time (in a randomized order). After each video the subjects expressed an evaluation

of the perceived video quality of the 3D reconstructed human. Additional rating dimension of the perceived realism into the scene (explained as the integration of the 3D human render with the virtual environment) was given.

Each dimension has been evaluated by using the MOS – Mean Opinion Score [91], allowing to collect the average of the opinions ("votes") on the single given conditions. The rating scale (on 5 points: Excellent 5, Good 4, Fair 3, Poor 2 , Bad 1) was printed and left on the table as reference for helping the subject to reply. The test protocol applied allows collection of the subjective evaluations on the single video just after it is watched, on the basis of the immediate impression. For each test variable, the mean value and the standard deviation of the statistical distribution of the assessment grades has been calculated. Table 16 shows modest differences between original and decoded meshes. Pairwise comparison of each codec setting with the original mesh via paired samples t-tests, showed that only for the geometry driven codec (0.008) the mean difference was significant ($p<0.05$). The paired sampled t-test is a statistical test to compare the means of two different normally distributed variables. To check its validity normality assumptions have been asserted. Next, we investigated the effect of nearby and far away camera views in the 3D word. For geometry driven both near and far were tested significantly lower compared to the original mesh ($p<0.05$). On the other hand, differences of the original with connectivity driven were not found significant.

Table 16 MOS and Compression Methods

	MOS and Compression Methods							
	Original		Connectivity Driven		Geometry Driven		Geometry Driven enhanced	
	Mean	st.dev	Mean	st.dev	mean	st.dev	Mean	st.dev
Quality	3,2	0,9	3,0	1,0	2,7	0,9	2,9	1,0
Realism	3,3	0,8	2,9	0,9	3,0	0,9	2,9	1,0

Table 17 MOS and Virtual Distance

	MOS and Virtual Distance			
	Near		Far	
	Mean	Std. dev	Mean	Std. dev
quality	2,8	0,9	3,1	1,0
realism	2,9	0,9	3,2	0,9

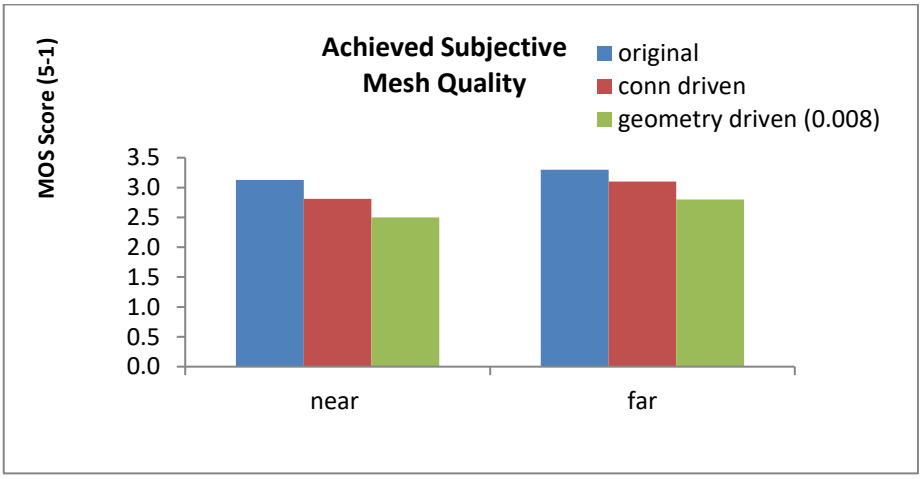


Fig. 57 Subjective results evaluation natural user quality (original, conn driven, geometry driven)

5.5 Conclusion and Discussion

This chapter started work on lossy geometry compression and subjective evaluation. To do so we first introduce some important aspects related to objective and subjective quality assessment of geometry compression. Next we develop a lossy highly adaptive geometry driven codec. By reducing the vertex count, lower bit rates can be achieved compared to the codec from chapter 4 and in addition the bit-rate can be

finely tuned. The subjective and objective evaluation shows the complementary nature of these codecs compared to the codec designed in the previous chapter. The codec presented in this chapter can be used for low priority streams (far away in the scene), while the other codecs (chapter 4, MPEG-4 TFAN) are better for highly realistic (near lossless). In addition, as the subjective evaluation shows the distortion introduced by all codecs is limited compared to the perceived quality of the original reconstruction. The set of coding approaches presented so far have been integrated to support 3D mesh based tele-immersive communication in different scenarios. Despite covering both the high quality real-time and lossy/adaptive scenario, still some unaddressed issues remain. The first one is the exploitation of inter-frame redundancy. The second is to consider the transmission of point data only (without connectivity). Such data should be easier to acquire and would be more compact as the connectivity information need not be stored. In the next chapter, we will address these two issues in a design for a time varying point cloud codec.

Chapter 6 Time Varying 3D Point Cloud Compression

The previous chapters developed compression and transmission techniques for 3D tele-immersive communications based on 3D Meshes. The codecs developed addressed the range from near lossless to lossy. The next step that is taken in this chapter is to develop compression for an alternative representation: 3D Point Clouds. 3D Point Clouds are similar to meshes but do not contain connectivity information. This makes them slightly easier to acquire and process than 3D meshes. This chapter focusses on the problems of lossy coding using inter predictive coding, exploiting redundancy between frames. In addition, it develops a technique for lossy coding of point cloud colour attributes based on their spatial correlation. This addresses two key challenges related to point cloud compression for 3D Tele-Immersive communication. This chapter addresses research question 4:

Research Question 4: How do we design improved Real-time Compression of 3D Time Varying Point Clouds with improved lossy colour coding and inter-prediction with a negligible perceptual distortion (compared to the current state of the art in practical real-time point cloud compression?

This work extends the state of the art in point cloud compression, prior art did not include lossy inter predictive coding, lossy real-time colour coding, parallelization. These were all not addressed in prior art. In addition, this work is geared towards the requirements of tele-immersive video, which are addressed by these new methods.

The work in this chapter is based on the following publication:

Mekuria R.N. Blom K., Cesar P. "Design, Implementation and Evaluation of a Point Cloud Codec for 3D Tele-immersive Video" IEEE Transactions on Circuits and Systems for Video Technology to appear in sept. 2016

6.1 Introduction

Traditionally, 3D polygon meshes have often been used to represent 3D object based visual data (as in the previous chapters). However, point clouds are simpler to acquire than 3D polygon meshes as no triangulation needs to be computed, and they are more compact as they do not require the topology/connectivity information to be stored. Therefore, 3D point clouds are more suitable for real-time acquisition and communication at a fast rate. However, realistic reconstructed 3D Point clouds may contain hundreds of thousands up to millions of points and compression is critical to achieve efficient and real-time communication in bandwidth limited networks.

Compression of 3D Point Clouds has received significant attention in recent years [93] [89] [94]. Much work has been done to efficiently compress single point clouds *progressively*, such that lower quality clouds can be obtained from partial bit streams (i.e. a subset of the original stream). To compare different solutions, often the *compression rate* and the *geometric distortion* have been evaluated. Sometimes the *algorithmic complexity* was analysed, and in addition schemes for *attribute coding* (i.e. colours, normals) have been proposed. While these approaches are a good starting point, in the context of immersive, augmented and mixed reality communication systems, several other additional factors are of importance.

One important aspect in these systems is *real-time performance* for encoding and decoding. Often, in modern systems, parallel computing that exploits multi-core architectures available in current computing infrastructures is utilized. Therefore, the *parallelizability* becomes important. Second, as in these systems point cloud sequences are captured at a fast rate, inter-frame redundancy can be exploited to achieve a better compression performance via *inter-prediction* which was usually not considered in existing static point cloud coders.

Furthermore, as tele-immersive codecs are intended for systems with real users *subjective quality assessment* is needed to assess the performance of the proposed codec in addition to the more common *objective quality assessment.* Further, a codec should be *generic*, in a sense that it can compress point cloud sequences coming from different setups with different geometric properties (i.e. sampling density, manifoldness of the surface etc.).

With these requirements in mind we introduce a codec for time varying 3D Point clouds for augmented and immersive 3D video. The codec is parallelizable, generic, and operates in real time on commodity hardware. In addition, it exploits inter-frame redundancies. For evaluation, we propose an objective quality metric that corresponds to common practice in video and mesh coding. In addition to objective evaluation using this metric, subjective evaluation in a realistic mixed reality system is performed in a user study with 20 users.

The codec is available as open source and currently serves as the reference software framework for the development of a point cloud compression technology standard in JTC1/ SC29/WG11 (MPEG)[3]. To facilitate benchmarking, the objective quality metrics and file loaders used in this work have all been included in the package. In addition, the point cloud test data is available publicly.

The rest of the chapter is structured as follows. Section 6.3 details the lossy attribute/colour coding and progressive decoding scheme. In section 6.4 the inter-predictive coding algorithm is detailed. Section 6.5 details the experimental results including subjective test results in a mixed reality system, objective rate distortion and real-time performance assessment. The next section provides an overview of the codec, and this section, the context.

[3]http://wg11.sc29.org/svn/repos/MPEG-04/Part16-Animation_Framework_eXtension_(AFX)/trunk/3Dgraphics/

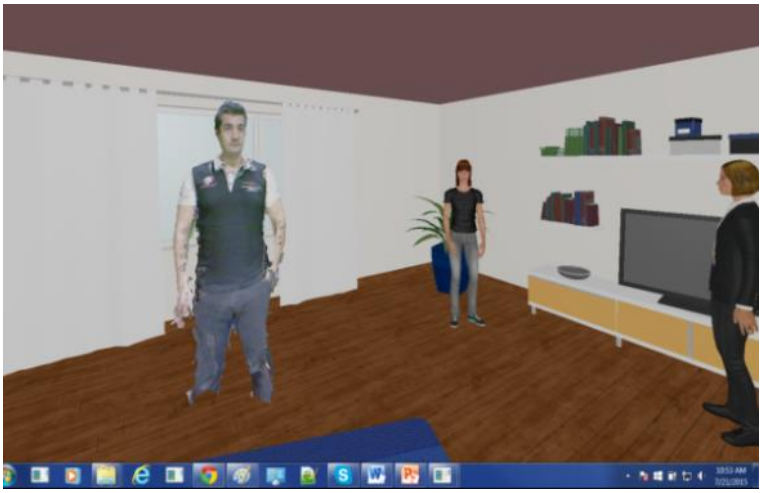


Figure 58 Screenshot of composite rendering in virtual room based on the REVERIE System. The Point cloud naturalistic content is rendered compositely with synthetic content. Navigation and user support enables interaction between naturalistic point cloud and synthetic avatar users

Figure 59 Low cost point cloud capturing setup, point clouds are reconstructed from multiple 3D input streams of calibrated Microsoft Kinect devices (deployment at University of Illinois during a guest visit).

6.1.1 Contributions of this Chapter

This chapter proposes a novel compression framework for progressive coding of time varying point clouds for 3D immersive and augmented video. The major contributions are:

- Generic Compression Framework: that can be used to compress point clouds with arbitrary topology (i.e. different capturing setups or file formats)
- Inter-Predictive Point Cloud Coding: correlation between subsequent point clouds in time is exploited to achieve better compression performance
- Efficient Lossy Colour Attribute Coding: the framework includes a method for lossy coding of colour attributes that takes advantage of naturalistic source of the data using existing image coding standards
- Progressive Decoding: the codec allows a lower quality point cloud to be reconstructed by a partial bit stream
- Real-Time Implementation: the codec runs in (near) real-time on commodity hardware, benefiting from multi-core architectures and a parallel implementation

While work in the past has delivered point cloud compression frameworks, lossy inter-prediction, attribute coding and parallelization have received little consideration. Further, subjective evaluation on point cloud codec designs was typically not performed.

6.1.2 Related Work

Point Cloud Compression: There has been some work on point cloud compression in the past, but most works only aim at compression of static point clouds, instead of time-varying point clouds. Such a codec was introduced in [95] based on octree composition. This codec includes bit-reordering to reduce the entropy of the occupancy codes that represent octree subdivisions. This method also includes colour coding based on frequency of occurrence (colourization) and normal coding based on efficient spherical quantization. A similar work in [96] used surface approximations to predict occupancy codes, and an octree structure to encode colour information. Instead, the work in [93] introduced a real-time octree based codec that can also exploit temporal redundancies by XoR operations on the octree byte stream. This method

can operate in real-time, as the XoR prediction is extremely simple and fast. A disadvantage of this approach is that by using XoR, only geometry and not colours can be predicted, and that the effectiveness is only significant for scenes with limited movement (which is not the case in out envisioned application with moving humans). Last, [94] introduced a time varying point cloud codec that can predict graph encoded octree structures between adjacent frames. The method uses spectral wavelet based features to achieve this, and an encoding of differences, resulting in a lossless encoding. This method also includes the colour coding method from [89], which defines a small sub-graphs based on the octree of the point cloud. These subgraphs are then used to efficiently code the colours by composing them on the eigenvectors of the graph Laplacian. The resulting co-efficient are subsequently quantized and entropy encoded using appropriate techniques.

Another approach for point cloud compression based on spanning tree was presented [97] , this is a single rate codec without progressive levels, and looks unsuitable for our application of interest in tele-immersive communications.

Point cloud representation have also been useful in the robotics area where 3D scans of the environment are used to help with the navigation. An efficient representation to store point cloud data in memory was developed in OctoMap [98] that presents a useful data structure based on octree that can be kept in the memory of typical robotic device.

Other works related to point cloud representation deal with other aspects such as rendering, capturing and sometimes even object reconstruction and or detection. We consider them to be outside of the scope of this work which particularly aims at the compression of point clouds for tele-immersive applications.

6.2 Overview of Point Cloud Codec

6.2.1 Requirements and use case

3D Video based on point clouds is relevant for augmented and mixed reality applications as shown in Figure 58. In addition, it has various applications such as to store data used in geographic information systems and 3D printing applications. This

chapter focuses on time-varying point cloud compression for 3D immersive and augmented video and follow the requirements for point cloud compression as defined in [11]:

Partial Bit Stream Decoding: it is possible to decode a coarse point cloud and refine it.

Lossless Compression: the reconstructed data is mathematically identical to the original.

Lossy Compression: compression with parameter control of the bit-rate.

Time Variations and Animations: temporal variations i.e. coding of point cloud sequences, should be supported

Low Encoder and Decoder Complexity: this is not a strict requirement but desirable for building real-time systems.

6.2.2 Schematic Overview

The architecture design of the proposed 3D Video based on point clouds combines features from common 3D Point cloud (octree based) codecs [96], [95] and common hybrid video codecs such as MPEG-4 p.10 AVC and HEVC (that include block based motion compensation). Based on the numbering in the diagram in Figure 60 the most important components in the codec are detailed, which also correspond to contributions of this chapter.

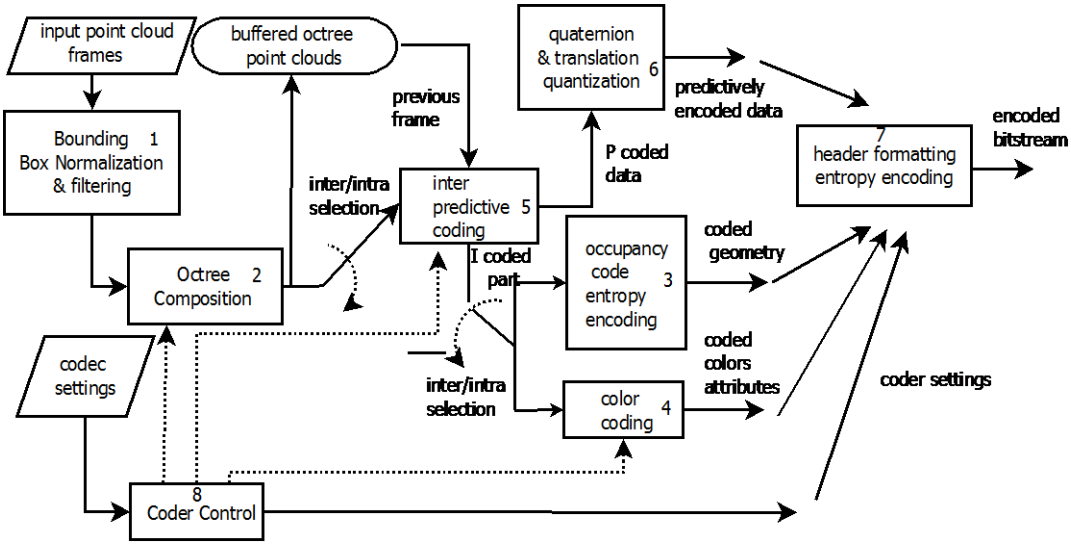


Figure 60 Schematic of time-varying point cloud compression codec

Bounding Box Alignment and filter (1): A specific algorithm for bounding box computation and alignment has been developed. This algorithm is applied to the point cloud before the octree composition (2) is performed. This algorithm aligns subsequent frames by computing a common expanded bounding box. This allows common axis and range representation between frames which facilitates the inter-predictive coding and a consistent assignment of the octree bits. In addition, it includes an outlier filter, as preliminary experiments have shown that outlier points can reduce both the effectiveness of the bounding box alignment and of inter predictive coding, and should be filtered out from the input clouds. See 6.3.1 for all details.

Constructing the Progressive Octree (2): The encoder recursively subdivides the point cloud aligned bounding box into eight children. Only non-empty children voxels continue to be subdivided. This results in an octree data structure, where the position of each voxel is represented by its cell center. Its attributes (colour) is set to the average of enclosed points and needs to be coded separately by the attribute encoder. Each level in the octree structure can represent a level of detail (LoD). The final level of detail is specified by the octree bit settings. This is a common scheme

for both regularizing the unorganized point cloud and for compressing it. (See 6.3 for all details).

Coding of the Occupancy Codes (3) (LoD): To code the octree structure efficiently, the first step is the encoding of (LOD's) as sub-divisions of non-empty cells. Contrary to previous work [96] [95], we develop a position coder that uses an entropy coder in the corresponding LoD's. In addition, by avoiding surface approximations as in [96] and occupancy re-ordering as in [95] we keep the occupancy code encoder as simple and fast (See 6.3.2 for all details). In particular we set the level of decodable LoD's to two, thus reducing overhead

Coding of Colour Attributes (4): In order to code the colour attributes in the point cloud in a highly efficient manner, we integrated methods based on mapping the octree traversal graph to a JPEG image grid, exploiting correlation between colour attributes in final LoD's. The rationale of the JPEG mapping is that as the colour attributes result from natural inputs, comparable correlation between adjacent pixels/points exists. By mapping the octree traversal graph to a JPEG grid we aim to exploit this correlation in an easy fast way suitable for real-time 3D tele-immersion. See 6.3.3 for all details.

Inter Predictive Frame Coding (5): This chapter introduces a method for inter-predictive encoding of frames based on previous inputs that includes both rigid transform estimation and rigid transform compensation. We first compute a common bounding box between the octree structures of consecutive frames i.e. block (1). Then we find shared larger macroblocks on K (=4) levels above the final LoD voxel size. For blocks that are not empty in both frames, colour variance and the difference in point count are used to decide if it is needed to compute a rigid based transform using the iterative closest point algorithm, or not. If this is the case and the iterative closes point algorithm is successful (converges) the computed rigid transformation can be used as a predictor. The rigid transform is stored compactly in a quaternion and translation quantization schema **(6).** This prediction scheme can typically save up to 30% bit-rate. This is important to reduce the data volume at high capture rates. See 6.4 for all details.

Coder Control (8) and Header Formatting (7): The coder uses pre-specified codec settings (via a configuration file) which include the octree bit allocation for (2), macroblock size and prediction method for (5) and colour bit allocation and mapping mode for (4). In addition it includes the settings for the filter and the colour coding modes. We use common header format and entropy coding based on a static range coder to further compress these fields.

6.3 Intra Frame Coder

The intra frame coder consists of three stages (1, 2 and 3 in Figure 60). It first filters outliers and computes a bounding box. Second, it performs an octree composition of space. Third, entropy encoding of the resulting occupancy codes is performed.

6.3.1 Bounding Box Normalization and Outlier Filter

The bounding box of a mesh or point cloud frame is typically computed as a box with a lower corner *(x_min,y_min,z_min)* and an upper corner *(x_max,y_max,z_max)*. This bounding box is used as the root level of the octree. The bounding box can change from frame to frame as it is defined by these extrema. As a consequence, the correspondence of octree voxel coordinates between subsequent frames is lost, which makes inter-prediction much harder. To mitigate this problem, Figure 61 illustrates a scheme that aims to reduce bounding box changes between adjacent frames. The scheme enlarges (expands) the bounding box with a certain percentage δ, and then, if the bounding box of the subsequent frame fits this bounding box, it can be used instead of the original bounding box. As shown in Figure 61 the bounding box of an intra frame is expanded from the BB_IE (x_min – bb_exp, y_min – bb_exp, z_min – bb_exp) to upper corner (x_max+ bb_exp, y_max+ bb_exp,z_max+bb_exp), where bb_exp was computed from δ and the ranges of the original bounding box. Then, the subsequent P cloud is loaded and if a bounding box computed for this frame, BB_P, fits the expanded bounding box BB_IE, the P is normalized on BB_IE. BB_IE is subsequently used as the octree root and the frame P can be used by the predictive coding algorithm presented in section 6.4.1. Otherwise, the expanded bounding box is computed as BB_PE and the cloud P is normalized on BB_PE. In this case, the cloud P is intra coded instead, as the predictive coding algorithm only works when the bounding boxes are aligned.

The bounding box operation is an important pre-processing step to enable efficient intra and inter-coding schemes. It can be assumed that point clouds represent segmented objects reconstructed from multi-camera recordings and in our experiments we also worked with such data. We discovered that in some cases the segmentation of the main object (human or object) is not perfect and several erroneous background/foreground points exist in the original cloud. As these points can degrade the bounding box computation and alignment scheme, filters have been implemented to pre-process the point cloud. The method is based on a radius removal filter. It removes points with less than K neighbours in radius R from the point cloud. This filter removes erroneously segmented points from the point cloud and improves the performance of the subsequent codec operations.

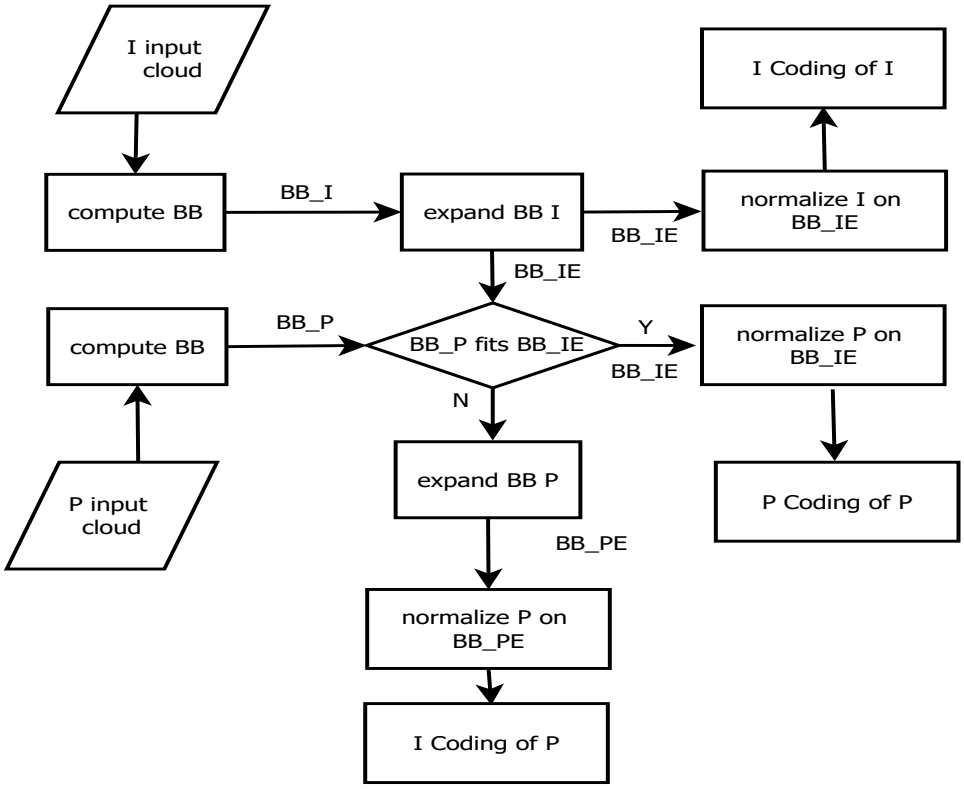


Figure 61 Bounding box alignment scheme

6.3.2 Octree Subdivision and Occupancy Coding

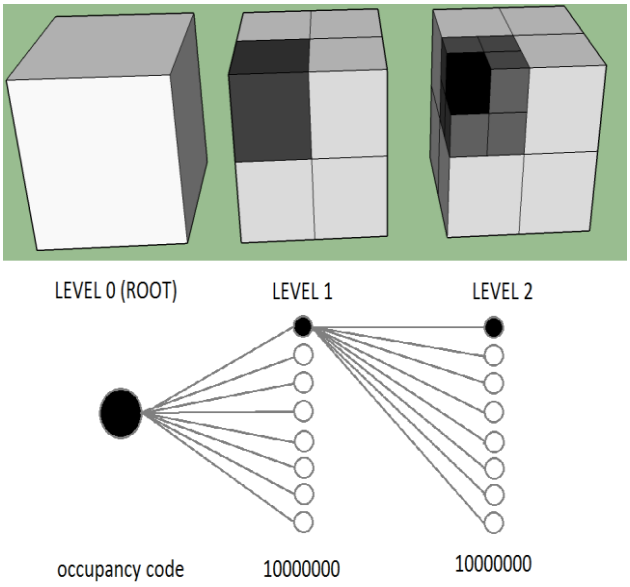


Figure 62 Octree Composition of Space

Starting from the root node, each octree subdivision results in an 8 bit occupancy code as shown in Figure 62. Only non-empty voxels are sub divided further and the decoder only needs the occupancy codes to reconstruct each LoD of the point cloud. Therefore, efficient coding of occupancy codes is central in point cloud codecs such as [1] and [96]. The work in [96] aims at reducing the entropy by using surface approximations at each level to predict likely occupancy codes. The method in [95] introduces a method based on reordering the order of the bits in the occupancy code, by traversing the 8 child cells in different orders. This traversal order is computed by identifying the point neighbourhood in each subdivision. These approaches are not so practical for 3D immersive Video based on point clouds as such continuous surface approximations and statistics are computationally expensive and hinder parallelization. Instead, we follow a modified approach as taken in [93] (based on a carry less byte based range coder), which we applied to the different decodable LoD's. Experiments showed that this gave better results to compression of all LoD's at once

While previous work on progressive static point cloud compression enabled many decodable LoD's for interactive rendering, it limited the number of decodable LoD's in the implementation to 2 to avoid introducing a large overhead in the coding of colour attributes that need to be coded for each level.

6.3.3 Colour Coding

Apart from coding the object geometry position, coding of colour attributes is essential to achieve a high visual quality rendering. In the past, methods based on DPCM [96], colourization (keeping a frequency table of colours) [1] and based on octree structures have been proposed [96]. However, these methods are all characterized by low compression efficiency as they cannot exploit correlation between multiple co-located points. An attempt to improve this was made by Zhang et al in [89], by modelling subsets of points as a weighted graph that can be directly derived from the octree. The colour attributes are then modelled as a signal on this graph. The signal values are then projected on each of the eigen vectors, resulting in spectral representation. This method imposes a large compuational overhead in computing the eigen values and vectors. Therefore, instead this chapter introduces a method based on existing legacy JPEG codec that exploits correlation present in the point cloud recontructed from natural inputs. Instead of mapping the octree directly to a graph and treating the colour attributes as a graph signal, we map the colour attributes directly to a structured JPEG image grid from a depth first tree traversal. We have defined a mapping that enables efficient coding of the colours based on a zig-zag pattern. As shown in Figure 63 the colours are written to 8x8 blocks based on the depth first octree taversal. The grid entries in Figure 63 correspond to the order of the octree traversals, while the grids themselves correspond to pixels in the pre-allocated image grid. So based on the depth first octree traversal pixels are written to an image grid. This grid is re- allocated based on the original leaf count of the octree. The method takes advantage of the fact that in the depth first traversal subsequent pixels are often co-located and therefore correlated. The DCT transform in the JPEG codec later exploits this from the mapping to 8x8 blocks. As for some octree traversal steps (big jumps) the assumption of correlation does not hold, some distortion could be introduced. This can be combatted by storing residuals. However, subjective assessment of the decoded data in section 6.5.5 did not assert that distortion introduced in this method was perceptually significant.

0	1	2	3	4	5	6	7	64
15	14	13	12	11	10	9	8	79
16	17	18	19	20	21	22	23	80
31	30	29	28	27	26	25	24	…
32	33	34	35	36	37	38	39	…
47	46	45	44	43	42	41	40	…
48	49	50	51	52	53	54	55	…
63	62	61	60	59	58	57	56	…

Figure 63 Scan line pattern for writing octree values to an image grid

6.4 Inter-frame Coder

To perform inter-frame encoding between subsequent point clouds, this section presents an inter-frame prediction algorithm that re-uses the intra frame coder presented in the previous section in combination with a novel lossy prediction scheme based on iterative closest point algorithm (ICP). All abbreviations used to describe the algorithms are given in Table 18.

Table 18 Symbols used for predictive encoding

SYMBOLS USED IN INTER PREDICTIVE POINT CLOUD COMPRESSION ALGORITHM

Symbol	Description
ICP	Iterative Closest Points
I	Preceding Reference Point Cloud intra frame
P	Point Cloud that will be predictively encoded
C_p	Predictively coded compressed part of *P*
C_i	Intra coded compressed part of *P*
M_i	*I* frame Macroblocks organized in an octree
M_p	*P* frame Macroblocks organized in an octree
M_s	Macroblocks non empty in both I and P
M_px	Macroblocks non-empty in *P* only
M_p_i	Macroblocks in P that could not be predicted
pc	Point count range percentage

6.4.1 Predictive Algorithm

Figure 64 outlines the inter-predictive coding algorithm. The algorithm codes the data in the *P* frame in two parts. An *I coded part (**C_i**)* of data is coded that contains the vertices that could not be predictively coded and a *P coded part (**C_p**)* with data that could be predicted well from the previous frame. The algorithm starts with the normalized and aligned *I* and *P* clouds (based on the bounding box alignment algorithm presented in subsection 6.3.1). The Macroblocks ***M_i*** are (1) generated at level K above the final LoD of the octree *O* that was generated coding the intra frame in the previous iteration. K=4 was chosen resulting in macroblocks of 16x16x16. While K=3 resulting in 8x8x8 or K=5 resulting in 32x32x32 blocks are also possible, K=4 gave the best results in terms of block overlap and final convergence. A similar macroblock octree at K=4 is also computed for *P* resulting in ***M_p*** (1).

In the next step (2) each of the macroblocks ***M_p*** is traversed to find if a corresponding macroblock exists in ***M_i.*** For each of these blocks ***M_s*** (that are non-empy in *I* and *P*) the inter-predictive coding algorithm continues. Blocks occupied only in ***M_p*** and not in ***M_i*** are stored in ***M_px*** and written to the *intra coded part (C_i)* of the compressed data that will be coded with the intra coder presented in section 6.3. In the current implementation, all data coded to the intra coded part is written to a point cloud data structure, which will be later coded with the intra coding algorithm from 6.3.

Each block in ***M_s*** will be a candidate for the prediction; Two extra conditional stages are traversed before the predictive coding of the block is started. First, the number of points is asserted to be in the range of the two corresponding Macroblocks in ***M_i*** and ***M_p*** (4). The threshold to do this was set to plus or minus *pc*%. Only when the number of points is in range between the two blocks, prediction is possible. We set the value of *pc* to 50%. This parameter *pc* can be tuned by comparing the percentage of macroblocks that are shared to the percentage of blocks suitable for prediction and the resulting quality. In the second stage, a check is done on the colour variance of the points in the macroblock. We only perform inter-prediction in areas of the point cloud that have low colour/texture variance.

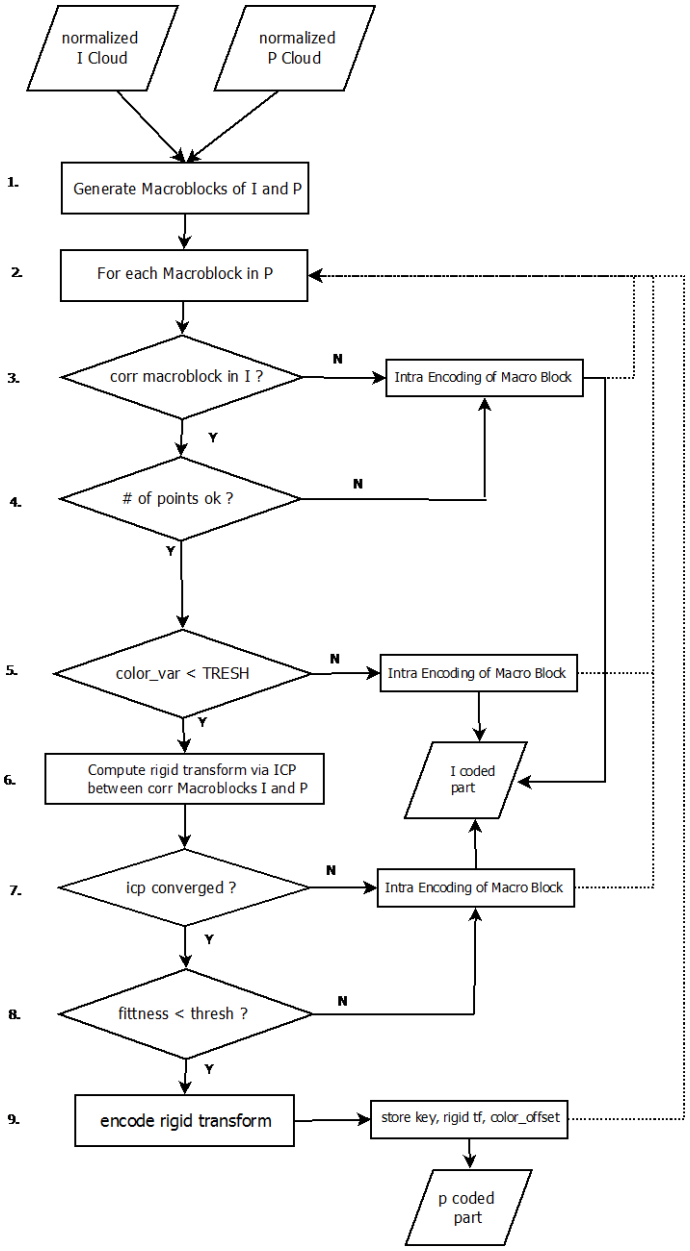


Figure 64 Inter Predictive Point Cloud Coding Algorithm

Table 19 Data Structure of an inter coded block of data

Key_x		Key_y		Key_z
Quat1		Quat2		Quat3
T1		T2		T3
C_off1	C_off2	C_off3		

The reason is that inter predictive coding may result in visible artefacts in high variance texture image regions. The prediction is performed based on computing a rigid transform between the blocks mapping the points M_i to M_p. This computation is based on the iterative closest point algorithm, which only takes the geometric positions into account and not the colours. The threshold of the total variance in the blocks *C_var_thresh* was set to 100 in experiments. In these low variance corresponding macroblocks, finally a motion vector can be computed as a rigid transform via *ICP*. In case the *ICP* converges and the achieved fitness level is below a certain threshold (*icp_fit_thresh*), (in our case this was chosen to be 4 times the target resolution of the point) the points are inter predictively coded. In this case the predictor is coded (that is the rigid transform and the (x,y,z) key k in the macroblock grid ***M_p***). Otherwise the points are written to the intra coded part. This is a point cloud data structure that is coded after the inter-prediction terminates. The encoding of the rigid transformation **T** is as follows. It is first composed as a rotation matrix **R**, and a translation vector **t**. The rotation matrix **R** is converted to a quaternion **q_or(s,t,u,v)** and quantized using a quaternion quantization scheme of only 3 numbers Quat1, Quat2, Quat3 (16 bits per number). The translation vector **t** is quantized using 16 bits per component (T1, T2, T3). Further details on the scheme for rigid transform coding resulting from ICP are presented in the next subsection.

Last, a colour offset is optionally coded to compensate for colour difference offsets between the corresponding macroblocks. This colour offset can optionally be used to compensate fixed colour offsets between blocks due to changes in lighting that have resulted in a brightness difference. The position in ***M_p*** is stored as a key (x,y,z) *k* using 16 bit integer values for each component and corresponds to a macroblock in ***M_i*** **(***k***)**.

This key can be used to decode the predicted blocks directly from the previously decoded octree frame ***M_i*** *in any order*. This random access indexing method also enables parallel encoding/decoding of the data which is important for achieving real-time performance. The memory outline of the data field is presented in Table 19. Each row represents 6 bytes, and the data structure consists of 18-21 bytes (as coding the colour offset is optional). The final *predictively coded part* **C_p** of the byte stream consists of a concatenation of all predicted blocks resulting from the prediction. All blocks that could not be predicted ***M_p_i*** are stored in a single point cloud that is coded using the method developed in section 6.2.

6.4.2 Rigid Transform Coding

The 4 by 4 rigid transform matrix **T** resulting from ICP based prediction is composed into a (unitary) 3 by 3 rotation matrix **R** and a translation 3 by 1 translation vector **t**.

$$\begin{matrix} \boldsymbol{R} & \boldsymbol{t} \\ \mathbf{0} & 1 \end{matrix}$$

The matrix rotation ***R*** is more compactly represented as a 4 element quaternion.

$$\boldsymbol{q}(s,t,u,v)$$

The quaternion is not only more compact, but also makes the rotation more susceptible to round off errors that are introduced in the quantization of its elements. The resulting quaternion is a unit quaternion that satisfies the relationship.

$$s^2+t^2+u^2+v^2=1$$

This allows one component to be coded implicitly based on this relationship. Based on [99], we chose the largest component of ***q*** to be coded implicitly. In addition, the sign of this element needs not be stored as it we can make sure it is always positive (by negating the quaternion, this does not change the represented rotation). We scale the other values in the range $\left[0,1/\sqrt{2}\right]$ as $1/\sqrt{2}$ is the maximum possible value of the second largest element of the quaternion. This scheme allows the rotation to be stored by 3 elements that are quantized using 16 bits (including 1 sign bit). The decoded quaternion **q_d** may still suffer from round off errors and numerical instability. In experiments we have observed that this is in less than 4% of the cases. We

test for these cases in the encoder and when they occur they are quantized by the linear independent two rows ***R*** and a sign vector for the third dependent vector to code the rotation matrix directly. This alternative quantization scheme guarantees correct recovery of the rotation in all cases. The vector ***t*** is quantized using 16 bits per component.

6.4.3 Parallelization

The proposed algorithm and bit streams have been designed to enable parallel execution. The algorithm in Figure 64 can be parallelized as follows. First, the generation of Macroblocks in I and P (1) can happen in parallel. Next, in the traversal of macroblocks in ***M_p*** (2), operations (5) and especially (6) are most computationally intensive but can happen using parallel computation. The computation of colour variance (5) in a point cloud subset can easily be offloaded to a GPU. Most importantly, the ICP prediction can be parallelized, as the ICP prediction on each macroblock in ***M_s*** can happen independently. Even more, due to the data structure outlined in Table 19, the results can be written to *C_p* in any order as the key index also enables decoding blocks independently. We have implemented this using Open MP for multi core processor Intel architectures.

The I coded (***C_i)*** part is a continuously updated point cloud with points that could not be predicted. This point cloud is later compressed (after the algorithm in Figure 64 terminates) resulting in ***C_i*** based on the octree based scheme in section 6.3. As in experiments it was observed that this algorithm spends a large fraction of the time in organizing the octree structure, parallelizing the octree composition is a good possibility to speed up this part of the algorithm. Octree data structures are common, and this has already been intensively studied, for example in [100]. While such techniques can be used to speed up the algorithm, for this work the focus of parallelization is on the ICP prediction, which distinguishes this codec and also uses a large fraction of the computation time.

6.5 Experimental Results

6.5.1 Datasets, Experimental setup

The hardware used in the experiments are a Dell Precision M6800 PC with intel core i7-4810MQ 2,8 MHz CPU and 16.0 GB of Ram running 64 win7 Operating system, a Dell Precision T3210 (Xeon 3.7 Ghz), and a custom built system with i7 3.2 GhZ running Ubuntu Linux. The datasets used for the evaluation (http://vcl.iti.gr/reconstruction/) have been used, which are realistic tele-immersive reconstructed data based on [2] and [101]. The software was implemented based on MS VC++ 2010 on windows platform based on algorithms available [102] and using gcc on the Linux operating system. Note that we compare to the available real time point cloud codec in [93] which uses an octree schema and DPCM colour coding and was used as base for the software implementation

6.5.2 Objective Quality Evaluation Metric

To evaluate the point cloud quality, we deploy a full reference quality metric that combines common practices from 3D mesh and Video Compression: a PSNR metric based on point to point symmetric root mean square distances.

The original point cloud V_{or} is a set of K points without a strict ordering (where v contains both position and colour information):

$$V_{or} = \{(v_i): i = 0 \ldots . . K - 1\} \quad (6.1)$$

The decoded cloud does not necessarily have the same number of points.

$$V_{deg} = \{(v_i): i = 0 \ldots . . N - 1\} \quad (6.2)$$

The full reference quality metric Q_{point_cloud} is computed by comparing V_{or} and V_{deg} as follows. Instead of distance to surface we take the distance to the most nearby point in V_{deg}, v_{nn_deg}. We defined geometric PSNR in (6.5) as the peak signal of the geometry over the symmetric rms distance defined in (6.4). The colour difference of the original cloud to the most nearby point in the degraded cloud v_{nn_deg} is used to compute the PSNR per YUV component (psnr_y psnr_u etc) (6.7). The v_{nn_deg}

is efficiently computed via a K-d tree in L2 distance norm based on the algorithm available in [102].

$$d_{rms}(V_{or}, V_{deg}) = \frac{1}{\sqrt{K}} \sum_{v_l \in V_{or,}} ||v_l - v_{nn\_deg}||_2 \quad (6.3)$$

$$d_{sym\_rms}(V_{or}, V_{deg}) = \ max(d_{rms}(V_{or}, V_{deg}), d_{rms}(V_{deg}, V_{or})) \quad (6.4)$$

$$psnr_{geom} = 10log_{10}(||\ max_{x,y,z}(V_{deg})||_2^2 / \ (d_{sym\ rms}(V_{or}, V_{deg}))^2) \quad (6.5)$$

$$d_y(V_{or}, V_{deg}) = \frac{1}{\sqrt{K}} \sum_{v_l \in V_{or,}} ||y(v_l) - y(v_{nn\_deg})||_2 \quad (6.6)$$

$$psnr_y = 10log_{10}(||max_y(V_{deg})||_2^2 / \ (d_y(V_{or}, V_{deg}))^2) \quad (6.7)$$

Additional metrics for the quality assessment include partial bit rates for colour, attribute and geometry data, and encoder/decoder time. These quality evaluation metrics are well aligned with existing practice in mesh and video compression and are recommended for evaluation of point cloud codecs [103]. For evaluation of time varying point cloud compression in 3D tele-immersive virtual room we will also perform subjective tests with real users.

6.5.3 Intra Coding Performance

To illustrate the effect of Level of Detail on decoding the bit stream, we show the level of detail (LoD), bit rate and quality in Figure 65. From experience with the given datasets with the realistic 3D immersive system LoD 11 gives a realistic representation when in close range of the point cloud. For point clouds at a distance in the room, lower quality LoD would be sufficient. The results in Figure 65 plots the quality metric defined in (3) based on PSNR against the bitrate in Kilobytes per frame. In Figure 66 we show the results of the bitrate of the colours per output point for different LoD's. For the DPCM quantization of 4, 5, 6 and 7 bits per colour component was set. For the JPEG colour coding scheme, 75 and 80 have been used for the JPEG quality parameter. For higher LoD's both the DPCM and JPEG colour coding scheme are more efficient, as correlation between neighbouring points increases. Overall, the colour coding scheme based on JPEG provides much lower bitrates and lower quality. However, at approximately the same quality, bit-rate is up

to 10 times lower compared to DPCM based coding. Such configuration is very useful for the targeted application of 3D Tele-immersion, where low bit rates are needed. In addition, the difference in objective quality does not necessarily correspond well to the subjective quality. In 6.5.5, we will show in the subjective studies that the colour quality degradation introduced by the colour coding method is negligible even compared to 8-bits DPCM based coding.

In Figure 67– Figure 69 we show the number of bytes per output point versus the objective colour quality based on the metric defined in formula (6.7) for the Y, U and V Components for each of the different LoDs (7-11). As for higher LoD's, the quality increases (more points) and the bytes per output point decreases, therefore, curves show higher quality for lower bitrates. The JPEG based scheme gives qualities compared to 4, 5 and sometimes even 6 bit DPCM at up to 10 times less bytes per output point. This is the main advantage gained by the colour coding methods, i.e. low bit rate lossy coding of the colour attributes. As the method re-uses a fast optimized JPEG implementation, we do not introduce any extra computation time compared to the DPCM based methods (i.e. the encoding speed and decoding speed have been similar for both methods), which is an advantage compared to methods such as [89]. For the U and V components the achieved quality with JPEG is slightly lower compared to the Y component.

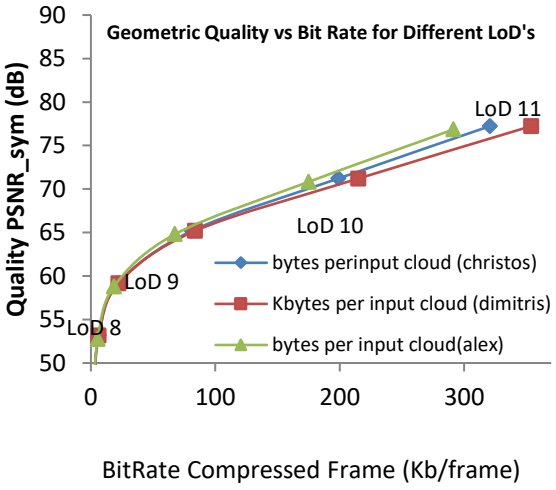


Figure 65 Level of detail Bit Rate vs Quality

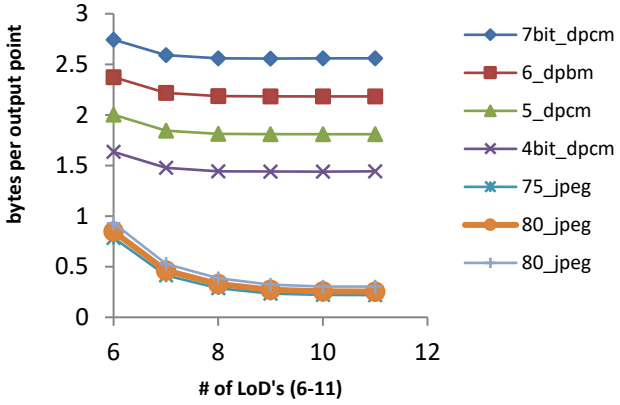


Figure 66 Bytes per point for colour coding per LoD (right)

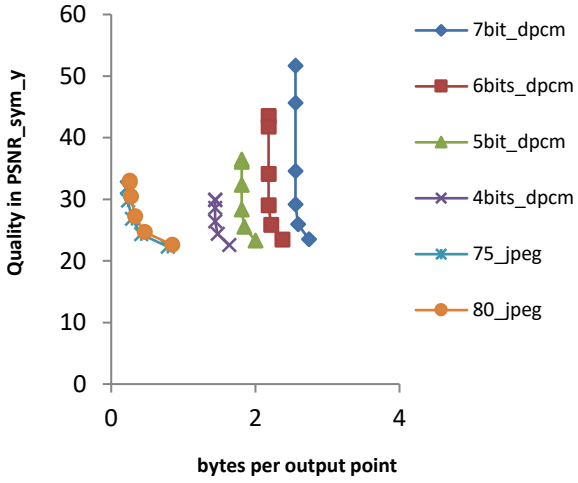


Figure 67 Rate Distortion for colour coding (Y component) (left)

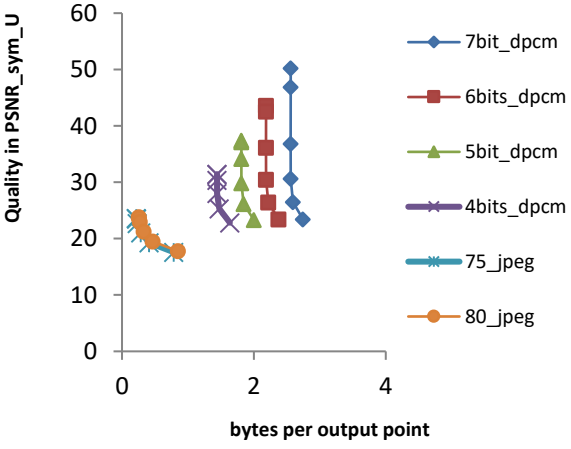


Figure 68 Rate Distortion for colour coding (U component) (right)

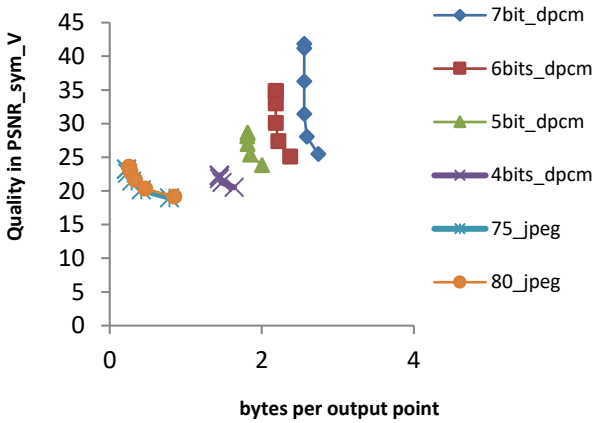


Figure 69 Rate Distortion for colour coding (V component)

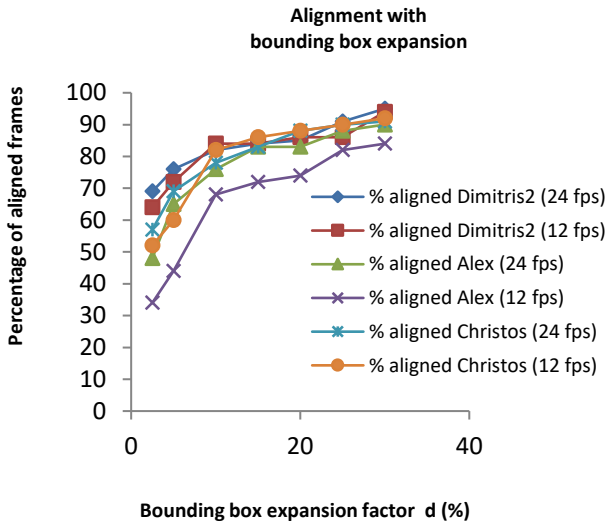
6.5.4 Inter Predictive Coding Performance

In Figure 70 we show the percentage of shared macroblocks in the coarse octree between aligned frames for LoD 11 (which is the preferred LoD for encoding and decoding) at K=4 (16x16x16 blocks), (d=20%) and the ICP convergence percentage. For all tested datasets (based on capturing at 12 or 24 fps) over 60% of macroblocks are shared in aligned frames, and approximately 30% are both shared and converging in the ICP stage (these are used for prediction). The percentage that is shared and converged is shown in red, and relates largely to the bitrate savings (which are over 30% for the Dimitrios dataset that has the largest percentage of shared blocks). Figure 71 compares the compressed size of intra coded frames with predictive frames, for the Alex dataset the size was reduced by 20.5 percent compared to intra coding only, for Christos this was 25.5 % and for Dimitris 34.8 %. Again, the bit saving relates largely to the percentage of shared blocks between the octree based point clouds.

The comparison of the objective quality of predictively and intra coded frames is shown in Figure 72 based on the metric (6.5). The loss in geometric PSNR in pre-

dictively encoded frames is about 10 dB in each of the tested datasets. We will explore the subjective effect of this degradation in the next section. Figure 74 to Figure 77 show the results for both predictive and JPEG based lossy colour coding.

Inter-Predictive coding can be applied when two frames are aligned using the bounding box alignment scheme from 6.3.1. In Figure 70 we show the results of bounding box alignment on the 3D Tele-immersive point cloud datasets that we use in the evaluation. For a bounding box expansion factor of 20 percent we achieve a good alignment percentage > 75 % and we use this setting in each of the following experiments. All point clouds have been rendered under the exact same conditions (lighting, camera position) using MeshLab Software[4]. For the LoD 11 and LoD 10 prediction artefacts are not visible. For the data under test, a good convergence and shared block percentage was achieved for LoD 11 (11 bit octree setting, i.e. 11 quantization bits per direction). For different datasets, some tuning could be necessary to find the best setting. The software implementation of codec can be configured via a parameter_config text file.



[4] http://meshlab.sourceforge.net/

Figure 70 Bounding box alignment (left)

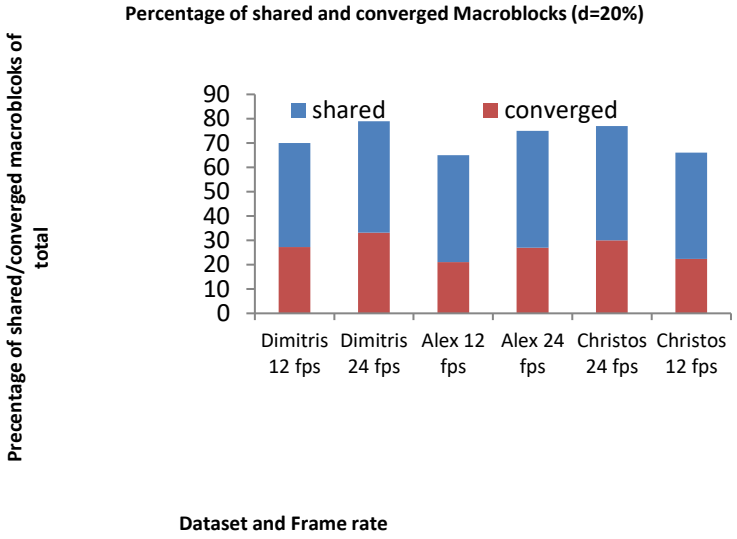


Figure 71 percentage of shared macroblocks (right)

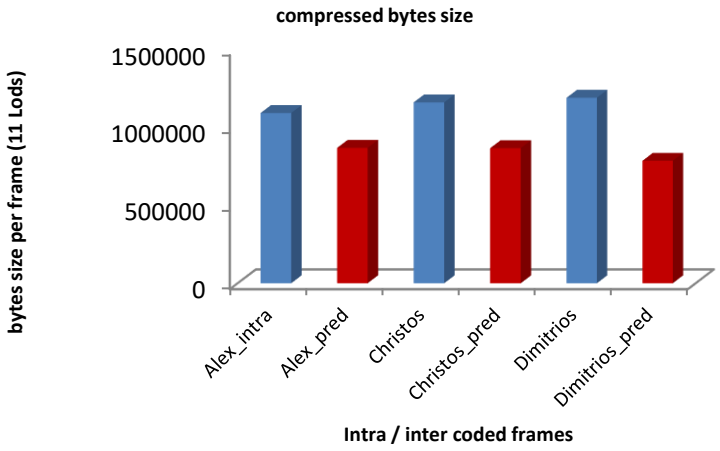


Figure 72 compressed byte size (right)

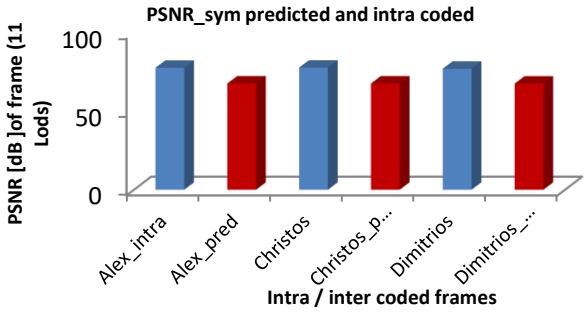


Figure 73 Inter and inter-prediction Prediction Quality

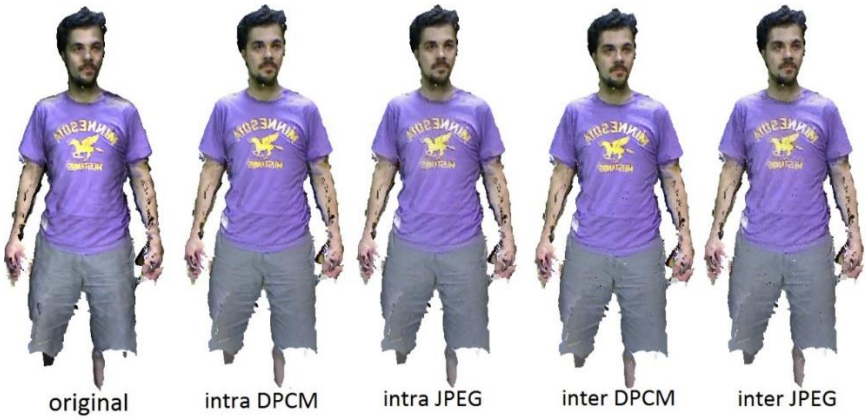


Figure 74 Results at LoD Christos 11 (11 bit octree)

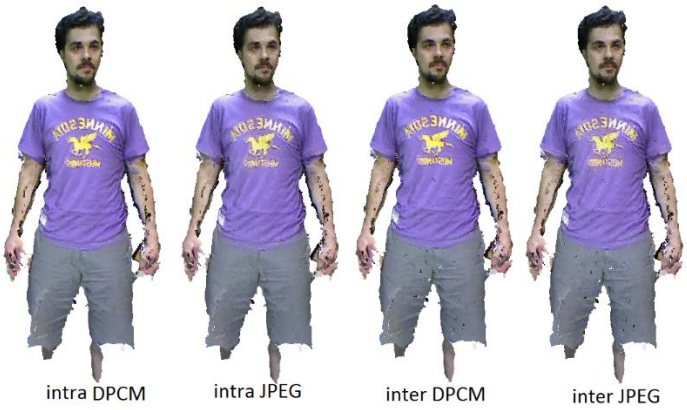


Figure 75 Results at LoD Christos 10 (10 bit octree)

Figure 76 Results at LoD Alex 11 (11 bit octree)

Figure 77 Results at LoD 10 (10 bit octree)

6.5.5 Subjective Quality Assessment

We evaluate the subjective quality of the codec performance in a realistic 3D tele-immersive system in a virtual 3D room scenario where users are represented and interact as 3D avatars and/or 3D Point clouds. We show the system in Figure 78. The user is represented as an avatar and can navigate in the room (forward, backward, left, right, rotate) and look around using the mouse (free viewpoint). The point cloud video represents the remote user that is rendered compositely in the room using OpenGL. The Reverie Rendering system allows different modules to render compositely, which enables mixed reality of both synthetic (avatar) and naturalistic (point cloud) content in the scene. The system has implemented basic collision detection and navigation, so users cannot navigate through any of the occupied 3D space, e.g. point cloud, avatar or other world objects.

20 test users have been recruited to work with the system and assess the performance. The users have been introduced to the goal and context of the experiment, and signed the consent form for using this information for scientific purposes. The group consisted of 4 women and 16 men, in age range from 24 to 62. All were working in Information Technology (electronic engineering, computer science or mathematics). Nationalities were mixed (10 Dutch, 3 Spanish, 2 Italian, 2 Chinese, 2 German, 1 English), education levels above B.Sc and the participants have been exposed to different versions of the point clouds at LoD 11 shown in Figure 74 and Figure 76 (based on the point cloud data in Alex-Zippering and Christos Zippering available at http://vcl.iti.gr/reconstruction/). The inter-predicted frames are presented in an I-P-I-P interleaved pattern to simulate a realistic presentation. All datasets are rendered at 23 fps when presented to the user (this corresponds to the capture rate). Each of the users have been exposed to the three codecs (DPCM, JPEG, Inter predicted) conditions on two different datasets (Alex, Christos) in a random order, resulting in a total of six test conditions. The controlled setup consisted of a 47 inch LG TV (model num. 47LB670V) screen positioned at 1.5 meters from the users running the Reverie system, shown in Figure 78, and a laptop + mouse to control the avatar. After each test condition the users have been requested to fill the questionnaire on 8 different quality aspects on a 1-5 scale ranging from Bad to Excellent based mean opinion score scale (MOS) (Bad=1, poor=2, fair=3,good=4, excellent=5). These aspects include the overall quality of the 3D human, the quality of the colours, the perceived

realism of the point cloud, the motion quality, the quality from near/far and how much the user felt "together" as being in a room with a real user comparing the avatar and the point cloud representations. The results in Figure 80 to Figure 85 can be interpreted as follows. The values 1-5 correspond to the Mean Opinion Scores of the results for each quality aspect. The error bars represent the standard deviation around the mean opion score. Due to the usage of a low cost capturing setup with Multiple Microsoft Kinect 1 which is based on structured infrared light which results in interference, the original quality is already not free of artifacts. Nevertheless, many parts of the 3D reconstruction and the motion are photorealistic and natural, which was also indicated by most users. The main aim of the experiments is to check that the developed codec does not introduce significant extra subjective distortions. All point clouds are stored and decoded from a 11 LoD octree, with DPCM (Or. In Figure 74-Figure 78), the proposed colour coding (JPEG) and prediction (Pred. in Figure 74-Figure 78.). As can be seen for the Alex dataset the colour coding and prediction do not introduce a significant degradation in quality, while for Christos some difference is observed (but not signficant). The assessment of colour quality in Figure 81 compares the 8 bit DPCM coding to the JPEG coding and inter-prediction. Surprisingly, the JPEG colour coding method does not introduce significant subjective distortion, as this method codes at up to 10 times lower bit rates compared to the 8 bit DPCM colour data. The results on perceived realism and the quality of the motion are shown in Figure 83 and Figure 84. All versions represent the user well and are judged between fair and good at 3.45 and 3.75 without significant differences. Figure 84 compares the perceived quality of the point cloud from near and far away, this shows that from nearby the point cloud quality is seen as low quality while from distance it is between fair and good. This is mainly due that the rendering of the point clouds from nearby allows you to see through the points, while from a distance this is not the case. Perhaps looking into alternative rendering methods or converting the point cloud to a mesh when nearby might solve this issue. Last, this experiment investigated how the users valued the presence of the 3D point cloud video in the room compared to a simple computer avatar in terms of "feeling together", which is the ultimate aim of tele-precence technologies. In this respect the point cloud representation scored significantly better than the computer avatar (Figure 85).

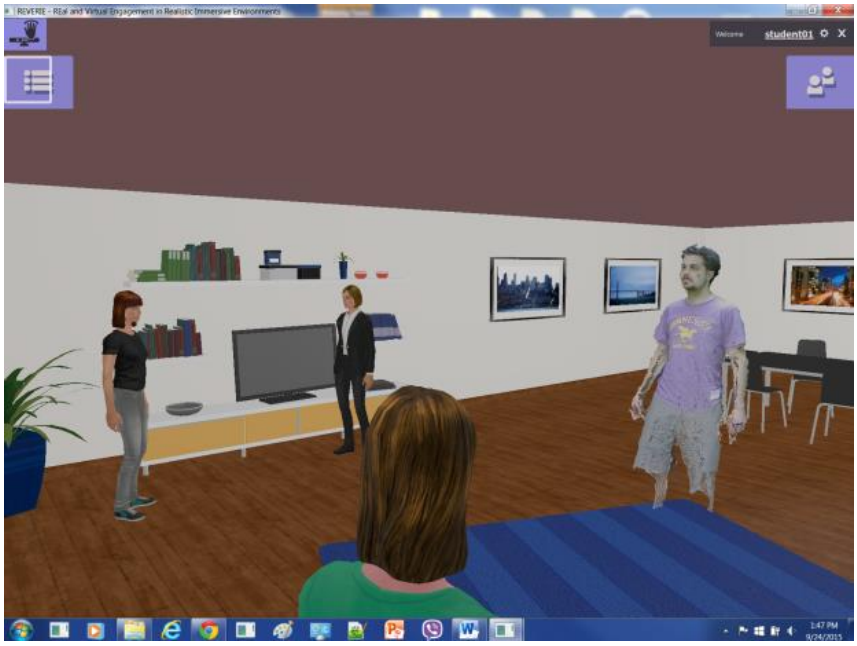


Figure 78 Practical Immersive System (left)

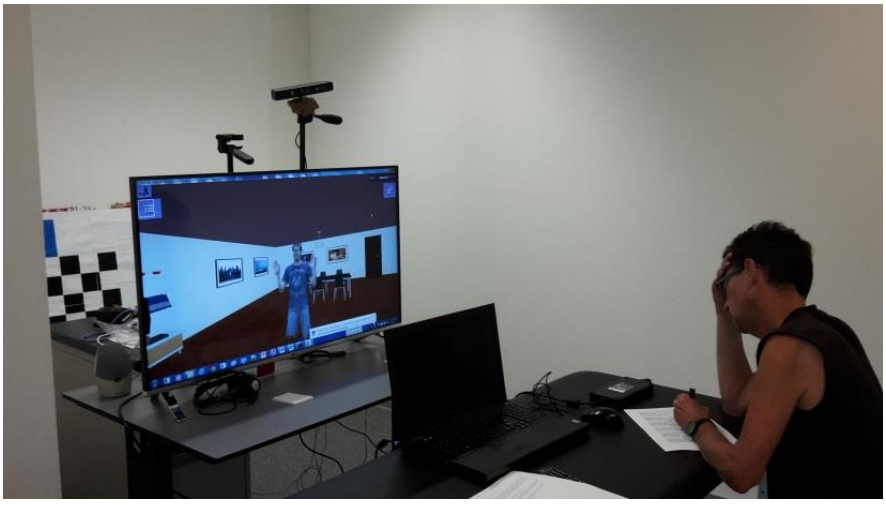


Figure 79 Test setup with user filling in the questionnaire (right)

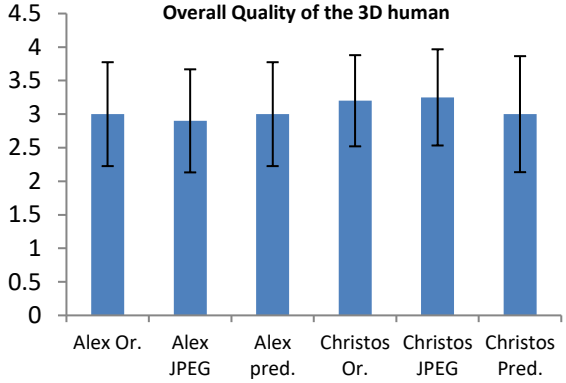


Figure 80 Subjective Results, perceived quality of the 3D Human (left)

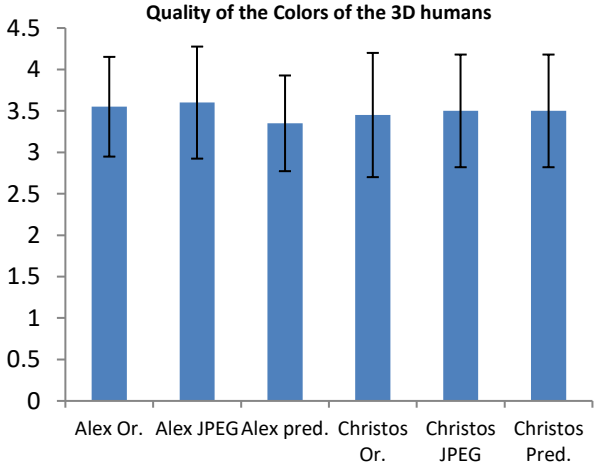


Figure 81 Subjective Results, colours of the 3D Human (right)

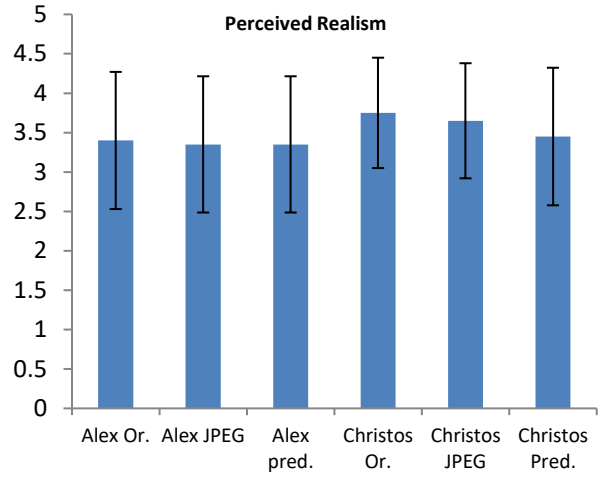


Figure 82 Subjective Results, Quality of the motion (left)

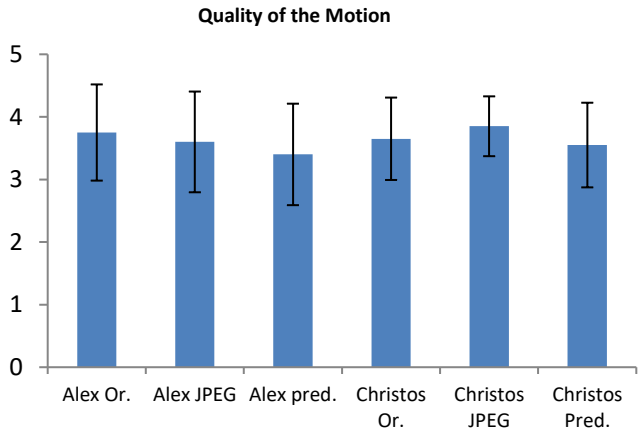


Figure 83 Subjective Results, perceived realism (right)

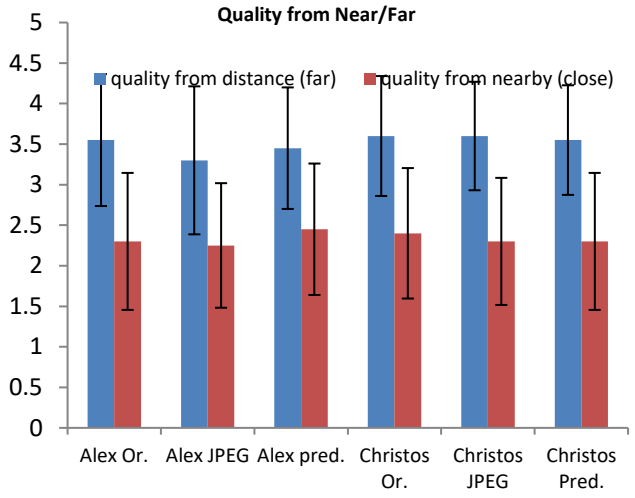


Figure 84 Subjective Results Quality, near vs far (left)

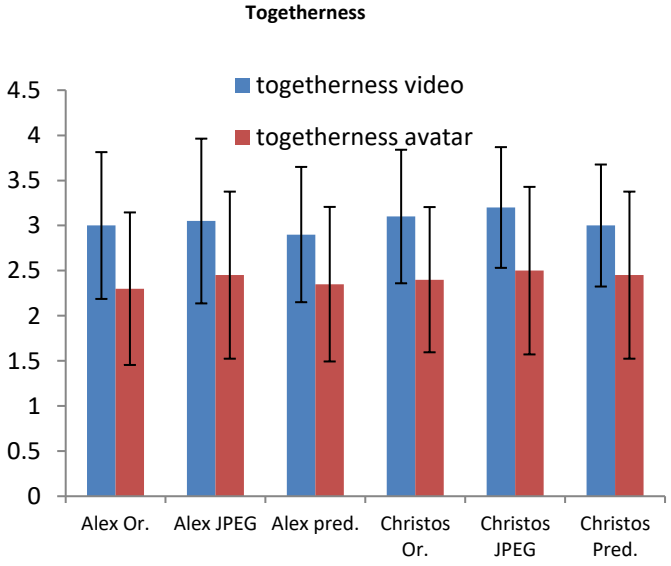


Figure 85 Feeling of togetherness, avatar user versus 3D Video user (right)

6.5.6 Real-Time Performance and Parallelization

We have assessed the real-time performance of the codec on various computer hardware showing that the codecs run in real time (<400 ms) at 11 LoD's for the intra coder and near real time (<1s) for the inter-predictive encoder. All tests have used the Dimitrios2-Zippering dataset running on 3 intel based computers, one based on i7 2.8 GhZ and win7, one Xeon 3.7 GhZ running win7 (both compiled using VC++ 2010) and one i7 3.20 GhZ running Ubuntu Linux (gcc compiler suite). The real-time results for intra encoding for different LoD's and both proposed (JPEG) and original DPCM based colour coding are shown in Figure 86. The decoding times are also lower (similar pattern) and are omitted due to space constraints. In addition, we have been able to speed up the inter-predictive encoding significantly by parallelizing its execution based on OpenMP for multi core intel architectures (We measured up to 20 % improvement on the windows platforms). This hints in the direction that that the design requirement for parallelization is met. The real-time performance of the inter-predictive coding algorithm and its parallelization are shown in Figure 87, for 11 LoD's plotting against the number of cores/threads used to compute the ICP.

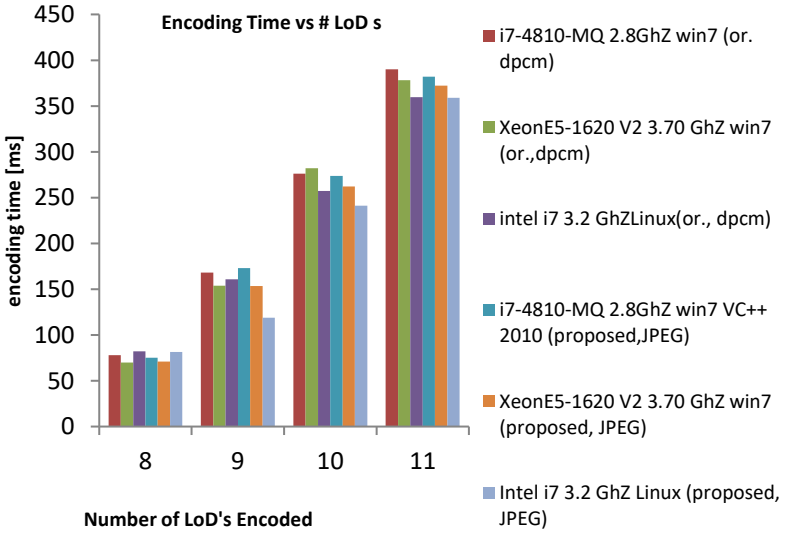


Figure 86 Real-Time Performance of intra coding for different configuration (left)

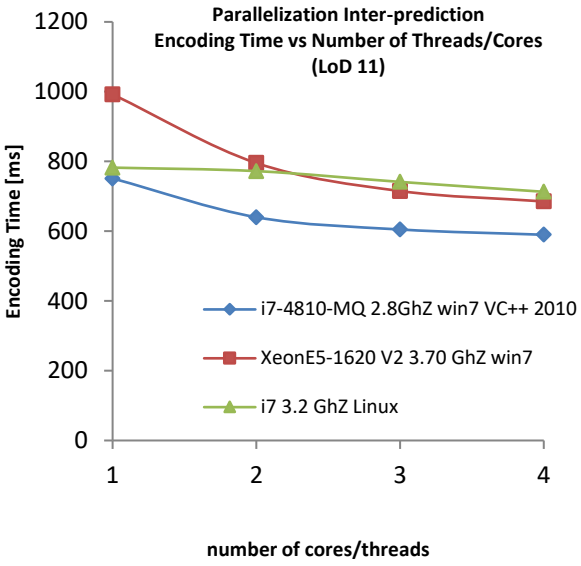


Figure 87 Real-Time Performance of inter coding for different configuration (right)

6.6 Conclusion and Discussion

In this chapter we introduce a point cloud compression method that is suitable for 3D tele-immersive and mixed reality systems. It presented a hybrid architecture for point cloud compression that combines typical octree based point cloud compression schemes with hybrid schemes common in video coding. The architecture improves the state of the art by providing a mode for inter-prediction on the unorganized point cloud and a lossy real-time colour encoding method based on legacy JPEG methods. This enables easy implementation and real-time performance. We have implemented this architecture in an implementation that is both available as open source https://github.com/RufaelDev/pcc-mp3dg, (based on the point cloud library) and available in the important media standardisation body: MPEG as a first experimentation platform for development of point cloud compression in MPEG. We rigorously evaluated the quality of the solution objectively; using a novel PSNR based quality metric that combines metrics from 3D Meshes and video for point clouds. In addition, this chapter performed subjective evaluation in a real mixed reality system that combines natural point cloud data and computer graphics based 3D content with

20 users. The results show that the degradation introduced by the codecs is negligible. In addition, the point cloud has an added value in terms of “feeling together” compared to simple avatar representations, highlighting the importance of point clouds in these applications. These results further assert the importance of work on compression of point clouds and its standardisation for immersive and augmented reality systems.

The next chapter will investigate the integration of codecs and transmission methods in a 3D tele-immersive streaming engine.

Chapter 7 3D Tele-immersive Streaming Engine

Beyond 3D compression and transmission, large challenges in the 3D tele-immersive streaming context arise from the multi-stream, multi-site nature of these systems. Multiple heterogeneous sites need to inter-connect and send multiple streams over bandlimited networks towards each other. This introduces the need for session management and stream setup between these sites taking heterogeneity into account (light render only to very heavy clients). In addition, the streaming engine should provide an Application Programming Interface (API) for stream creation and teardown and media synchronization. This chapter presents the implementation and evaluation of a 3D streaming engine that provides these functionalities that have not been presented in previous work in both 3D tele-immersion and media streaming. This chapter addresses research question 5:

Research Question 5: how can we design a 3D media streaming engine for heterogeneous 3D tele-immersive communications that is compatible with the current internet infrastructure (i.e. existing transport and signalling protocols)*?*

This chapter is based on the following publication:

Mekuria R.N. *Frisiello A., Pasin., M, Cesar P.S. "Network Support for Social 3-D Immersive Tele-Presence with Highly Realistic Natural and Synthetic Avatar Users" in ACM workshop on Massive MultiUser Virtual Environments MMVE'15, Portland USA, in conjunction with ACM MMSys'15*

This chapter will use the codecs and transmission schemes developed in the rest of this thesis to develop the overall streaming engine and its API as integrated and deployed in the REVERIE tele-immersive system.

7.1 Introduction

Figure 88 shows an example of a live reconstructed mesh rendered in a 3D virtual scene with computer animated 3D avatars. This mesh was reconstructed in real-time

from streams of the first generation of Microsoft's Kinect based on the work in [2]. To obtain a real-time distributed shared experience that combines such natural and synthetic contents, signalling, session management and the Application Programming Interface become important (assuming the basic transmission and compression problems have been addressed in the earlier parts of this thesis).

Figure 88 A natural user rendered in a virtual scene with synthetic users

The **contributions** of this chapter are the implementation and evaluation of the complete framework to support networked multi-site 3D tele-presence. Emphasis will be on the synchronization of 3D audio and 3D visual streams and session management that have not been addressed in previous chapters. This chapter introduces the multi-purpose UDP/TCP **streaming kernel** that address the streaming needs with a convenient Application Programming Interface. Further, the framework includes error resilient real-time transmission and support for basic **media synchronization** between heterogeneous streams. This chapter also briefly discuss the implemented **session management** protocol for signalling custom data types that was deployed on top of the XMPP presence protocol. For natural users, this framework integrates the **3D Mesh compression** designed in the previous chapters. For avatar based users and messaging data, this framework integrates and evaluates different **real-time messaging** solutions based on available publish and subscribe protocols. Lastly, this chapter further details the integration of this solution in a larger 3D tele-immersive

test bed that includes 3D audio, avatar users, natural users and rendering in a virtual 3D room. This chapter also evaluates the overall **streaming performance** in a 3-way scenario with heterogeneous sites (light, to heavy) in the REVERIE tele-immersive framework.

7.2 3D Streaming Engine

This section details the design and implementation of the system. The real-time TCP/UDP streaming engine is presented in section 7.2.2. The session management scheme is discussed in section 7.2.3. Together they comprise the networked infrastructure for social immersive 3D communication with natural and synthetic users. In the next subsection the architecture is first presented.

7.2.1 Modular 3D Immersive Communication Framework Architecture

Figure 89 shows a simple outline of a modular 3D tele-immersive system architecture such as in the REVERIE platform. Any user runs a main application that loads modules for specific render and capture capabilities based on its local configuration (which are stored in an xml format for convenience). For example, there exists a module for natural 3D reconstruction via depth cameras, one for animating pre-recorded data via skeleton tracking, for real-time complex human shape pose modelling, 3D audio rendering and capturing, 3D computer controlled avatar characters and for navigation and representation of the world. For example application A loads natural user data (module A), audio capture (Capture B) and a module to render incoming natural user streams (i.e. Render A). The information about the loaded modules is shared at login to the session management system which provides authentication and user login and setup of streams. When user B, that can only render natural users (it is a passive user), logs in it can request the natural user stream from user A, but user A will not request any such streams from user B. The modular architecture for tele-immersive systems is useful as it can enable different types of clients to be implemented. However, this should be supported by a network and session management protocol. In this combination, this modular architecture could allow terminal scalability from passive, very light users to users with powerful capture and render capabilities. To allow stream setup between corresponding modules, the system needs to keep track of module and payload types and their correspondence. In the

implementation, we do this via an administered data table kept up specific to the framework. In a practical deployment, such payload types could be registered in bodies such as the internet assigned naming authority (IANA) or other similar organizations.

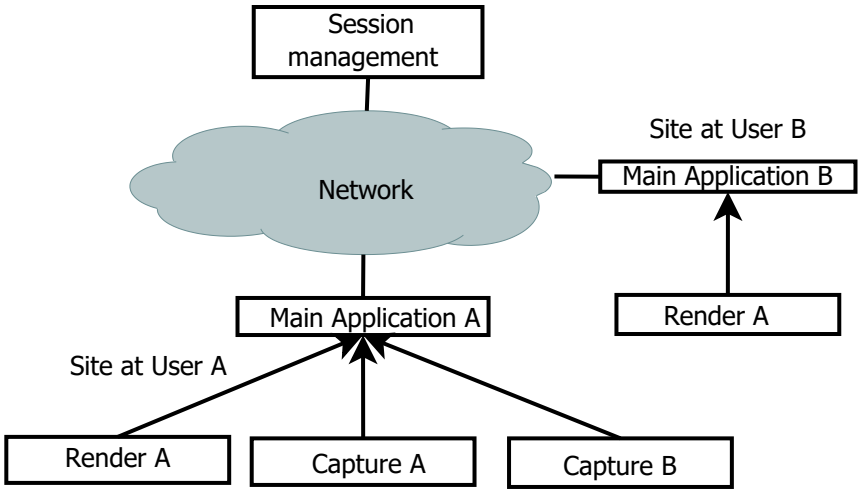


Figure 89 Modular 3D immersive Architecture with Terminal Scalability, end terminals load different types of modules

7.2.2 Real-Time Streaming Framework

The streaming framework for real-time data is illustrated in Figure 90. It is needed to develop a novel streaming framework based on custom transport protocols, as existing standard protocols and popular media streaming frameworks do not support the data types such as point clouds, meshes and 3D audio. One of the reasons is the standard protocols work with registered data types, as registered by internet assigned naming authority (IANA) (e.g. mostly audio and video defined on a rectangular grid [104]). For novel data types such numbers are not available, and protocol implementation would not work.

To adress this limitation, the framework illustrated in Figure 90 is developed that can handle different types of incoming and outgoing media streams (UDP/TCP) representing 3D data and real-time messaging. Critical for real-time communication of

natural users are pure TCP and UDP with real-time FEC as described in [86] and the previous chapters. For command and avatar messaging, the Publish and Subscribe paradigm based on web-socket, UDP and XMPP have been integrated. The media synchronization module operates independently of the incoming/outgoing streams and their specific implementation. It sits on a higher level and provides media synchronization services to the modules by keeping track of end-to-end latencies of the relevant streams. This allows modules to request target inter-sender and target inter-media skews and do synchronization based on a local play-out buffer on a best effort basis. To facilitate this, the streaming framework includes a virtual clock for global time synchronization based on a PTP like synchronization protocol and a function that allows receiver modules to report stream processing latency (i.e. decoding time).

To control admission of incoming and outgoing streams, the allowed streams table is managed by the session management protocol (this will be discussed in the next section). The monitor component keeps track of all the processing and network latencies that streams experience (i.e. compression, FEC, rendering, capturing, network delay etc.) which allows the system to detect anomalies and problems in the pipeline. The UDP Src and Sink Components handle the socket based network communication. For UDP streams based on [86], per packet decoding (progressive decoding) of the incoming packets needs to be performed, to achieve this efficient multi-threaded process in UDPSrc was implemented to simultaneously receive and decode packets. The API to the 3D tele-immersive framework is based on a simple pull/push frame API. The sender and receiver can simply push data to the network (send), or pull data from the network. This API includes both blocking and non-blocking methods for sending and receiving data. This allows modules to serve their specific needs.

The instance of the streaming framework resides in the main application, and can be used by the different render/capture modules. This allows modules that are otherwise unaware of each other to perform the synchronization of media streams and share a network service. The streams all follow an RTP like format where the payload_type specifies the specific administered payload type for the 3D module. The sequence number is only used by UDP and signals the ordering of the packet in the overall frame. The ssrc is a unique randomly generated identifier for the stream;

frame_id is the frame count number, timestamp the globally synchronized capture time of the frame (or an approximation of this). Source Id uniquely determines the sending host, The NC_header_size signals the existence of a forward error correction that contains additional information for packet FEC decoding. Session Id and Routing are reserved for distinguishing session and overlay routing.

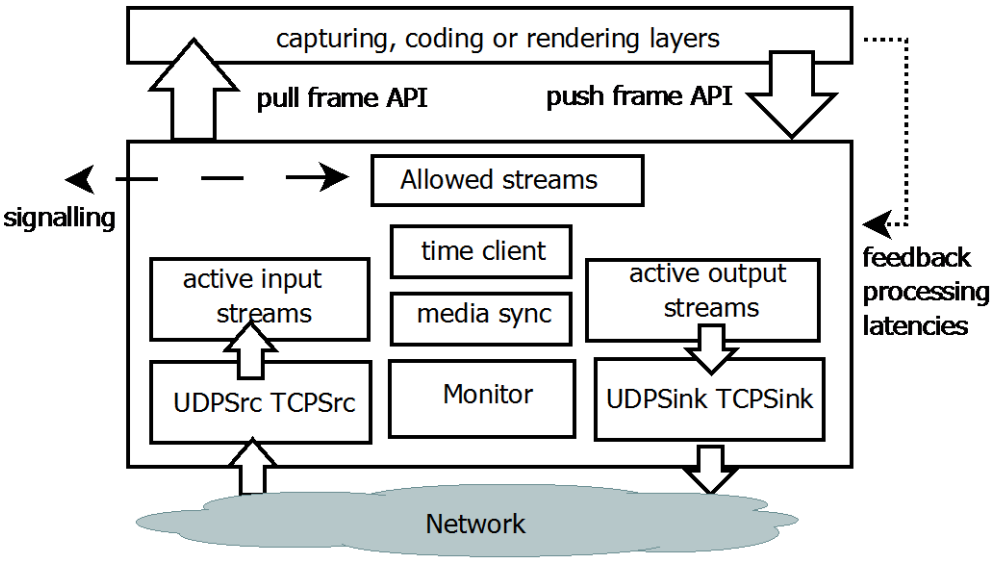


Figure 90 Data Streaming Framework Implementation

7.2.3 Basic Session Management Protocol and Implementation

The open source XMPP presence protocol was used to signal user presence, module availability and stream signalling. The XMPP 0030 Device Discovery extension was used to exchange capabilities of the 3D tele-immersive network entity (i.e. the preloaded modules and the related stream numbers). The login works as follows. Clients that log in provide their capture and render modules as discovery items which are triplets of the form <name, type, category>, where the category field is used to signal the module name and the type field for some additional parameters. Further additional XMPP messages have been defined to do the stream setup. The MEDIA_STREAMS was defined for requesting an overview of the available media streams, REQUEST_STREAMS to request a specific stream from a 3DTI terminal

and ACK_REQUEST_STREAMS to acknowledge or decline such requests. In Figure 91 shows the sequence diagram with two users. First TLS/SASL based authentication happens (part of XMPP). Once this is successful XMPP messages can be exchanged. The user's module configuration is exchanged to the server via discovery items (XMPP 0030). When another user logs in, XMPP signals presence and the server updates stream information in the MEDIA_STREAMS. This message lists the available 3D streams, (aggregated from the discovery information provided at login times by each user). In this case the second client requests the stream from client 1 first (this updates the allowed streams table in Figure 90). When client 1 receives this request it checks if it can send the stream, and sends an ACK_REQUEST_MESSAGE which indicates if the stream request was acknowledged or not. If it is, the media stream is transmitted via UDP or optionally TCP.

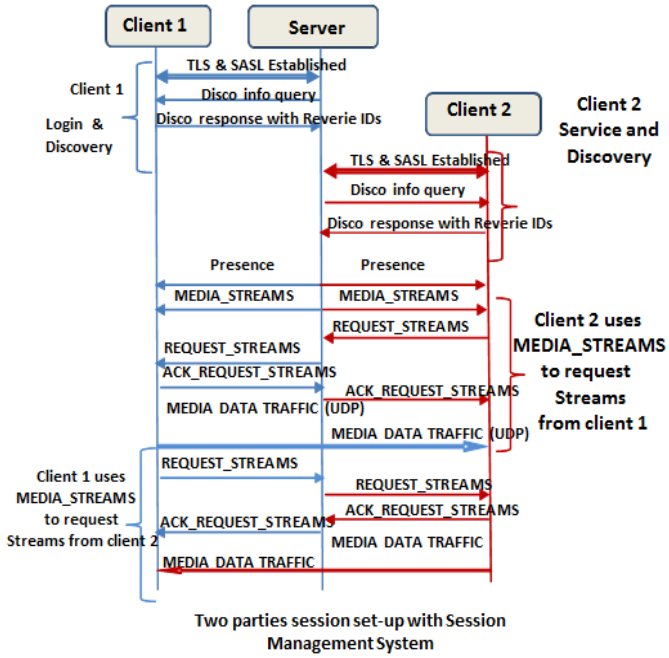


Figure 91 3DTI Session Management based on XMPP

Table 20 Common Header for TCP/UDP based streams

payload_type	Sequence number
Ssrc	Frame_id
Time_stamp	
Packet count	Routing id
Source id	Session id
Nc_header_size	xxxxxxxxxxxxx

7.2.4 AI and Avatar Commands messaging

To support small messages related to artificial intelligence and simple pre-programmed animations, experiments with 3 implementations of publish and subscribe paradigm based protocols have been performed.

The protocols implemented were the following.

- UDP based using the channels in the streaming engine (i.e. a dedicated publish and subscribe channel)
- XMPP based, using the XMPP server and the XMPP XEP 0060
- WebSocket based via a real-time cloud service (realtime.co)

From tests, the most reliable performance was achieved by using the web socket based implementation. The UDP based method suffered from occasionally lost messages, and the XMPP based implementation occasionally resulted in large delays.

The overall performance of the web socket messaging was tested as follows.

A test module was sending data via the main application outgoing message queue over the web socket to the cloud. The message is then again received in another module on another host. Both small messages (20 bytes) and large messages (500 bytes) have been tested at different frequencies from 10 Hz to 200 Hz. We utilized the clock synchronization service of the Real-Time Streaming module to measure

the latency of the message dissemination. We tested the performance in realistic conditions with other network traffic and system processes also running (tele-immersive system online). All delays were tested below 50 ms on average up to 100Hz, only at 200 Hz end-to-end delay becomes over 100 ms up to uncontrollable. To maintain performance, the outgoing message sending rate was limited to a maximum of 100 messages per second. This rate was sufficient to support the intended AI and animation data.

7.3 Experimental Evaluation

7.3.1 Natural User Transmission

The developed framework was integrated with capturing and rendering, enabling the multi-site transmission to be tested with 3 or more users. We select the geometry driven codec from Chapter 5 that gives better compression, resulting in smaller frame sizes and better frame rates. The experimental setup was deployed in a LAN network with 3 machines connected via a switch as shown in Figure 92. PC1 is a natural user reconstructed from 5 Kinects, PC2 is a natural user reconstructed from 1 Kinect and PC3 runs an avatar based synthetic user. The Machine and setup specifications are given in Table 21. We use the network emulator for Windows [105] to emulate losses, delays (10-100 ms) and jitter (10-50 ms, 50 % of the delay each time) on incoming packets on each site. Additionally the incoming bandwidth is limited to 100 Mbps. The PC 3 avatar user site is a relatively modest laptop computer, while PC1 and PC2 are more powerful machines. We deploy both UDP based plus FEC based on 4.2 and TCP. For TCP we apply rate control on the data by skipping late frames from the sender queue, (i.e. a last in first out policy towards the TCP socket, LIFO, dropping al skipped frames). For UDP transmission we fixed the sending frame rates. We measured delays from capture time to render time including codec delays.

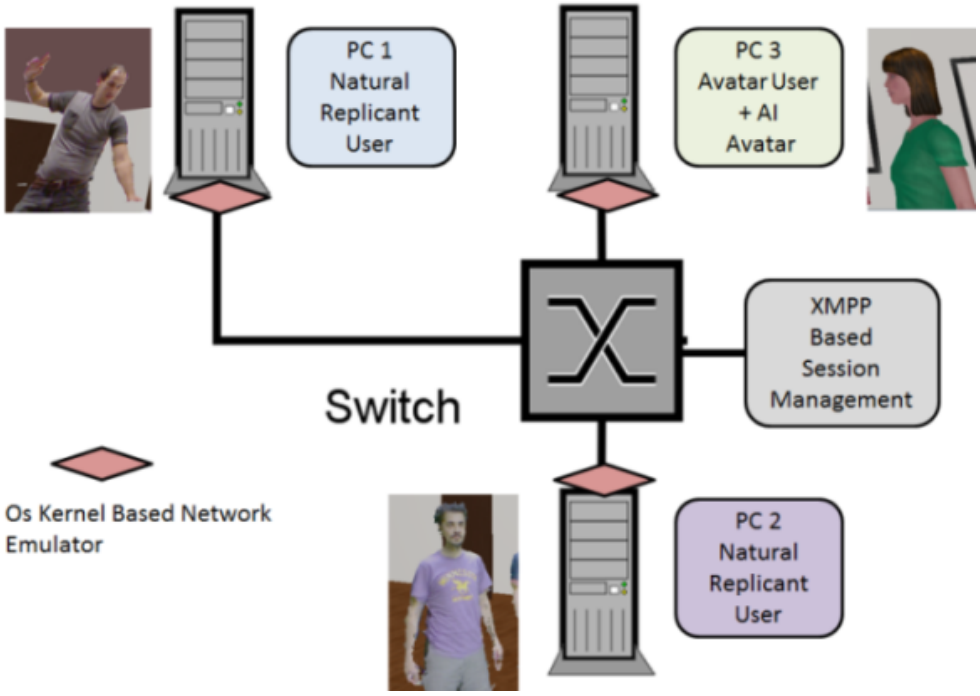


Figure 92 Experiment setup, 3 sites are connected via a switch and network impairments are introduced via a network emulator on incoming data at each site

Table 21 Machine and component specs

Component	**Specs**
PC 1	Desktop Intel i7 2,8 GhZ , 8 GB Ram, Win 7 64-bit, NVIDIA GeForce GTX 760
PC 2	Desktop Intel i7, 3,4 GhZ, 16 GB, Win 7 64 Bit, NVIDIA Geforce GTX 470
PC 3	Laptop Intel i7 , 1,6 GhZ, 4GB, Win7 64 Bit, AMD Radeon
Switch	NetGear GS 105 Gigabit Switch
Network Emulator	Network Emulator for Windows [105]

We have performed measurements across each site. Results were consistent and we only show the results for PC3 that is a modest laptop receiving both of the natural user streams (results were consistent across sites). The UDP based transmission limits the end-to-end delay to the computational and network latencies that could even be further reduced by implementation. On the other hand, transmission with TCP introduces delay in network conditions with loss and delay. In the current implementation both the 5 Kinect and 1 Kinect streams are received within 300 ms bounds for UDP as shown in Figure 93 and Figure 94. Also, the frame rates achieved for the received streams are shown in Figure 94. For UDP the achieved frame rates are stable in varying network conditions and at heterogeneous sites (contrary to TCP which was not shown here)

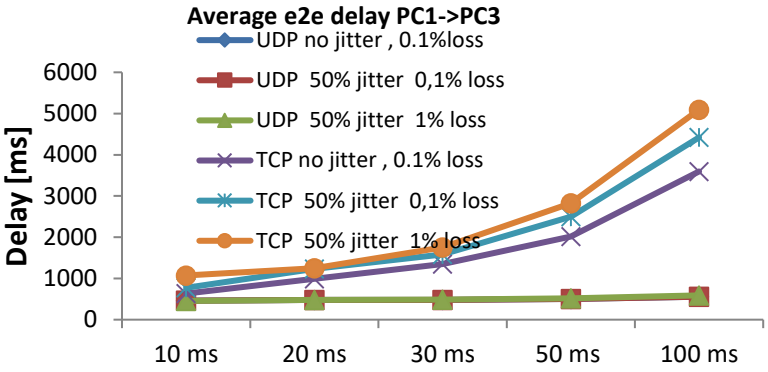


Figure 93 End-to-end delay of PC1 stream received at PC3

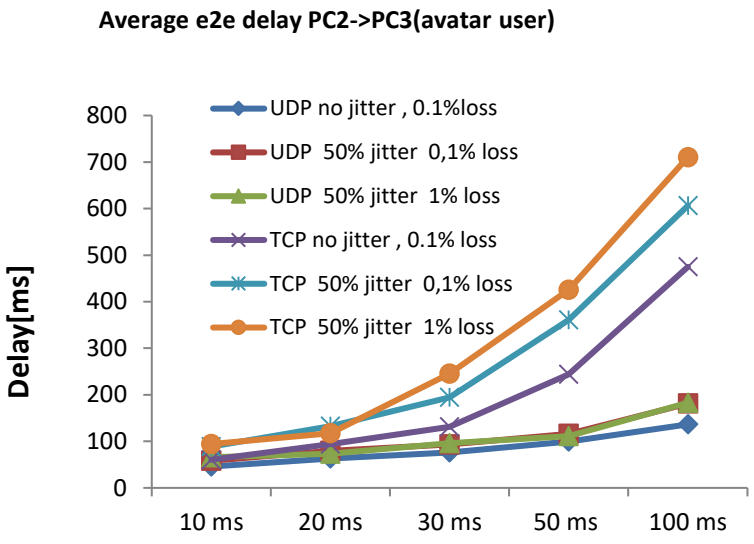


Figure 94 End-to-End delay of PC2 stream received at PC 3

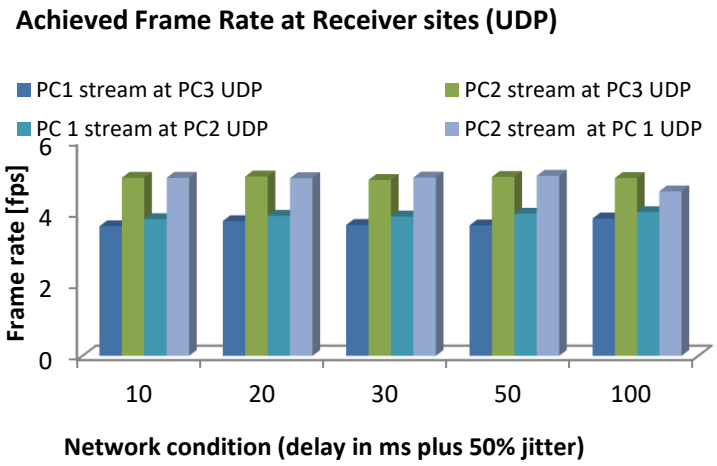


Figure 95 Achieved frame rates for each stream at each site

7.3.2 Frame Skew and Media Synchronization

Currently, support for media synchronization at the rendering tier is provided via play-out buffering. The basic support procedure is listed in Table 22. The streaming module keeps track of all the streams and their latency based on the synchronized timestamps from the virtual clock in the payload headers. As some incoming streams experience extra computational latencies before rendering (i.e mesh decoding), modules report back these times to the real-time streaming framework via a function (such that they can be accounted for). Modules can request the target sync latency, which is the extra time the module needs to wait before rendering frames to achieve the required sync with the other streams (as long as it is within real-time bounds 300 ms). Such waiting can be done using a play-out buffer implemented in the modules. The design of the play-out buffer for 3D audio was trivial, as based on the audio sample rate the frame buffer-size could be directly computed, enabling synchronization. We implemented a play-out buffer in a 3D Audio module that captures and transmits monophonic 44.1 KHz 16 bits PCM samples over TCP such that it can synchronize with the mesh streams. The opus wideband codec is used for compression of these samples[5].

For buffering meshes with possibly varying rates the situation was more challenging. To address the problem of synchronization with varying frame rates, we made continuous running estimates of the instantaneous frame-rate. Based on this the buffer is dimensioned to K mesh frames based on these estimates.

Table 23 details the achieved average media skews and deviations between audio and mesh data at PC1 and PC 2, and between incoming meshes from PC1 and PC2 at PC3. The measurements were done with PC1 sending 5 Kinect meshes at 6 fps, PC2 Sending meshes with 8 fps, 25 ms delay, 10 ms jitter and 0.1% packet loss. We report both the mean skew and the standard deviation to better capture the achieved synchronization quality. The implemented play-out buffers reduces the skew to negligible proportions.

[5] *https://www.**opus-codec**.org*

Table 22 Media Synchronization Support

MediaSync in Real-Time Streaming Framework Tracks all network stream delays
Modules provide their computational processing latency (decoder time etc.)
Modules request target latency (either per sender)
Modules based playout buffer provides mean synchronization

Table 23 Achieved Skews with Play-out Buffers

		Sync control Off		Sync control On	
skew type	Location	mean [ms]	Standard Deviation	Mean [ms]	Stddv
PC2-Mesh – PC2 3D Audio	PC1	32	28	2	16
PC1 5-K-Mesh PC1 3D Audio	PC2	150	35	10	26
PC1 5-K-Mesh - PC2 1 K -Mesh	PC3	178	25	33	18

7.3.3 Discussion and Conclusion

We described the design, implementation and evaluation of a practical network support system for modular social 3D tele-immersive interaction with natural and synthetic users. The session management system was deployed on top of XMPP and real-time communication based on publish/subscribe via web sockets was implemented. The architecture can be useful to 3D immersive communications in tele-immersive frameworks. For Example, user credentials from a social network portal can be easily imported. The API of the framework includes support for synchronization via a distributed virtual clock. Buffering of 3D audio and mesh data has led to approximate synchronization of audio and different 3D frames. In future work more effective adaptation could be studied.

Chapter 8 Conclusion

8.1 Achieved results

This thesis presented a contribution in the experimental field of distributed multimedia systems. It looked at the next generation distributed multimedia systems beyond current practice of photo and video sharing, interactive gaming and video conferencing. It envisions systems that blend techniques from 3D Graphics, 3D Computer Vision and 3D scanning, Artificial intelligence and Computer Networks to provide novel types of shared experiences. These type of shared experiences, in particular, mixed or tele-immersive virtual reality, are initiating a convergence between virtual and real worlds. Naturalistic, live recorded content is used seamlessly in a virtual world, including interaction with autonomous avatars and composite rendering. In a practical instantiation of such an application, a real user can step inside a virtual world, realistically represented as himself, and interact and communicate with other characters either human or computer controlled.

Based on this vision and to support such application, this thesis introduced the architecture of such a system in chapter 1 based on the larger scale REVERIE system. The components in this architecture were categorized in four categories: Rendering, Capturing, artificial Intelligence for autonomy, and Networking. The architecture illustrates the interconnection between these sub-blocks, and how such a system could be linked to social network services. All components are critical to achieve the envisioned mixed reality application, such as including state of the art rendering and 3D capture. However, the practical implementation introduced large challenges to the networking and compression components. As these could not be addressed with currently available tools and techniques, the main part of the thesis focussed on overcoming these bottlenecks.

Based on these challenges, this thesis aimed, from the beginning on, to extend the state of the art in compression and networking for data types used in the chosen 3D Virtual Reality application. The thesis implemented new codecs, comparing them to standardized and open source implementations on various aspects of importance to the intended application. Particularly, it compared compression performance (rate-distortion), latency characteristics (encoding time, decoding time, parallelizability) and subjective quality assessment. The work revealed the gap in existing codecs to

support 3D data suitable for immersive virtual reality, (typically based on point clouds and meshes). In particular, the work in the 3D graphics community on mesh and point cloud compression did not take into account realistic system constraint such as real-time, low memory footprint etc. In addition, it did not address important aspects such as inter-frame and attribute coding well. Our work in chapters 3,4,5 and 6 has shown various improvements over the state of the art in terms of real-time encoding at the same rate-distortion (chapter 3 and chapter 4), highly adaptive coding with bit-rate control (chapter 5), and lossy attribute and inter predictive coding (chapter 6). While 3D compression is a challenging field, and many further improvements are expected once the use case is well understood, these contributions constitute a first step in the direction of supporting point cloud and mesh compression for tele-immersive virtual reality.

The importance of the applications and the need to reconsider codec design towards this aim has been recognized by the industry. The codec developed in chapter 3 was invited for presentation at the 104th MPEG meeting in Incheon, South Korea. This eventually led to the consideration to update existing MPEG Mesh codecs with the techniques developed in this chapter. This work led to new core experiments and subsequently exploration and evidence that a new type of 3D coding to be developed in MPEG (to support tele-immersive virtual reality). In addition, the compression work in chapter 4 was invited for presentation at the IEEE Conference on Audio Signal Speech processing (ICASSP), by the research lab of Microsoft research, Redmond. Since, this leading research lab has been investigating this topic and contributed papers on point cloud and mesh compression for tele-immersive virtual reality to leading Journals and Conferences. This renewed industry and standardisation interest will likely spark a new wave of research in point cloud and 3D mesh compression. The work in this thesis has identified many of the research challenges and requirements that such work could focus on.

Beyond compression, this work also identified the need for real-time transmission of point cloud and mesh compression. In chapter 3 and 4 approaches based on UDP and linear rateless coding with progressive decoding have been integrated for experimentation. Experimentation showed that such techniques can be used to achieve error resilient real-time communication of 3D mesh and point cloud data. In addition, in chapter 7 we integrated a synchronization layer and session management layer for

tele-immersive networking. It showed how tele-immersive sessions can be signalled using extensions on top of existing protocols such as XMPP. In addition, it addressed some of the challenges on synchronization issues (inter-stream, inter-sender) in a multi- site environment, enabling approximate synchronization of variable framerate streams.

Based on systems deployment, some initial subjective tests on using realistic point clouds and meshes in virtual environment have been performed in chapter 5 and 6. While the results have not been extensive and conclusive, in general they all point towards the benefits of using naturalistic point cloud and meshes in tele-immersive virtual reality. In future work, more extensive user studies are needed to study the user experience more closely. Nevertheless, our experiments hinted that these representations provide an increased feeling of remote tele-presence to users and are beneficial.

The transmission and quality of experience aspects could have been addressed more extensively in this thesis. On the other hand, the work on compression of point clouds and 3D meshes has made both a large impact on the 3D standards and scientifically. Both MPEG and JPEG are currently working on new standards that include point cloud and possibly mesh compression. The activity in MPEG has been driven from contributions in this work, the author has led the Ad Hoc Group on 3D graphics compression, with the mission to add point cloud compression to the MPEG tools. In addition, the author has contributed to the requirements for a new JPEG standard on plenoptic image compression that also aims to include point clouds as an underlying representation. The next section will provide more details on the standardisation work that resulted from this thesis work. In the rest of this section we provide a summary of the research questions that were addressed in this Thesis.

Main Research Question: How can we support Real-Time (<300ms) end-to-end media streaming for Mixed Reality and 3D Tele-immersion be supported in the current Internet with state of art compression rates ?

This research question has been addressed throughout this thesis with the development of compression and transmission in Chapters 3, 4, 5 and 6 and their integration in the mixed reality system.

Research Question 2: How can we achieve Real-time (< 300 ms on commodity hardware) compression and transmission of highly realistic reconstructed 3D humans based on meshes with State of Art Compression rates (i.e. MPEG-4)?

This research question has been addressed by developing real-time compression of triangle meshes based on block based quantization (chapter 3) and late differential quantization (chapter 3) and pattern based connectivity coding (described in chapters 3 and 4). Together with a transmission scheme based on LT coding, this achieved < 300 ms end to end delays in a practical prototype under these constraints.

Research Question 3: How can we achieve highly adaptive (lossy) real-time compression and transmission of highly realistic 3D replicants based on meshes that can be used for many different bit-rates and level of detail (i.e. a geometry coding with a large range of bit-rates)?

The highly adaptive geometry codec was presented in Chapter 5, the results have shown that contrary to the other codecs, this codec can achieve lossy compression with controlled bitrates (by changing the number of octree levels). For geometry compression this is preferable over simple quantization that does not provide sufficient granularity to achieve many different bit-rates.

Research Question 4: How do we design improved real-time Compression of 3D Time varying Point Clouds with improved lossy colour coding and inter-prediction with a negligible perceptual distortion (compared to the current state of the art in practical real-time point cloud compression)?

This research question was fully addressed in Chapter 6, the source code has been made available as open source and the implementation is used as a starting point for point cloud compression in MPEG. Main contributions are inter predictive coding, lossy colour coding and efficient parallelization.

Research Question 5: how can we design a 3D media streaming engine for heterogeneous 3D tele-immersive communications that is compatible with the current internet infrastructure (i.e. existing transport and signalling protocols)*?*

This research question as addressed in Chapter 7, we developed the complete streaming engine and developed basic support for synchronization and session management and signalling. We expect that once the compression technologies will be more advanced, more research in this area of multimedia streaming of points cloud and meshes in 3D tele-immersive scenarios will be performed. This work can serve as a starting point for such future work.

8.2 Standardisation Contributions

The work presented in this thesis deals with an area of application that, while not common today, could be a commodity to millions of users in the future when technology develops. For technologies deployed at such a large scale, interoperability between devices and services and standards become of critical importance. Taking this into account, the experimental work in this thesis has served as an exploration for future media compression standards in this area. The exploration served to distil the requirements, use cases, develop preliminary codecs (to be used as reference or anchor) and collect relevant datasets. Based on this preliminary work, it will then be possible to start a standardisation phase, where technologies can be contributed by prospects and technologies can be selected based on pre-defined evaluation criteria.

The specific standardisation body this thesis work contributed to is the Moving Picture Experts group[6]. The Moving Picture Experts Group is a consortium of industrial and academic partners that develop international standards under the international standardisation organization (ISO) related to multimedia systems. The benefit of this collaboration is that state of art insights from academia can be incorporated quickly in products developed by companies, while maintaining interoperability.

Current standards developed by this group are heavily used in practice. Examples include MPEG-4 advanced audio codec (AAC) for audio, MPEG-4 AVC (advanced video coding) for DVD and internet video, MPEG-2 Transport Stream (digital TV), MPEG-4 ISOBMFF (mp4 file format), and MPEG DASH (dynamic adaptive streaming over http for internet video). The group is always looking to expand its

[6] The Moving Picture Experts group mpeg.chiariglione.org

portfolio of standards addressing new applications and promising areas. Therefore, the work performed in this thesis was very relevant for MPEG. We have contributed to this consortium since April 2013, when our paper, a 3D tele-immersion system based on live reconstructed geometry was invited for presentation at MPEG 104 in Incheon South Korea by the chair of the MPEG group on 3D Graphics. This work has led to core experiments, and inclusion of a use case document for tele-immersive mesh and point cloud compression. The core experiments have been used to provide evidence for the usefulness of developing the standards. Therefore the MPEG AhG on 3D graphics compression was established in January 2014 with mandate to add tele-immersive coding tools to the MPEG compression tools. The requirements of the MPEG-4 standard have been updated accordingly in July 2014. The first anchor and evaluation software for point cloud compression has been contributed to MPEG in July 2015. In June 2016 all requirements, use cases, test data have been presented as public output documents and a draft call for proposals for point cloud compression was issued. This is the starting point of the standardisation of point cloud compression in MPEG, which has raised interest from many large companies (Microsoft, Qualcomm, Technicolour, Huawei, Canon, Mitsubishi Electric, Samsung Electronic and MediaTek Electronics). While it is still to be seen if the MPEG point cloud standardisation effort will be successful, this thesis has provided the ground work for exploration, the call for evidence and the call for proposals to enable this standardisation effort. This standardisation can potentially have a very big impact on the media and technology industry. As of october2016, the point cloud compression standard has been added under a new part of MPEG referred to as MPEG-I that deals with the development of technologies for immersive communications and VR. MPEG-I also includes other standards for 360 degree video and 3D audio. For completeness, we illustrate the standardisation timeline in Figure 96. We provide an overview of all standardisation contributions in the Appendix B. Besides MPEG, the work in this thesis has also developed the standardisation work in the Joint Picture Experts Group (JPEG). This group has started an activity on plenoptic image compression (PLENO) that also intends to include point cloud compression. While in this case point clouds are more used as an underlying sparse representation for plenoptic image data, still some of the requirements for 3D tele-immersive have been taken into consideration in its requirements scope. The author has presented some of the work from Chapter 6 in the inaugural workshop on JPEG PLENO in July 2015.

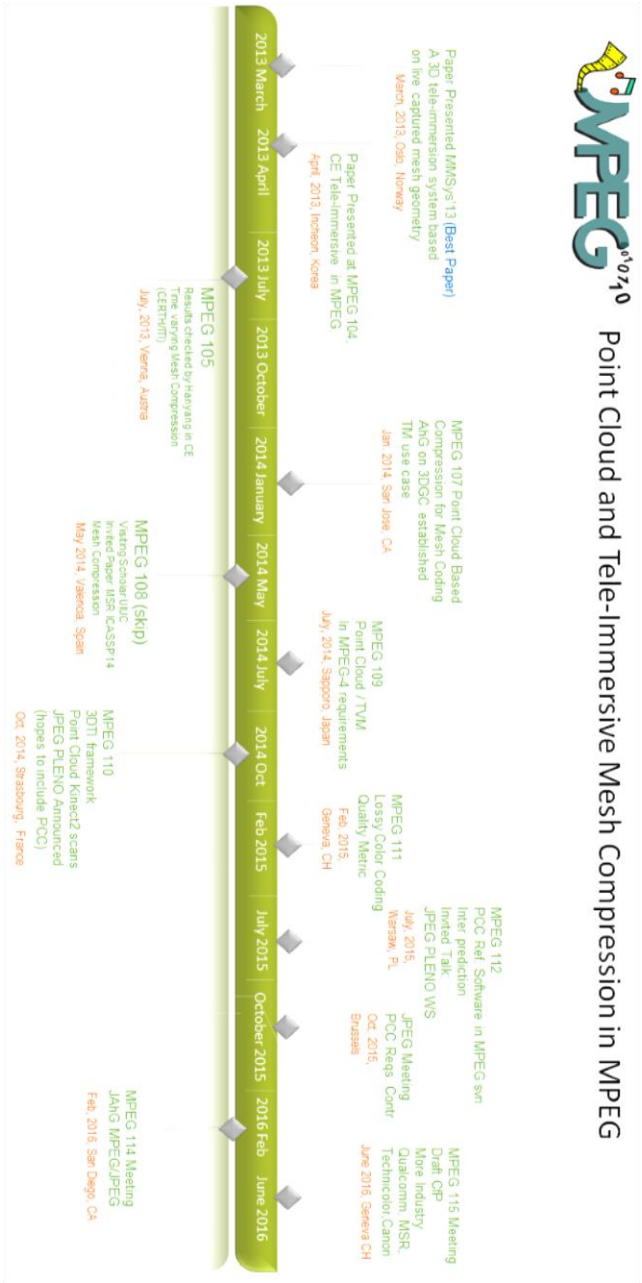

Point Cloud and Tele-Immersive Mesh Compression in MPEG
2013 March
2013 April
2013 July
2013 October
2014 January
2014 May
2014 July
2014 Oct
Feb 2015
July 2015
October 2015
2016 Feb
June 2016
Paper Presented MMSys'13 (Best Paper)
A 3D tele-immersion system based
on live captured mesh geometry
March, 2013, Oslo, Norway
Paper Presented at MPEG 104,
CE Tele-Immersive in MPEG
April, 2013, Incheon, Korea
MPEG 105
Results checked by Hanyang in CE
Triangle-only Mesh Compression
(CE TriMTF)
July, 2013, Vienna, Austria
MPEG 107 Point Cloud Based
Compression for Mesh Coding
AhG on 3DGC established
TM use case
Jan 2014, San Jose, CA
MPEG 108 (Skip)
Visiting Scholar UIUC
Invited Paper MSR ICASSP'14
Mesh Compression
May 2014, Valencia, Spain
MPEG 109
Point Cloud / TVM
In MPEG-4 requirements
July, 2014, Sapporo, Japan
MPEG 110
3DT framework
Point Cloud Kinect2 scans
JPEG PLENO Announced
(hopes to include PCC)
Oct, 2014, Strasbourg, France
MPEG 111
Lossy Color Coding
Quality Metric
Feb, 2015,
Geneva, CH
MPEG 112
PCC Ref. Software in MPEG svn
Inter prediction
Invited Talk:
JPEG PLENO WS
July, 2015,
Warsaw, PL
JPEG Meeting
PCC Reqs Contr
Oct, 2015,
Brussels
MPEG 114 Meeting
JAhG MPEG/JPEG
Feb, 2016, San Diego, CA
MPEG 115 Meeting
Draft CfP
More Industry
Qualcomm, MSR,
Technicolor,Canon
June 2016, Geneva CH

Figure 96 Timeline of contributions on Point Cloud and Tele-immersive mesh compression

8.3 Future development

While the work presented in this thesis might be a big step for the author, it is perhaps a small step in the direction of supporting tele-immersive and point cloud coding efficiently in tele-immersive systems. The reason is that the subject is interdisciplinary, and some of the underlying theory on some of the underlying technologies that are needed to achieve good performance are not yet available.

Perhaps the main problem is that powerful methods from signal processing, such as transform coding, up and down sampling, quantization etc. are not available for unstructured data such as point clouds or 3D meshes. This left us to use relatively simple quantization and octree based coding schemes and heuristic techniques.

The development of new signal processing techniques for unorganized data is highly desirable to achieve better direct compression of point cloud and mesh data. Research that could fit this can be found in the emerging field of signal processing on graphs [106]. This work presents down/up sampling and Fourier transforms for graph signals. Point clouds could be converted to graphs and then, in this case the defined signal operations on graphs could be re used. Nevertheless, this needs a lot of fundamental research in signal processing to define the fundamental techniques. Subsequently, these techniques then need to be applied for compression and implemented in intelligent ways to enable real-time processing and lossy signal representations.

Another strategy for point cloud compression that could be explored is mapping parts of the cloud to a texture projected on a surface. This could then allow reuse of existing video codecs such as AVC, HEVC. This approach might give highly efficient results, but will probably have some limitations in practice. Existing coding technologies for image and video are well developed, and from a practical viewpoint reuse of codecs is desirable. Nevertheless, it is might be very application specific how such an approach would work in practice.

The point cloud codec developed in chapter 6 currently served as a starting point for point cloud compression in MPEG. This codec could benefit from better transform coding, such as based on [89], inter predictive coding, scalable coding and other extension such as better region/view selectivity. As the codec has been contributed to MPEG already some extensions have been suggested, such as based on flatness based plane projection (early termination of the octree and coding the resulting points as projection on plane) proposed by University of Missouri and Qualcomm, or a Haar modified attribute coder, such as proposed by Microsoft Research Redmond. An important next step will be to explore and compare these approaches to the current implementation.

Another important topic to explore would be related to the streaming of compressed point cloud data. As HTTP based streaming of video is becoming increasingly popular, Adaptive HTTP based streaming of point clouds deserves attention. Adaptive streaming over HTTP would make it easier to deploy large scale distribution in the internet.

Last, more work on subjective testing, rendering and comparison of different datatypes in different applications will be useful. This could also help in defining better objective and subjective methodologies to evaluate point cloud compression, and shed some light on the differences between representations and rendering methods.

References

[1] J. Peng, C.S. Kim, C.-C. Jay Kuo , "Technologies for 3D mesh compression: A survey," *Journal of Visual Communication and Image Representation*, vol. 16, no. 6, pp. 688-733, December 2005.

[2] D. Alexiadis, D. Zarpalas, and P. Daras.,., "Real-Time, full 3-D reconstruction of moving foreground objects from multiple consumer depth cameras," *IEEE Transactions on Multimedia*, vol. 15, pp. 339-358, 2013.

[3] P., Kolev, K., Meier, L., Camposeco, F., Saurer, O. Pollefeys, M., Tanskanen, "Live Metric 3D Reconstruction on Mobile Phones," in *ICCV*, Sydney, 2013.

[4] google inc. (2015, Nov.) google tango. [Online]. https://www.google.com/atap/project-tango/

[5] MPEG. (2016, Jan.) The Moving Picture Experts Group. [Online]. http://mpeg.chiariglione.org/

[6] JPEG. (2016) Joint Picture Experts Group. [Online]. https://jpeg.org/

[7] REVERIE FP7. (2015, Aug.) reverie fp7. [Online]. http://www.reveriefp7.eu/

[8] MPEG, "ISO/IEC 14496 MPEG-4," MPEG, International standard.

[9] JPEG. (2015, March) JPEG PLENO Abstract and Executive Summary. [Online]. https://jpeg.org/items/20150320_pleno_summary.html

[10] Reverie. (2016, Jan.) Reverie FP 7. [Online]. www.reveriefp7.eu

[11] J. Ostermann, "MPEG-4 requirements (Sapporo Revision)," MPEG, MPEG Output Documents n14662, 2014.

[12] R.N. Mekuria, M.Preda L Bivolarksy, "Tele-immersion use case and associated draft requirements," MPEG, MPEG Output Document n14197, 2014.

[13] AhG on 3D Graphics, "Current status on Point Cloud Compression (PCC)," Geneva, MPEG Output Document n15869, 2015.

[14] Mekuria R.N., "Point Cloud Compression," in *JPEG PLENO Workshop Proceedings, ISO/IEC JTC1/SC29/WG1, wg1n69022*, Warsaw PL, 2015.

[15] D. Alexiadis, D. Zarpalas, P. Daras , "Real-Time, full 3D Reconstruction of moving foreground objects from multiple consumer depth cameras," *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 339-358, Februari 2013.

[16] K. Mamou. , T. Zaharia, and F. Preteux, "FAMC: The MPEG-4 standard for Animated Mesh Compression," in *15th IEEE International Conference on Image Processing, 2008. ICIP 2008.*, San Diego, 2008, pp. pp.2676,2679.

[17] A. Aydin Atalan, P. Benzie, N. Grammalidis, S. Malassiotis, J. Ostermann, S. Piekh, V. Sainov, C. Theobalt, T. Thevar, X. Zabulis E. Stoykova, "3-D Time Varying Scene Capture Technologies - A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 11, pp. 1568-1586, 2007.

[18] G. Saygili, "Depth Extraction, Refinement And Confidence Estimation From Image Data," Delft, 2015.

[19] C. Zhang, T. Chen , "A Survey on Image Based Rendering, representation, samping and compression," Pitsburgh, 2003.

[20] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and Andre Fitzgibbon , "Real-time 3D reconstruction and interaction using a moving depth camera," in *UIST*, Charlotte, NC, 2011.

[21] M. Zollhöfer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M.Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, and M. Stamminger , "Real-time non-rigid reconstruction using an RGB-D camera," *ACM Trans. Graph.*, vol. 33, no. 4, 2014.

[22] D. Zarpalas, P. Daras D. Alexiadis, "Fast and smooth 3D reconstruction using multiple RGB-Depth sensors," in *IEEE Visual Communications and Image Processing*, Valetta, 2014.

[23] G. Sellers, R. S. Wright, N. Haemel , *OpenGL SupeBible, sixth edition*.: Addison Wesley, 2015.

[24] Microsoft. (2016, Jan.) Direct 3D. [Online]. https://msdn.microsoft.com/en-us/library/windows/desktop/hh309466(v=vs.85).aspx

[25] Khronos. (2016, Jan.) WebGL 1.0 specification. [Online]. https://www.khronos.org/registry/webgl/specs/1.0/

[26] Nicolas Devere. (2016, Jan.) Java GL. [Online]. http://javagl.sourceforge.net/

[27] (2016, Jan.) PyOpenGL 3.0. [Online]. http://pyopengl.sourceforge.net/

[28] I. Sodagar, "The MPEG DASH Standard for Multimedia Streaming over the Internet," *IEEE Multimedia*, vol. 18, no. 4, pp. 62-67, October - December 2011.

[29] WebRTC. WEbRTC. [Online]. https://webrtc.org/

[30] H. Schulzrinne et al., "RTP: A Transport Protocol for Real-Time Applications RFC 3550," IETF, Request for Comments 2003. [Online]. https://tools.ietf.org/html/rfc3550

[31] K. Nichols et al., "Definition of the Differentiated Services Field (DS Field) RFC 2474," IETF, Request for comments RFC 2474, 1998.

[32] D. Clark., S. Shenker R. Braden., "Integrated Services in the Internet Architecture: an Overview," ietf, Request for comments RfC 1633, 1994.

[33] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner , "OpenFlow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69-74, 2008.

[34] H. E Egilmez, S. Civanlar, A. M. Tekalp , "An optimization framework for qos-enabled adaptive video streaming over OpenFlow networks," *IEEE Transactions on Multimedia*, no. 99, pp. 1-1, 2013.

[35] Hilmi E Egilmez, "Distributed QoS architectures for multimedia streaming over software defined networks," *IEEE Transactions on Multimedia*, vol. 16, pp. 1597-1609, 2014.

[36] E. Nygren, R. K. Sitaraman, and J. Sun , "The Akamai network: a platform for high-performance internet applications," *SIGOPS Oper. Syst. Rev.*, vol. 3, pp. 2-19, August 2010.

[37] D. D. Clark and D. L. Tennenhouse, "Architectural considerations for a new generation of protocols," in *Proceedings of the ACM symposium on Communications architectures & protocols (SIGCOMM '90).*, New York, 1990, pp. 200-208.

[38] R.Pantos, "HTTP Live Streaming," Informational Internet draft 2015.

[39] Microsoft. (2008) Smooth Streaming. [Online]. http://www.iis.net/downloads/microsoft/smooth-streaming

[40] A. Smolic, "3D Video and Free Viewpoint Video - From Capture to Display," *Elsevier Pattern Recognition*, vol. 44, pp. 1958 - 1968, 2011.

[41] T. Wiegand, G. Sullivan, G. Bjontegaard, G. and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560-576, 2003.

[42] G.J. Sullivan, J. Ohm, Woo-Jin Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649-1668, 2012.

[43] Saverio Blasi, "High Efficiency Prediction Methods for Current and Next Generation Video Coding," London, UK, April 2014.

[44] P. Schröder, and W. Sweldens. A. Khodakovsky, "Progressive geometry compression," in *ACM SIGGRAPH*, 2000.

[45] Philip A. Chou, Yinpen Chen Ha Q. Nguyen, "Compression of Human Body Sequences Using Graph Wavelet Filter Banks," in *IEEE ICASSP*, Florence, Italy, 2014.

[46] A. Smolic., M. Kautzner., T. Wiegand K. Mueller, "Rate-Distortion Optimization in Dynamic Mesh Compression," in *ICIP*, 2006.

[47] Gauthier Lafruit, Leonardo Chiarglione R.N. Mekuria, "3D Visual Capture Inventory," MPEG, MPEG input document m37050, 2015.

[48] MPEG Requirements, "3D Audio Visual Capture Model," MPEG, geneva CH, MPEG output document ISO/IEC JTC 1/SC 29/WG 11 W15737, 2015.

[49] W. Chen, R Yang S. Kum H. Fuchs H. Towles, "3D Tele-collaboration over interent2," in *International workshop on tele presence*, Juan les Pins, 2002.

[50] D.E. Ott and K. Mayer Patel, "Coordinated multi-streaming for 3D tele-immersion (2004)," in *ACM International Conference on Multimedia*, 2004, pp. 596-603.

[51] Z. Yang., W. Wu K. Nahrstedt., G. Kurillo , "Enabling muli-party 3D tele-immersive environments with ViewCast," *ACM Transactions on Multimedia Computing and Communications*, vol. 6, no. 2, 2010.

[52] W. Wu., K Nahrstedt, R. Rivas, A. Arefin Z. Huang, "SyncCast: Synchronized dissemination in multi-site interactive 3D tele-immersion (2011)," in *ACM Multimedia Systems Conference (MMSys'11)*, 2011, pp. 69-80.

[53] K Kurillo, E. Lobaton, T. Bernadin, O. Kreylos, R. Bajcsy, K Nahrstedt R. vasudevan, "High Quality Visualization for Geographically Distributed 3D Tele-immersive Applications," *IEEE Transactions on Multimedia*, vol. 13, no. 3, pp. 573-584, 2011.

[54] A Arefin, G. Kurillo, P. Argawal, K Nahrstedt W Wu, "Color plus depth level of detail in 3D Tele-immersive video," in *ACM International Conference on Multimedia*, 2011, pp. 13-22.

[55] Z. Yang, B yu, R. Diankov., W. Wu, R. Bajcy , "Collaborative dancing in Tele-immersive environment," in *ACM International Conference on Multimedia*, 2006, pp. 723-726.

[56] R. Bajcy, O. Kreylos., R. Rodriguez G. Kurillo, "Tele-immersive environment for remote medical collaboration," in *Medicine meets virtual reality*, pp. 148-150.

[57] G. Kurillo M. Forte, "Cyberacrcheology - Experimenting with Tele-immersive Archeology," in *International Conference on Virtual Systems and Multimedia*, 2010, pp. 155-162.

[58] Nahrsted et al., "Symbiosis of Tele-immersive Environments with Creative Choreography," in *ACM Workshop on Supporting Creative Acts beyond dissemination*, 2007.

[59] W. Tsang Ooi., S. mondet., R. Grigoras G. Morin W. Cheng., "Modelling progressive Mesh Streaming: Does data dependency matter ?," *ACM Transactions on Multimedia Computing Communications and Applications*, vol. 7, no. 2, p. 24, March 2011.

[60] G. Al Regib Y Altunbasak, , "An application-layer protocol for streaming 3-D graphics," *IEEE Transactions on Multimedia*, vol. 7, no. 6, pp. 1149-1156, 2005.

[61] M. Li, B. Prabhakaran H. Li, "Middleware for streaming 3D progressive meshes over lossy networks," *ACM Transactions on Multimedia Comp. Comm. and Applications (TOMMCAP)*, vol. 7, no. 6, pp. 282-317, November 2006.

[62] T. Gotsman., C. Touma, "Triangle Mesh Compression," in *ACM SIGGRAPH*, 1998, pp. 26-34.

[63] J. Rossignac., R. Pajarola., "Compressed progressive meshes," *IEEE Trans. Vis. Comp. Graph.*, vol. 6, no. 1, pp. 79-93, 2000.

[64] P. Lindstrom M. Isenburg, "Streaming Meshes," in *IEEE Visualisation*, 2005, pp. 688-733.

[65] Lee S., B., Kim D. son Jang E.S., "Fast 3D Mesh Compression using shared vertex analysis," *ETRI Journal*, vol. 32, no. 1, february 2010.

[66] d. Santa Cruz., T. Ebrahimi N. Aspert., "MESH: Measuring Error between Surfaces using the Haussdorff distance (2002)," in *IEEE International Conference on Multimedia and Expo*, 2002, pp. 705-708.

[67] R. Mekuria, D. Alexiadis, P. Daras, P. Cesar , "Real-time Encoding of Live Reconstructed Mesh Sequences for 3D Tele-immersion," in *Proceedings of the International Conference on Multimedia and Expo. International Workshop on Hot Topics in 3D*, 2013, pp. 1-6.

[68] J. Kannala, J Heikkila D. Herrera, ""Joint depth and color camera callibration with distortion correction"," *IEEE Trans. on Pattern Anal and Machine Intelligence (PAMI)*, vol. 34, no. 12, 2012.

[69] W. Strasser S. Gumhold., "Real-Time Compression of triangle mes connectivity," in *ACM SIGGRAPH*, 1998, pp. 133-140.

[70] F. Prêteux., T. Zaharia K.Mamou., "TFAN: A low complexity 3D Mesh Compression Algorithm," *Computer Animation and Virtual Worlds*, vol. 20, no. 12, pp. 343-354, June 2009.

[71] A. khodakovsky, P Schroeder., W. Sweldens , "Progressive Geometry Compression," in *ACM SIGGRAPH'00*, 2000.

[72] J. Rossignac G. Taubin., "Geometric Compression through topological Surgery," *ACM Trans. Comput. Graphics*, vol. 17, no. 2, pp. 84-115, 1998.

[73] R.N. Sanna, M. Asioli,S. Izquierdo, E. Bulterman, D.C.A. and Cesar Mekuria, "A 3D Tele-Immersion System Based on Live Captured Mesh Geometry," in *ACM Conference on Multimedia Systems (MMSys 2013)*, Oslo, 2013.

[74] M. Luby, M. Mitzenmacher and A. Rege J.W. Beyers, "A digital fountain approach to reliable distribution of bulk data," *ACM SIGCOMM Compt. Commun. REv*, vol. 28, no. 4, pp. 56-67, 1998.

[75] A Shokrollahi, M. Watson, T. Stockhammer M. Luby, "RFC 5053: Raptor forward error correction scheme for object delivery," RFC 5053 2007.

[76] M. Watson., T. Stockhammer M. Luby A. Shokrallahi, "RFC 6330: RaptorQ Raptor Forward Error Correction Scheme for Object Delivery," IETF, RFC 633-, 2011.

[77] Wu Y., Jain K. P.A. Chou., "Practical Network Coding," in *Allerton Conference in Communication, Control and Computing*, Monticello, IL, Oct. 2003, 2003.

[78] R. Yeung, N Cai S. Li., "Linear Network Coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371-381, 2003.

[79] M.Levoy, G.Turk and , "Zippered polygon meshes from range images," in *21st annual conference on Computer graphics and interactive techniques (SIGGRAPH '94)*, 1994, pp. 311-318.

[80] M. Gailly J.L. Adler., "zlib, A Massively Spiffy yet Delicately Unobrtrusive Compression Library," in *http://zlib.net*, p. 2013.

[81] D. Alexiadis, D. Zarpalas, and P. Daras., , "Real-Time, full 3-D reconstruction of moving foreground objects from multiple consumer depth cameras," *IEEE Transactions on Multimedia*, vol. 15, pp. 339-358, 2013.

[82] D. Alexiadis. (2013, Oct.) Datasets of Multiple Kinects-based 3D reconstructed meshes. [Online]. http://vcl.iti.gr/reconstructions/

[83] N. Aspert, D. Santa-Cruz, T. Ebrahimi, , "MESH, Measuring Error between Surfaces using hte Haussdorff distance," in *IEEE International conference on media and expo*, 2002.

[84] Institut Telecom. (2013) mymultimediaworld.com. [Online]. mymultimediaworld.com

[85] B. Jovanova, M. Preda, F.Preteux, , "MPEG-4 Part 25: A graphics compression framework for XML-based scene graph formats," *Signal Processing: Image Communication*, vol. 24, no. 1/2, p. 101/114, January 2009.

[86] R.N. Mekuria, M. Sanna, E.Izquierdo, D.C.A. Bulterman, P.S. Cesar , "Enabling Geometry Based 3D Tele-immersion with Fast Mesh Compression

and Linear Rateless Coding," *IEEE Transactions on Multimedia*, vol. 16, no. 7, pp. 1809-1820, November 2014.

[87] Microsoft Research Asia. (2010, january) Network Emulator Toolkit (NEWT). [Online]. https://blog.mrpol.nl/2010/01/14/network-emulator-toolkit/

[88] ITU, "G.1050 : Network model for evaluating multimedia transmission performance over Internet Protocol," ITU, Geneva, Recommendation G.1050 E,.

[89] D. Florencio, C.Loop C. Zhang, "Point cloud attribute compression with graph transform," in *IEEE ICIP*, Paris, France, 2014, pp. 2066-2070.

[90] R. Mekuria, P. Cesar , "A Basic Geometry Driven Mesh Coding Scheme with Surface Simplification for 3DTI.," *MULTIMEDIA COMMUNICATIONS TECHNICAL COMMITTEE e-letter*, vol. 9, no. 3, pp. 6-7, May 2014.

[91] ITU-T, "Methods for objective and subjective assessment of speech quality. Mean opinion score interpretation and reporting," ITU Recommendation P.800.2, 2013.

[92] G. Charness, U Greezy, U., Kuhnc, M.A.., "Experimental methods: Between-subject and within-subject design," *Journal of Economic Behaviour & Organization*, pp. 1-8, Aug. 2012.

[93] J. Kammerl. , N. Blodow, R.B. Rusu, S. Gedikli, and E. Steinbach M Beetz, "Real-time compression of point cloud streams," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on , pp.778,785, 14-18 May*, 2012.

[94] D. Thanou, Dorina, P.A Chou, Philip A., P. Frossard , "Graph-based compression of dynamic 3D point cloud sequences," in *Cornell Library, arxiv, Computer Vision and Pattern Recognition*, 2015.

[95] Y. Huang, J. Peng, C. Kuo, C. Gopi , "A generic scheme for progressive point cloud coding," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 440-453, 2008.

[96] R. Schnabel, and R. Klein , "Octree-based Point-Cloud Compression," in *SPBG*, 2006, pp. 111-120.

[97] Y. Huang and J. Peng Y. Fan, "Point cloud compression based on hierarchical point clustering," , Kaohsiung, 2013, pp. 1-7.

[98] K. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, A. Hornung., "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Springer Autonomous Robots*, pp. 189-206, February 2013.

[99] Z. Adami, "Quaternion Compression," in *Game Programming Gems*, Dante Treglia, Ed.: Charles River Media, 2002, ch. 2.4., pp. 187-191.

[100] S. Aluru B. Hariharan, "Efficient parallel algorithms and software for compressed octrees with applications to hierarchical methods," *Parallel Computing*, vol. 31, no. 4, pp. 311-331, March-April 2005.

[101] A. Doumanoglou, D. Alexiadis, D. Zarpalas, P. Daras , "Towards Real-Time and Efficient Compression of Human Time-Varying-Meshes," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 12, December 2014.

[102] R.B. Rusu, "3D is here: Point Cloud Library," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, Shanghai, 2011.

[103] L. Bivolarski, R. Mekuria, M. Preda. , "N15578 Preliminary guidelines for evaluation of Point Cloud compression technologies," ISO/IEC JCT1 SC29 WG11 (MPEG), geneva, mpeg output document MPEG-4, 2015.

[104] IANA. Session Description Protocol (SDP) Parameters. [Online]. http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xhtml

[105] Microsoft Asia, "NEWT Network Emulator for Windows," Microsoft Software, 2013.

[106] SK Narang, P Frossard, A Ortega, P Vandergheynst D Shuman, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83-98, 2013.

Appendix A 3D Audio Visual Capture Model

Figure A.1 shows a general block diagram of an audio/visual system where 6 interfaces are defined. This appendix is based on an update in from MPEG requirements, the following document: ISO/IEC JTC 1/SC 29/WG 11 N15737 Geneva, CH – October 2015 [48]

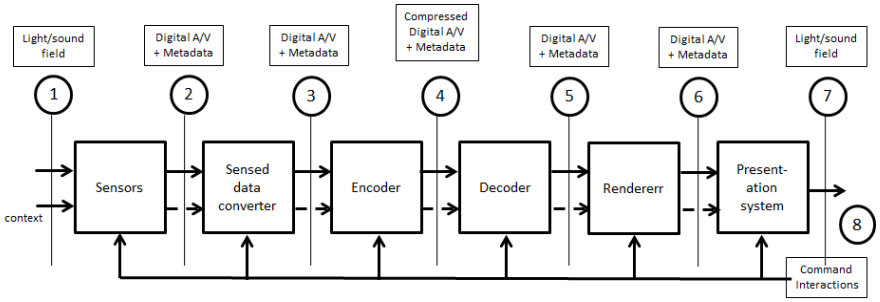


Fig. A.1 – Components and interfaces of a 3D A/V system

Legend

a. **Sensors:** a set of devices capable of sensing the light/sound field and the context in which the field is captured
b. **Sensed data converter:** converts the sensed data to a format that can be used to feed the encoder
c. **Encoder:** compresses audio-visual information
d. **Decoder:** processes the data from Encoder to produce a version of the data input to Encoder
e. **Renderer:** converts the data from Decoder for presentation
f. **Presentation System:** generates audio and visual fields from the data generated by Decoder
g. **Command interactions**: converts human actions or information from environment to input to the 3D AV system

Examples of interfaces (numbers refer to interface formats)

1. Examples
    a. Natural audio/video field with context information
    b. Scene context information, e.g. architectural setup of the scene (cf. reverberation on walls in audio)
2. Output of sensors
3. AV data with a specified format
    a. Fusion of multiple, low-density point clouds taken from different viewpoints of the same scene (mostly static to be able to reposition the same high-cost sensor, e.g. LIDAR) into a highly-dense single point cloud (e.g. KinectFusion, SLAM (Simultaneous Localization and Mapping)) [comment: in a sense, this is multi-sensor pre-processing, while multi-cam light fields typically code the raw data and the "fusion" is done at the rendering stage with Depth Image-Based Rendering, cf. (6)]
    b. Audio
        i. Channels
        ii. Objects
        iii. Ambisonic
    c. Visual
        i. 2D video
        ii. Multi-camera rigs = Light Field camera rigs = Multiple views (2D video + depth)
        iii. Multi-camera in-a-box = Light Field cameras
        iv. Coloured point clouds (LIDAR and Time-of-Flight capture, with accompanying RGB sensor)
        v. Holographic interference fringes (needs trillions/zillions of capturing pixels [even though the pixels can be binary or 2 bits, with 1 million times more pixels at the source, the compression should be really very effective to do better than transmitting a point cloud and calculating the holographic fringes]
4. Compressed AV data
    a. Audio
        i. MPEG-H 3D Audio
    b. Video
        i. MPEG-2 Multiview profile

   ii. MPEG-4 3D-HEVC
   iii. …
5. Same formats as #3, e.g. even in Multiview+depth where depth is not rendered, the depth is used in the renderer (6) to do proper inpainting
6. Output of renderer:
   a. Rectangular 2D images:
      i. Without decoder post-processing, i.e. just pixel-by-pixel frame buffer copy
      ii. With decoder post-processing, e.g. resampling filter to render in larger or smaller window
   b. 2D rectangular projections of 3D data:
      i. Inpainting in Depth Image Based Rendering (occlusion handling)
      ii. Splat rendering (= kind of inpainting) in Point Clouds
      iii. Rasterization of mesh triangles, cf. OpenGL fragment shader
   c. Holographic rendering = illumination with laser light (physical equivalent of raytracing)
7. Natural audio/video field + other information
8. Sensors and actuators
   a. Viewer/listener (yaw, pitch, roll) orientation information and (x,y,z) position information

Appendix B Standardisation Contributions

MPEG INPUT CONTRIBUTIONS:

R. MEKURIA, M. SANNA, S.ASIOLI, E. IZQUIERDO, D. C.A. BULTERMAN P. CESAR A 3D TELE-IMMERSION SYSTEM BASED ON LIVE CAPTURED MESH GEOMETRY M29315 JTC1 SC29 WG11 APRIL 2013, INCHEON KOREA

R. MEKURIA P.S CESAR TECHNICAL REPORT CE4 - CUSTOM BLOCK BASED IMMERSIVE CODEC AND SC3DMC M29973 JTC1 SC29 WG11 JULY 2013, VIENNA, AUSTRIA

R. MEKURIA P.S CESAR EXPLORING LOW BIT-RATE COMPRESSION OF LIVE RECONSTRUCTED DYNAMIC GEOMETRY M32400 JTC1 SC29 WG11 SAN JOSE, JANUARY 2014

R. MEKURIA P.S CESAR M.PREDA INITIAL SET OF REQUIREMENTS FOR 3D TELE-IMMERSION APPLICATION M32641 JTC1 SC29 WG11 SAN JOSE, JANUARY 2014

R MEKURIA ADDING SUPPORT FOR COMMON 3D MESH FILE FORMATS IN THE GRAPHICS CODECS M34442 JTC1 SC29 WG11 SAPPORO, JULY 2014

R. MEKURIA P. CESAR 3DTI MESH CODING ARCHITECTURE AND BASIC IMPLEMENTATION M35196 JTC1 SC29 WG11 STRASBOURG, OCTOBER 2014

R. MEKURIA P. CESAR MESH TO FAMC CONVERTER M35197 JTC1 SC29 WG11 STRASBOURG, OCTOBER 2014

Z. XU, Q. ZHANG, E. IZQUIERDO, R. MEKURIA P. CESAR EXPERIMENTAL DATASETS OF 3D POINT CLOUDS CAPTURED WITH KINECT V2 M35198 JTC1 SC29 WG11 STRASBOURG, OCTOBER 2014

R. MEKURIA POINT CLOUD ENCODER AND DECODER WITH IMPROVED COLOUR CODING M35490 JTC1 SC29 WG11 GENEVA, FEBRUARY 2015

R. MEKURIA .CESAR PROPOSAL FOR QUALITY AND PERFORMANCE METRICS POINT CLOUD COMPRESSION M35491 JTC1 SC29 WG11 GENEVA, FEBRUARY 2015

R.MEKURIA K.BLOM STATUS OF THE POINT CLOUD COMPRESSION PLATFORM REFERENCE SOFTWARE M36527 JTC1 SC29 WG11 WARSAW, JULY 2015

R. MEKURIA ALGORITHM FOR INTER-PREDICTIVE CODING OF UNORGANIZED POINT CLOUD DATA M36528 JTC1 SC29 WG11 WARSAW, JULY 2015

R.MEKURIA, P.CESAR DOCUMENT DESCRIBING THE QUALITY METRIC AND ITS USAGE IN THE POINT CLOUD COMPRESSION REFERENCE SOFTWARE M36529 JTC1 SC29 WG11 WARSAW, JULY 2015

R. MEKURIA, G. LAFRUIT, L. CHIARGLIONE 3D VISUAL CAPTURE INVENTORY M37050 JTC1 SC29 WG11 GENEVA, OCTOBER 2015

R.MEKURIA BLOCK DIAGRAM OF CURRENT PCC REFERENCE SOFTWARE ENCODER/DECODER M37057 JTC1 SC29 WG11 GENEVA, OCTOBER 2015

R.MEKURIA, K. BLOM, P.S. CESAR POINT CLOUD CODEC FOR TELE-IMMERSIVE VIDEO M38136 JTC1 SC29 WG11 SAN DIEGO, FEBRUARY 2016

R. MEKURIA, P. CESAR MP3DG-PCC SOFTWARE PLATFORM FOR POINT CLOUD COMPRESSION M38753 JTC1 SC29 WG11 GENEVA, JUNE 2016

R. MEKURIA, P. CESAR PROPOSAL FOR MPEG-4 PCC FEATURES ROADMAP M38754 JTC1 SC29 WG11 GENEVA, JUNE 2016

B. KATHARIYA, Z. LI, Y.K. WANG, R. MEKURIA PLANE PROJECTION APPROXIMATION FOR VOXEL COLOUR ATTRIBUTES COMPRESSION M38801 JTC1 SC29 WG11 GENEVA, JUNE 2016

K. AINALA, Z. LI, Y.K WANG, R. MEKURIA POINT CLOUD GEOMETRY COMPRESSION WITH PLANE PROJECTION APPROXIMATION AND LEARNING BASED COMPRESSION M38816 JTC1 SC29 WG11 GENEVA, JUNE 2016

MPEG OUTPUT DOCUMENTS:

VAR. EDIT. CORE EXPERIMENTS DESCRIPTION FOR 3DG N13603, JTC1 SC29 WG11 INCHEON KOREA, APRIL 2013

R. MEKURIA, P. CESAR, L. BIVOLARSKI, M. PREDA. 3D TELE-IMMERSION USE CASE AND ASSOCIATED DRAFT REQUIREMENTS N14197 JTC1 SC29 WG11 SAN JOSE, JANUARY 2014

S. W. LEE, I. KANEKO, R. MEKURIA INVESTIGATION ON 3D PRINTING N15303 JTC1 SC29 WG11 GENEVA, FEBRUARY 2015

L. BIVOLARSKI, R.MEKURIA, M. PREDA PRELIMINARY GUIDELINES FOR EVALUATION OF POINT CLOUD COMPRESSION TECHNOLOGIES N15578 JTC1 SC29 WG11 WARSAW, JULY 2015

R.MEKURIA, L. BIVOLARSKI CURRENT STATUS ON POINT CLOUD COMPRESSION N15869 JTC1 SC29 WG11 GENEVA, OCTOBER 2015

R. MEKURIA , P.S CESAR, DESCRIPTION OF PCC SOFTWARE IMPLEMENTATION N16121 JTC1/SC29/WG11 SAN DIEGO, FEBRUARY 2016

R. MEKURIA , P.S CESAR, CURRENT STATUS ON POINT CLOUD COMPRESSION (PCC) N16122 JTC1/SC29/WG11 SAN DIEGO, FEBRUARY 2016

R. MEKURIA, C. TULVAN, Z. LI REQUIREMENTS FOR POINT CLOUD COMPRESSION N16330 JTC1 SC29 WG11 GENEVA, JUNE 2016 **(PUBLIC)**

C. TULVAN, R. MEKURIA, Z. LI, S. LASERRE USE CASES FOR POINT CLOUD COMPRESSION N16331 JTC1 SC29 WG11 GENEVA, JUNE 2016 **(PUBLIC)**

R. MEKURIA, Z. LI, C. TULVAN, P.A CHOU EVALUATION CRITERIA FOR POINT CLOUD COMPRESSION N16332 JTC1 SC29 WG11 GENEVA, JUNE 2016 **(PUBLIC)**

C. TULVAN, R. MEKURIA, Z. LI DRAFT DATASET DESCRIPTION FOR POINT CLOUD COMPRESSION N16333 JTC1 SC29 WG11 GENEVA, JUNE 2016 **(PUBLIC)**

R. MEKURIA, Z. LI, C. TULVAN, DRAFT CFP FOR POINT CLOUD COMPRESSION N16334 JTC1 SC29 WG11 GENEVA, JUNE 2016 **(PUBLIC)**

JPEG OUTPUT CONTRIBUTIONS:

R. MEKURIA POINT CLOUD COMPRESSION JTC1 SC29 WG1 WARSAW, JULY 2015 (PART OF PROCEEDINGS OF JPEG PLENO WORKSHOP ON JPEG PLENO)