Dynamic Adaptive Mesh Streaming for Real-time 3D Teleimmersion

Simon Crowle[1] Alexandros Doumanoglou[2] Benjamin Poussard[3] Michael Boniface[1]
Dimitrios Zarpalas[2] Petros Daras[2]

1: IT Innovation Centre, University of Southampton, SO16 7NS, U.K. Email: {sgc, mjb}@it-innovation.soton.ac.uk
2: Centre for Research and Technology Hellas, Information Technologies Institute,
6^{th} Km Charilaou-Thermi rd, 57001, Thessaloniki, Greece. Email: {aldoum, zarpalas, daras}@iti.gr
3: Arts et Mtiers ParisTech, LAMPA, 2 Bd du Ronceray, Angers, France. Email: {benjamin.poussard}@ensam.eu

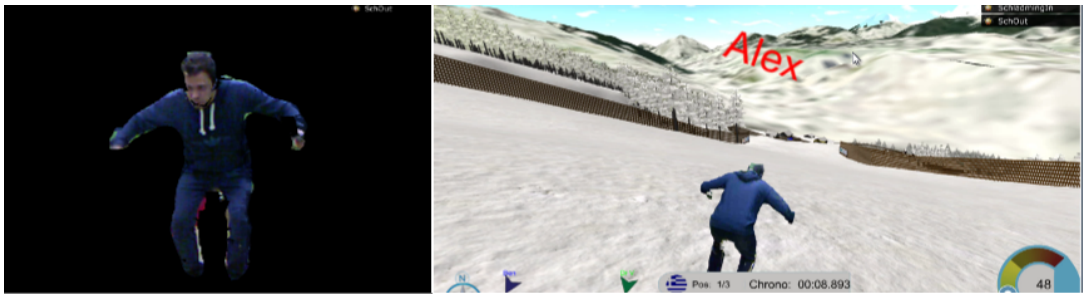

Figure 1: *3D-LIVE Mixed Reality Platform. In-Game Human 3D-Reconstruction.*

Abstract

Recent advances in full body 3D reconstruction methods have lead to the realisation of high quality, real-time, photo realistic capture of users in a range of tele-immersion (TI) contexts including gaming and mixed reality environments. The full body reconstruction (FBR) process is computationally expensive requiring comparatively high CPU, GPU and network resources in order to maintain a shared, virtual reality in which high quality 3D reproductions of users can be rendered in real-time. A significant optimisation of the delivery of FBR content has been achieved through the real-time compression and de-compression of 3D geometry and textures. Here we present a new, adaptive compression methodology that allows a TI system called 3D-LIVE to modify the quality and speed of a FBR TI pipeline based on the data carrying capability of the network. Our rule-based adaptation strategy uses network performance sampling processes and a configurable rule engine to dynamically alter the compression of FBR reconstruction on-the-fly. We demonstrate the efficacy of the approach with an experimental evaluation of system and conclude with a discussion of future directions for adaptive FBR compression.

CR Categories: [Networks]: Network performance evaluation—Network performance modeling, [Networks]: Network performance evaluation—Network simulations, [Networks]: Network performance evaluation—Network experimentation. [Networks]: Network performance evaluation—Network performance analysis. [Theory of Computation]: Design and analysis of algorithms—Data structures design and analysis: Data compression.

Keywords: content adaptation, adaptive compression, network monitoring, QoS.

1 Introduction

Tele-immersion (TI) technology aims to enable users in distributed geographical locations to interact in real-time inside a shared virtual world as if they were physically co-present [Hasenfratz et al. 2004]. To accomplish this, TI technology employs computer vision for image and/or depth acquisition and 3D reconstruction; information theory for data compression and coding; networking techniques for data exchange between remote sites; and computer graphics for rendering and 3D visualization.

In general, many different 3D scene representations for Teleimmersion can be found in the literature that are extensively described in [Smolic 2011]. These representations lie between two extremes: *image-based* and *geometry-based* modelling. Image-based modelling does not use any 3D geometry at all and virtual views are synthesized from natural camera views by interpolation. In geometry-based modelling, the scene is represented on the basis of rendering 3D meshes. This means that in geometry-based modelling robust 3D reconstruction that produces three dimensional meshes is required; virtual views are rendered using the resultant data using classical computer graphics methods. In between the two extremes there are other methods that combine image-based modelling with 3D information like depth maps [Fehn 2004], [Matsuyama et al. 2004].

In a demanding TI scenario, geometry based representations would be preferable as opposed to image based ones [Alexiadis et al. 2014a]. Usually, a TI system would implement functionalities that involve interactions between virtual world elements and the physical world being captured (e.g. collision detection). This kind of functionality is rather difficult to be accomplished by any representation other than geometry-based since camera view dependent image projections cannot meaningfully represent volume in space. Moreover, in a multi-party TI scenario, where multiple users interact in the same virtual world and each user is being captured by multiple cameras, the high computational cost for each client to generate the synthesized view of the others, would degrade the overall system's performance. For these reasons the advantages of geometry-based representation in realizing TI are considered more effective for interactive systems of this kind.

To support this case, a key component of the TI pipeline is real-time 3D reconstruction and the ability to scan a 360^o view of the human body in real-time is essential. To date the related work in the literature in this particular field is quite limited. In [Alexiadis et al. 2013a], a method is presented for full 3D model reconstruction of moving objects from multiple RGB-Depth (Microsoft Kinect) streams in real-time using step discontinuity constrained triangulation (SDCT). Later, in [Alexiadis et al. 2013b] the same authors presented a method that fuses implicitly the information from all sensors to produce watertight textured models using the marching cubes volumetric reconstruction algorithm. In [Alexiadis et al. 2014b] again the same authors propose a technique for producing smooth watertight textured models but this time using a volumetric reconstruction algorithm based on Fourier-Transform.

In order for interactive communications to become a reality, a multi-disciplinary approach that embraces the scientific disciplines described above is needed to achieve fast and efficient solutions for full body reconstruction. In a typical TI pipeline a substantial factor for maximizing the system's performance and interactiveness is the data exchange rates between remote sites. Therefore, efficient data compression algorithms that permit real-time transmission of FBR data streams across networks of varying capability, are of significant importance. While for mesh compression techniques the scientific community has showed significant progress, the availability of literature that specifically address adaptive mesh compression schemes that are sensitive to network conditions in real-time is scare. Moreover, for reasons that will become clear in section 2 and particularly 2.3.5, adaptive mesh compression schemes in the aforementioned sense are subject to additional requirements when applied in TI scenarios.

The objectives of the work reported here are to study and propose an efficient solution to the problem of delivering TI content (i.e. 3D reconstructed meshes) in its best possible quality to multiple recipients, under varying network conditions, with the restriction of minimum delay and maximum update rate, to ensure real-time interactions. To this end a network monitoring method is proposed that provides compression heuristics as input to the TI pipeline process, provided by a run-time rule engine. The rule engine actively determines the compression parameters to use in a variation of the compression scheme in [Alexiadis et al. 2014a] ensuring best visual quality and real-time interactions under the observed network conditions.

The rest of the paper is structured as follows: In section 2 related work to the studied problem is given. Section 3 describes the proposed methodology and the experimentation used to examine the behaviour of our adaptive system in a real-world situation. Finally, section 4 discusses the outcomes of the approach and offers conclusions and ideas for future work.

2 Related work

The related work ahead is split into three categories: a) Tele-Immersion systems b) Mesh compression schemes c) Networking. In the rest of the section, relevant works are presented for each category, followed by an evaluation of the techniques and their application to the streaming of 3D FBR content.

2.1 Tele-Immersion systems

In the literature most of the TI systems indeed utilize geometry-based representation. In the TI system of [Vasudevan et al. 2011] RGB stereo cameras are used to generate depth maps. The depth values from each of the maps are then transmitted to other parties while at the rendering sites, intermediate views for different viewpoints are synthesized by combining depth maps; reconstruction is based on a specific triangulation rule. A conceptually similar system is described in [Maimone and Fuchs 2011]. In that work, depth maps are directly captured using multiple Microsoft Kinects but the compression and transmission scheme is not studied. Another TI system presented in [Mekuria et al. 2013] uses the 3D reconstruction scheme of [Alexiadis et al. 2013a] but the mesh connectivity coding is restricted to a specific triangulation rule. Additionally, in all the aforementioned works the mesh color is coded per vertex. Thus, in order to avoid blurring and aliasing in the rendered images, very dense geometric data are required to be compressed and transmitted to other parties. This degrades the performance of the system. In [Alexiadis et al. 2014a] a complete TI pipeline is presented that includes capturing, 3D reconstruction that produces textured meshes, mesh compression via static mesh coders, texture compression via H264 video coding and transmission. Finally, a TI system utilizing image-based representation with intermediate views being interpolated in the capture site and compressed using standard video coding is presented in [Dai and Yang 2013].

2.2 Mesh Compression Schemes

The result of a real-time 3D reconstruction process usually constitutes a time-varying mesh (TVM) [Doumanoglou et al. 2014]. Time-varying meshes are a series of meshes with varying number of vertices and triangles across frames. Real-time delivery of this data between users connected via a typical real-world, consumer network is highly challenging due to the large payload of TVM data that it is produced at run-time. TVMs can be compressed via static mesh compression, i.e. using coders that do not take advantage of potential redundancy of mesh data over time. Alternatively, some approaches to TVM compression have attempted to detect correlations over time and encode these in a continuous data stream. For the first case, the research community has presented many efficient and mature algorithms with the state of the art being [Mamou et al. 2009]; other mature works include [Rossignac 1999] and [Coors and Rossignac 2004]. On the other hand, temporally based TVM compression schemes are not yet mature enough to support real-time TI systems. The most mature method we know and the only that can compress both geometry and connectivity for use in a real-time TI environment is [Doumanoglou et al. 2014].

2.3 Networking

2.3.1 Network protocols

For networking in general, two network protocols are very commonly used to realise the foundations of computer communications: the Transmission Control Protocol (TCP)[Information Sciences Institute 1981] and the User Datagram Protocol (UDP) [Postel 1980]. TCP is optimized for accurate delivery rather than timely delivery,

and therefore, TCP sometimes incurs relatively long delays (on the order of seconds) while waiting for out-of-order messages or retransmissions of lost messages. Thus, time-sensitive and real-time applications often use the User Datagram Protocol (UDP) because dropping packets is preferable to waiting for delayed packets. However, this protocol may also be considered too primitive because guaranteed-order packet delivery is sometimes necessary. A variant of UDP, not yet standardized, is called Reliable UDP which offers enhancements to the original protocol; it extends UDP by adding acknowledgement of received packets, retransmission of lost packets and a few more features. Reliable UDP could be selected as an alternative to TCP, which although also more robust in the delivery of data, adds too much complexity/overhead for demanding, real-time applications.

2.3.2 Streaming

"Streaming" means sending audio, video or other kind of data in a way that the data can be processed before completely received. Streaming is not to be confused with progressive downloading. Progressive downloading is about receiving an ordinary file and starting to process it before it's completely downloaded. It does not require any special protocols, but it requires an appropriate file format that can be processed based on partial content. On the other hand, "streaming" uses a network streaming protocol to control the transfer. In this approach, during "Streaming" the content quality automatically changes in response to the transfer conditions. This is called "adaptive streaming": the content sender typically lowers (in a progressive fashion) the quality of the delivered content in order to ensure a delivery rate that is appropriate for the capabilities receiver. Streaming can be broadly divided into two categories: on-demand and real-time. With on-demand streaming, the client requests the content to receive, e.g. a recording or movie, among various different contents. In contrast, real-time streaming allows the sender to determine the sending content and the receiver plays it back as it is sent, with a slight and consistent delay. In both cases, the receiver plays the content back as soon as it is received. However, to keep up with transmission time and decoding processing time, a buffering mechanism of several seconds is employed to prevent skipping frames while playing back. Thus, playback is not synchronized with the source [McGath 2013].

The Real Time Transport Protocol (RTP) [Schulzrinne et al. 2003] is a transport protocol which is built on UDP and designed specifically for real-time transfers. It's closely associated with the Real Time Control Protocol (RTCP) [Schulzrinne et al. 2003], which operates at the session layer. The primary function of RTCP is to provide feedback on the quality of the data distribution, allowing actions such as adjusting the data rate. The Real Time Streaming Protocol (RTSP) [Schulzrinne et al. 1998] is a presentation-layer protocol that is described as a "network remote control", enabling the client to issue play, pause and forward/backward operations in the server's stream. The protocol stack of RTP, RTCP, and RTSP is sometimes referred to as "RTSP" and often used in streaming.

2.3.3 Network protocols and techniques used in on-line games

In on-line games it is common that TCP, UDP or both are utilized. In order to support real-time interaction between players, special network latency-hiding techniques are applied. For example, in a Real Time Strategy (RTS) game, user interactions with objects in a virtual world will first result in auditory feedback to the user to confirm the action has begun. The game engine will then provide visual feedback in the form of a 'preliminary' animation and only after that the object will perform the requested action. However, the network message for the intent of the action (as part of a logical game state change) is sent immediately; so by the time the screen responds to the player's input, the network messages have already been sent and acknowledged. Moreover, introducing an additional small delay (eg, 100ms) between the mouse click and the game object's response allows for timely network transmission while game responsiveness is not affected. Another trick for handling a possible delay between a command and its execution is something called a local perception filter. If the game client receives a move order some time after the actual order was given, the client actually knows when the unit should have started moving and also knows its end destination. Instead of tele-porting the game object to its correct location, the client may start its movement late and then utilize physics to speed it up slightly so that it can catch up to where it's supposed to be, and then slow it back down to put it in the correct place. Another technique commonly used is dead-reckoning [Murphy 2011]. Dead-reckoning is the name of a technique that uses details regarding an object's past location, direction and speed in order to predict its current location. Applying dead-reckoning to hide the lag time inside the game is called client-side prediction.

2.3.4 Network Protocols in Tele-immersion

In the literature, the only work we found to be related to TI network protocols was [Leigh et al. 2001]. In the respective paper, the use of Forward Error Correction (FEC) codes over UDP for real-time tele-immersion is suggested. This enables the receiver to recover when the number of errors in a packet's transmission is below some threshold without the need of retransmission and at the cost of sending redundant data within each network packet. An alternative suggestion includes using multiple parallel TCP sessions in order to transmit data possibly improving throughput and reducing the overall retransmission delay of the message.

2.3.5 Discussion

For a tele-immersion scenario, streaming in the sense described in section 2.3.2, is not applicable. The RTSP protocol targets scenarios that are different. In tele-immersion there is no sense of controlling the stream via play/pause/backward/forward commands. Instead, fast and reliable delivery of the live stream, without buffering, is what matters. Tele-immersion is most like real-time streaming with the additional constraint of playback being as "simultaneous with the source" as possible. Moreover, the way lag is hidden in online-games has little to do with how lag of transmitting reconstruction data may be handled in TI. Using FEC codes over UDP, as described in section 2.3.4, may be beneficial. However it can be argued that utilizing multiple TCP sessions in parallel in order to reduce transmission delay and improve throughput is doubtful. [Natarajan et al. 2009].

3 Methodology

Our approach to the design and development of a responsive mechanism to deliver adaptive, real-time 3D reconstructed mesh data was grounded in a real world interactive system created as part of an EU funded project, 3D-LIVE. [3D-LIVE]. The 3D-LIVE project provides three game based scenarios (skiing, golfing and jogging) in which remotely connected users simultaneously take part in a shared sporting activity. Whilst it is possible to configure the 3D-LIVE systems for a variety of game players using a wide range of interaction modalities (ranging from outdoor augmented reality and immersive CAVE environments), the current architecture supports the capture and delivery of full body reconstruction for 'indoor' users only. In this case, we imagine two users in geographically separate locations engaging in game play using the 3D-LIVE system; one of the users plays within a full body reconstruction game

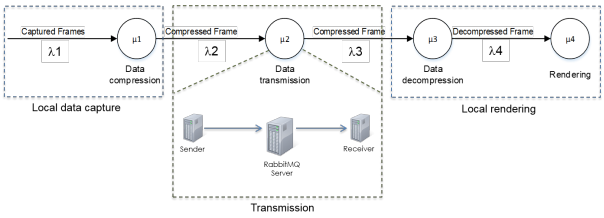

Figure 2: *Abstracted reconstruction pipeline*

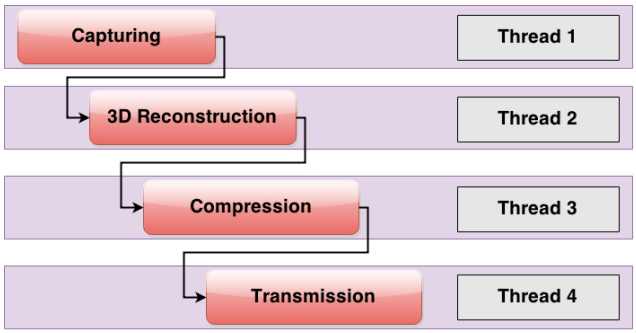

Figure 3: *Tele-Immersion Pipeline at the sender's side*

area - the other engages in a simpler gaming room (realistic for an 'at home' user) in which she can see the other player fully reconstructed in 3D within the virtual environment. Towards developing and evaluating our adaptive algorithm, we adopted a synthesis of model-based and empirical methods. These were comprised of three main phases: i) abstraction of the 3D content pipeline and its constituent processes, ii) design and implementation of an adaptive architecture to support dynamic behaviour in the pipeline; and iii) an empirical, experimental evaluation of the adaptive process using a sub-set of the 3D-LIVE platform updated to include real-time adaptive mesh compression whilst operating in real-world network conditions.

3.1 Phase 1: Pipeline performance modelling

In the first phase, our objectives were to define the macro-level stages of the TI pipeline and then identify those elements in the process that could impact the delivery of FBR frames to the end user. We began by developing an abstracted view of the reconstruction pipeline in its simplest form: one sender of FBR frame data to one receiver. This process can be decomposed into two 'local' processing phases and an intermediate network transmission phase, see figure 2. In the first local phase, vector $\lambda 1$ represents the FBR frame inter-arrival time (an aggregate of the time required for multiple Kinect image data sets to be captured and then fused into a single 3D model). The FBR frame is then compressed adaptively (this is described in more detail in section 3.2.2) at node $\mu 1$ before entering the network transmission phase.

Transmission across the network is approximated by two network data carrying steps $\lambda 2$ and $\lambda 3$ and a data distribution processing operation carried out on a message brokering system (RabbitMQ [RabbitMQ] in this case) during which FBR frame data is routed to the appropriate target. Finally, phase 3 represents the reception, decompression and rendering of the FBR frame by the machine that renders the 3D mesh to the end user (nodes $\mu 3$ and $\mu 4$).

The implementation of the first, local phase of our TI pipeline, starting at the sender's side is depicted in Fig 3. It is a multi-threaded phase consisting of 4 different modules, each one running in a separate thread. The pipeline is very much alike to the pipeline described in [Alexiadis et al. 2014a]. Capturing is performed via 4 Microsoft Kinects while 3D Reconstruction is utilizing the algorithm from [Alexiadis et al. 2014a]. Moreover, the compression module uses OpenCTM [OpenCTM] for mesh compression (like [Alexiadis et al. 2014a]) while for textures it makes use of standard JPEG compression [Wallace 1991], instead of the H264 codec [ITU 2003] that was used in [Alexiadis et al. 2014a].

As already mentioned, the network transmission in this process is supported using the TCP-based message broker system, RabbitMQ. This TCP-based approach was selected in favour of a UDP implemented for the reason that it maintains frame data integrity. Compressed FBR frame data would span across multiple UDP packets due to its increased size making it more likely to be subject to partial data loss. The mesh decompression algorithm employed would not be able to cope with packet loss, unless FEC codes were used. Although FEC codes would improve the performance of the system, even then, the transmission reliability is not certain. To cope with the real-time nature of the system and interactive frame-rates, RabbitMQ was configured to use a queue size of 1 message. Thus, whenever a new 3D FBR frame is available, any yet undelivered 3D frame is dropped (and thus never delivered to the client). This eliminates buffering and minimizes the effect of the receiver lagging behind the transmitter.

An examination of the performance characteristics of in-memory processes of the pipeline was addressed to determine the localised factors influencing transmission performance; these could be easily controlled and observed on a single machine. Data processing throughput of nodes $\lambda 1$, $\mu 1$, $\mu 3$, $\lambda 4$ and $\mu 4$ were modelled using normal distributions calculated from a series of instrumented observations of FBR data processing run on 'sender' and 'receiver' machines respectively. In order to constrain the problem space for these benchmarking exercises, we defined three levels of FBR reconstruction compression quality (with an inversely related level of data compression) that would be applied at the compression step (see table below and Fig. 4). To evaluate the visual quality of the compressed reconstruction, the Peak Signal to Noise Ratio (PSNR) metric is used. The 3D reconstructed mesh is rendered from a viewpoint defined by the Kinect camera's extrinsic parameters and projected to the image plane using the Kinect camera's intrinsic parameters. All the four views, one for each kinect, are used for PSNR calculation. Let $I_v^t(i,j,c)$ denote the value of pixel (i,j) in channel c (standard 3-channel RGB is assumed, $c \in \{0,1,2\}$) for the v-th kinect and frame t. Similarly, $K_v^t(i,j,c)$ represents the image produced by the (compressed) rendered 3D-Reconstruction. The Mean Squared Error (MSE) is then computed as follows:

$$\text{MSE} = \frac{1}{3TVMN} \sum_{t=0}^{T-1} \sum_{v=0}^{V-1} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \sum_{c=0}^{2} (I_v^t(i,j,c) - K_v^t(i,j,c))^2, \tag{1}$$

where T is the total number of frames, V the number of Kinects and M, N the vertical and horizontal resolution of the kinect images, respectively. The PSNR is then computed by:

$$\text{PSNR} = 10 \log_{10} \left(\frac{1}{MSE} \right), \tag{2}$$

Figure 4: *From left to right: Uncompressed, low compression, medium compression, high compression*

where the pixel values for each channel is assumed to lie in the interval [0,1].

The results of the PSNR calculation are given in Table 1. The PSNR of the uncompressed reconstruction was measured as well and found equal to 23.1 dB. Thus, the chosen different compression levels correspond to sufficiently varying visual quality levels.

Quality level	Average bytes/frame	Std.dev bytes/frame	PSNR (dB)
High	244391.52	17013.72	22.2
Medium	192811.64	15219.44	21.6
Low	163358.56	17013.72	20.8

Table 1: *Average file size per frame and PSNR for different visual quality levels*

Frame inter-arrival rate ($\lambda 1$), representing the arrival of a 'raw' FBR frame (and therefore not compressed at this stage) is sensitive to the physical person or object that is being virtual reconstructed at run-time. In this case, we took a representative sample from a previously conducted 3D-LIVE experiments (see [Crowle et al. 2014] for more information on 3D-LIVE trials) and used this as a standard: this value was benchmarked at an average rate of 155.3ms per frame (or about 6.4 frames/second).

Message brokering process time was modelled in a similar fashion by passing pre-recorded FBR frames through the RabbitMQ server operating on a local machine and calculating the normal delay distribution. A summary of the in-process benchmarks for these components of the pipeline are summarized in the table below, showing performance measures for corresponding FBR frame quality.

Pipeline node	High quality	Medium quality	Low quality
$\mu 1$ average (ms)	91.77	91.62	91.48
$\mu 2$ average (ms)	1.6198	1.4994	1.4294
$\mu 3$ average (ms)	30.77	28.56	28.8
$\mu 4$ average (ms)	33.3	33.3	33.3

Our results showed that the in-memory performance characteristics for FBR frame data carried at our pre-defined quality levels remain stable and are processed uniformly - meaning they can be treated as effective constants within the pipeline model. By a process of elimination, we had now only to consider the effect that network performance might have on the transmission phase of the pipeline. In real-world network contexts such as those used by the 3D-LIVE platform, users will have little or no control over these factors and so instead the system itself must respond to prevailing conditions (such as fluctuations in bandwidth, latency, packet loss and other factors). With these factors in mind, and our mitigating control being that ability to change the quality (or level of data compression) of the FBR frame data, we proceeded to designing an adaptation framework.

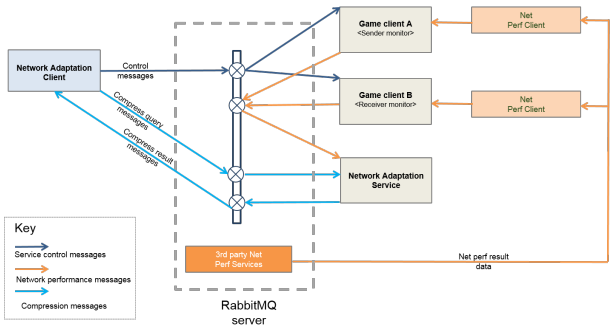

Figure 5: *NAS architecture overview*

3.2 Phase 2: Adaptation design and implementation

Selecting the 3D-LIVE platform provided concrete architectural requirements against which to define an adaptive compression algorithm and system design as well as a known set of performance characteristics for full body reconstruction that we previously captured (in a non-adaptive system) in earlier experiments [Crowle et al. 2014]. Based on this, the primary requirements that shaped the conceptualisation of the adaptive algorithm and system design were as follows:

- Adaptation must be configurable for use in different network conditions.
- Optimisation must be parity driven, delivering FBR data at a level of quality that is consistent for all users.
- Implementation of the adaptive behaviour must have a low impact on the 3D-LIVE system architecture.

Following on from these guiding principles, the Network Adaptation Service (or NAS) was designed and implemented to run in parallel with existing 3D-LIVE system architecture (for more information on the 3D-LIVE architecture, see [Poussard et al. 2014]). Here we restrict our description of the design of the NAS to its high-level architecture and the algorithmic process used to enact adaptation of time-varying mesh delivery at run-time.

3.2.1 NAS architecture

The NAS system is formed of distributed components that are connected via the same message brokering system already used in 3D-LIVE. This approach was chosen as it was well aligned with the communication software modules already in place and also provides us a means to approximate prevailing network conditions that reflect the direction of travel taken by FBR data frames during game play.

NAS architecture comprises of a central, on-line service that is responsible for estimating the overall network conditions (as defined in by the pipeline model) between 3D-LIVE game players and recommending adaptive actions to be taken by the full body reconstruction process. The indicative roles of components in our architectural design of the NAS (Figure 5) are summarised below.

Network Adaptation Client
This is a software client that controls the NAS through a series of commands sent via the RabbitMQ server. These commands control the network performance sampling behaviour of the service and also allows the client to query the Network Adaptation Service for a recommended compression level for full body reconstruction data.

Network Adaptation Service
Central to the system, the Network Adaptation Service orchestrates the sampling of network performance through communication with Monitors that run on game clients and are capable of capturing network statistics that are representative of network performance between the game client and the RabbitMQ server.

Game clients (Network Adaptation Monitors)
Each game client represents a machine that runs a) a 3D-LIVE indoor application and b) a Network Adaptation Monitor that samples network performance between itself and the RabbitMQ server. Sampling is carried out using third party network performance clients that connect to a network performance server (also installed on the RabbitMQ server machine). Network performance statistics are returned to the Network Adaptation Service via RabbitMQ. Monitors are defined as either senders (representing game clients sending FBR data) or receivers (game clients receiving FBR data).

RabbitMQ server
This server is used by the 3D-LIVE system to carry game content messages between all users during game play. An approximation of FBR data transmission performance between any two users in the game can be achieved by aggregating network performance statistics between each player and this server. Third party network performance services are installed on the RabbitMQ server machine and used by the game clients to capture network performance between it and them.

Net Perf Client
Network Adaptation Monitors use the corresponding third party network performance clients to gather statistics about their connection to the RabbitMQ server. In this case we used standard tools to approximate bandwidth and latency (*iPerf*[iPerf] and *ping*[Microsoft Corporation 2015b] respectively).

At run-time, the NAS periodically aggregates network performance statistics from the attached monitors and maintains a live approximation of the all paths leading from full body reconstruction senders to receivers (of which there may be multiples in both cases).

3.2.2 Configurable rule model

Based on our earlier pipeline investigation, our working hypothesis was that the NAS should be sensitive to network performance metrics and provide appropriate changes to mesh compression levels. These changes resolve to greater or lesser levels of compression being applied to FBR frames at run-time with the idea that, if the network allows, higher quality 3D reconstruction can be presented to the end user. Conversely, lower quality 3D frames may be sent to the user but at an increased update rate. Upper bounds on the movement along this quality/speed scale are determined by the constant factors in the pipeline process ($\lambda 1$, $\mu 1$, $\mu 2$ and $\mu 3$). Lower bounds are defined by compression level and available network conditions as defined by $\lambda 2$ and $\lambda 3$.

Therefore we developed a flexible and extensible rule engine that could be used to encapsulate a variety of network performance metrics as input parameters and select between them using run-time configurable rules to determine a compression level. The NAS associates the aggregated network statistics it collects with a specific rule set managed by the rule engine. For example, measurements of network throughput are mapped to a throughput rule set. Each rule classifies the statistics it finds into levels (*high*, *medium* and *low* are used). Once data for all rule sets are available, rules modify control flow through a reasoning process based on their current classifications of network performance to reach a terminating result that specifies the upper and lower bounds for mesh compression (see figure 6). In cases where the network is unable to sustain any

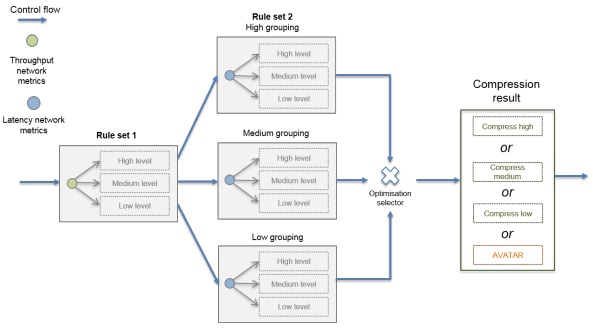

Figure 6: *Example of the NAS configurable rule process (two rule sets)*

kind of acceptable full body reconstruction data (at any quality or speed), the FBR data stream should be replaced by skeleton motion data (used to animate a conventional avatar instead).

3.3 Phase 3: Real-world experimentation

The NAS was developed to integrate with the 3D-LIVE game platform which is intended to support game players on multiple sites in geographically disparate locations. We took steps to replicate this arrangement whilst at the same time applying as many reasonable controls over extraneous influences on system behaviour as possible. To realise a real-world network deployment, we set up a 3D-LIVE full body reconstruction environment in Thessaloniki, Greece; a reconstruction receiver game client in Laval, France; the NAS adaptation service in Southampton, United Kingdom and the RabbitMQ server in Germany. To further mitigate against unexpected and difficult to repeat real-world conditions, we created a static, physical model that would be continuously reconstructed at run-time. This constraint provides us with the ability to hold $\lambda 1$ constant at run-time. The experiment procedure and metrics data capture was managed using the on-line experimentation system 'EXPERImonitor' from the EXPERIMEDIA project [Boniface et al. 2013] which controls experiment flow; aggregates metric meta-data and observations; and provides live visualisation and data export for subsequent analysis.

3.3.1 NAS benchmarking and configuration

Experimentation began with a benchmarking process that allowed us to configure the NAS to work within the real-world conditions available to us. The NAS was set up with two network metric monitoring instruments per user: a throughput sampler (based on *iPerf*) and a latency sampler (based on *ping*). Using NAS sampling, we were able to take benchmark values for the best currently available network conditions between the two 3D-LIVE machines, there were:

3D-LIVE machine	Latency to Rabbit	Throughput from/to Rabbit
Sender	48.45 ms (average)	12.48Mbps (average)
Receiver	50.8 ms (average)	8.398 Mbps (average)

In order to establish some influence over network conditions between the two users, we used Microsoft's *Network Emulator for Windows Toolkit* (NEWT) [Microsoft Corporation 2015a] to locally constrain network traffic on a single machine (on the sender side) whilst mainlining optimum network conditions on the receiver end. A series of preliminary, exploratory tests using NEWT to change a number of network behaviours whilst streaming live reconstruction data lead us to select *available bandwidth* as the principal factor to

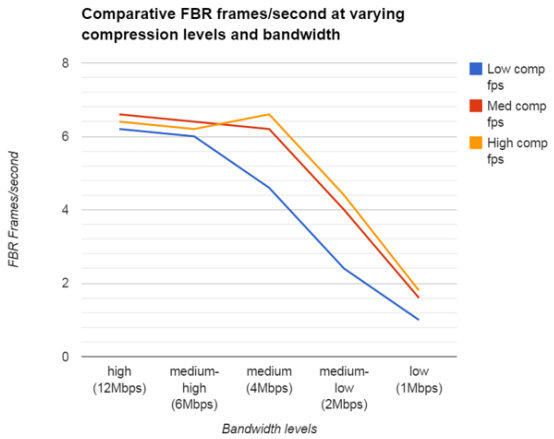

Figure 7: *Real-world benchmarking results for NAS configuration*

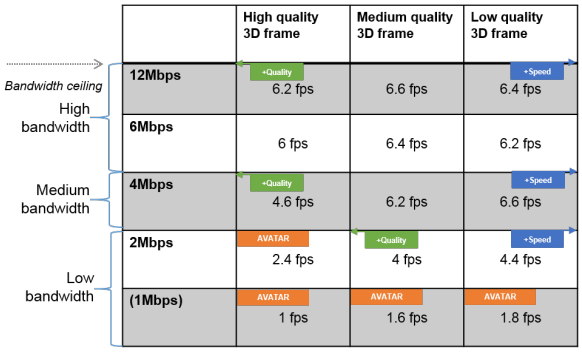

Figure 8: *NAS rule configuration*

investigate as an impact on the transmission of FBR data between users.

Our exploration of this influence was conducted at five bandwidth levels whilst at the same time changing FBR frame data compression. The results provided us with enough data to create a rule set that would allow us define a trial NAS rule set (based on bandwidth metrics) and a nominal threshold below which full body reconstruction would be considered non-viable (this was set a 3fps).

Figure 8 visualises the rule set that developed for subsequent run-time evaluation trials. The three classification boundaries were mapped to the benchmarking results and the resultant ranges where quality can be traded off for speed indicated. In this simple case, we can see that in high-bandwidth scenarios, high quality FBR frames can be sent at close to maximum performance. As available bandwidth drops, highest quality FBR frames can be maintained, but at the cost of a reduced frame-rate. Finally, in low bandwidth conditions, quality is initially limited and is then dropped altogether once frame rate falls below 3fps.

3.3.2 Run-time evaluation

Once calibrated, the NAS was integrated with the 3D-LIVE gaming platform to verify system adaptation responses to available network resources. Using the three bandwidth levels as use-cases, we ran the NAS and reconstruction streaming processes and periodically issued commands to either increase the speed or the quality of the full body reconstruction. During this time we observed the rate at

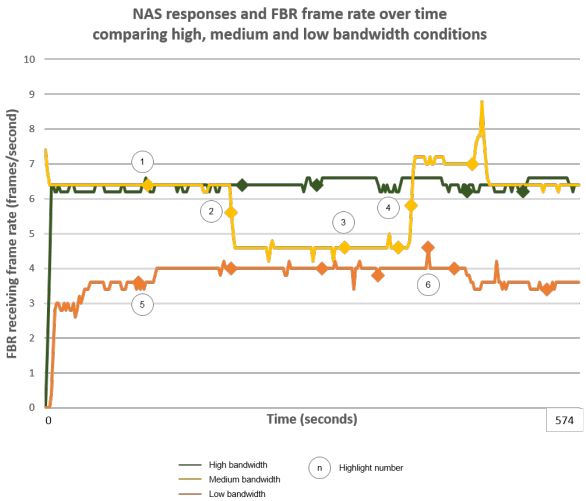

Figure 9: *Comparative adaptation of FBR compression in response to network conditions*

which FBR frames arrived at the receiver side and also noted the NAS responses to demands for change in quality of speed.

Figure 9 visualises FBR receiving frame rate changes over time in each of the three experimental bandwidth levels tested; we indicate the points in each series where changes in quality of speed were requested with diamond markers and selectively highlight points where these changes impact frame rate. Under the high bandwidth scenario, the full range of quality is available at the maximum rate at which FBR frames can be reconstructed by the sending machine: in this case there was no advantage to reducing the quality of frames sent.

Under medium bandwidth conditions we see a much clearer case that demonstrates where dynamically sacrificing quality at run-time can produce improvements in FBR frame throughput. In the table below we enumerate the six QoS medium bandwidth markers, comparing QoS change requests with NAS compression recommendations.

Marker	QoS request	NAS compression response
1	Quality+	Medium
2	Quality+	Low
3	Quality+	Low
4	Speed+	Medium
5	Speed+	High
6	Speed+	High

Marker and highlight 1 show the first change in QoS changing from an initial *low quality* setting to a medium quality - the NAS responds correctly with a reduction from high to medium compression. At a medium bandwidth level this change of quality can be sustained at the maximum reconstruction frame rate. Highlight 2 (and marker 2) shows the point where we make a further QoS request for quality and in this case compression goes down, but so too does FBR receiving frame rate. The next QoS change request (marker/highlight 3) shows that the system has reached the 'floor' of its QoS range and we see no change in performance or quality. Finally for this series, marker/highlight 4 begins a QoS change in the opposite direction towards improvements in speed which eventually return to the 'ceiling' frame rate.

A similar pattern for the low bandwidth scenario was also observed,

but to a smaller degree. Highlight 5 indicates the point where we request an increase of speed from an initial starting point of medium quality (NAS rules prevent high quality FBR reconstruction in this case). The low bandwidth series continues with attempts to increase speed: here the NAS starts to recommend avatar skeleton data as an alternative (FBR streaming is continued in any case). At the last highlight in figure 9 we show the first request for an improvement in quality - here there was an short delay before changes in frame rate by the receiver were observable.

4 Conclusions & Future work

Our results confirmed that the NAS responded with the appropriate mesh compression recommendations which were affected by the 3D-LIVE FBR service in real-time in three varying levels of network capability. The full body reconstruction frames per second performance at highest quality in these controlled tests are representative of the current maximum real-time performance for an interactive TI context. These compare with earlier 3D-LIVE user trials [Crowle et al. 2014] where equivalent observations of receiving frame rate were slightly lower than this performance ceiling - possibly due to the additional overhead of other game processes (such as voice communication). We also note that our bandwidth sampling processes had a significant impact on streaming when executed concurrently, meaning that we had to sample this network characteristic in-between simulated game sessions. Significant impacts to game performance caused by heavy-weight network performance sampling would not be acceptable to users and so would either have to be carried out at strategic moments during game-play (such as during game set-up scenes) or replaced with light-weight 'passive' methods.

This exploratory work offers a novel approach, design and evaluation methodology to realise an adaptive compression framework for real-time full body reconstruction in a tele-immersion environment. Based on this foundation, we anticipate further refinements and improvements to the approach. These include the ability of the NAS to automatically generate its own rule set based on an automated benchmarking process of the network and the TI pipeline itself; the development of low-impact, passive network capability sampling methods integrated into the NAS protocol; and enhancements to data compression techniques including support for layered bitstreams similar to the methods used in contemporary video streaming technologies, by utilizing progressive compression of geometry and textures.

5 Acknowledgements

This work was supported by the EU funded project 3DLIVE, GA 318483. http://3dliveproject.eu/.

References

3D-LIVE. 3D-LIVE: 3D Living Interactions through Visual Environments. [Online] http://3dliveproject.eu/wp/.

ALEXIADIS, D. S., ZARPALAS, D., AND DARAS, P. 2013. Real-time, full 3-d reconstruction of moving foreground objects from multiple consumer depth cameras. *IEEE Transactions on Multimedia 15*, 2, 339–358.

ALEXIADIS, D. S., ZARPALAS, D., AND DARAS, P. 2013. Real-time, realistic full-body 3d reconstruction and texture mapping from multiple kinects. In *11th IEEE IVMSP Workshop: 3D Image/Video Technologies and Applications, Yonsei University, Seoul, Korea, 10-12 June*.

ALEXIADIS, D. S., DOUMANOGLOU, A., ZARPALAS, D., AND DARAS, P. 2014. A case study for tele-immersion communication applications: From 3d capturing to rendering. In *2014 IEEE Visual Communications and Image Processing Conference, VCIP 2014, Valletta, Malta, December 7-10, 2014*, 278–281.

ALEXIADIS, D. S., ZARPALAS, D., AND DARAS, P. 2014. Fast and smooth 3d reconstruction using multiple rgb-depth sensors. In *2014 IEEE Visual Communications and Image Processing Conference, VCIP 2014, Valletta, Malta, December 7-10, 2014*, 173–176.

BONIFACE, M., PHILLIPS, S., VOULODIMOS, A., SALAMA, D., AND MURG, S. 2013. Experimedia: Technology enablers for a future media internet testing facility. In *Proceedings of the NEM Summit 2013*, Eurescom GmbH, Heidelberg, Germany, 28–30.

COORS, V., AND ROSSIGNAC, J. 2004. Delphi: geometry-based connectivity prediction in triangle mesh compression. *The Visual Computer 20*, 8-9, 507–520.

CROWLE, S., BONIFACE, M., POUSSARD, B., AND ASTERIADIS, S. 2014. A design and evaluation framework for a tele-immersive mixed reality platform. In *Augmented and Virtual Reality*, L. T. De Paolis and A. Mongelli, Eds., vol. 8853 of *Lecture Notes in Computer Science*. Springer International Publishing, 151–158.

DAI, B., AND YANG, X. 2013. A low-latency 3d teleconferencing system with image based approach. In *Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, ACM, New York, NY, USA, VRCAI '13, 243–248.

DOUMANOGLOU, A., ALEXIADIS, D. S., ZARPALAS, D., AND DARAS, P. 2014. Toward real-time and efficient compression of human time-varying meshes. *IEEE Trans. Circuits Syst. Video Techn. 24*, 12, 2099–2116.

FEHN, C. 2004. 3D-TV using depth-image-based rendering. Proceedings of the of Picture Coding Symposium (PCS), San Francisco, CA, USA.

HASENFRATZ, J.-M., LAPIERRE, M., AND SILLION, F. 2004. A real-time system for full body interaction with virtual worlds. In *Proceedings of the Tenth Eurographics Conference on Virtual Environments*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, EGVE'04, 147–156.

INFORMATION SCIENCES INSTITUTE, U. O. S. C., 1981. Rfc 793: Transmission control protocol.

IPERF. [Online] https://iperf.fr/.

ITU, I. T. S. S. O., 2003. H.264: Advanced Video Coding for Generic Audiovisual Services. [Online] http://www.itu.int/rec/T-REC-H.264-200305-S/en.

LEIGH, J., YU, O., SCHONFELD, D., ANSARI, R., HE, E., NAYAK, A., GE, J., KRISHNAPRASAD, N., PARK, K., CHO, Y.-J., HU, L., FANG, R., VERLO, A., WINKLER, L., AND DE FANTI, T. 2001. Adaptive networking for tele-immersion. In *Immersive Projection Technology and Virtual Environments 2001*, B. Frhlich, J. Deisinger, and H.-J. Bullinger, Eds., Eurographics. Springer Vienna, 199–208.

LIEN, J.-M., KURILLO, G., AND BAJCSY, R. 2009. Multi-camera tele-immersion system with real-time model driven data compression: A new model-based compression method for massive dynamic point data. *Vis. Comput. 26*, 1 (Nov.), 3–15.

MAIMONE, A., AND FUCHS, H. 2011. Encumbrance-free telepresence system with real-time 3d capture and display using commodity depth cameras. In *10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011, Basel, Switzerland, October 26-29, 2011*, 137–146.

MAMOU, K., ZAHARIA, T. B., AND PRÊTEUX, F. J. 2009. TFAN: A low complexity 3d mesh compression algorithm. *Journal of Visualization and Computer Animation 20*, 2-3, 343–354.

MATSUYAMA, T., WU, X., TAKAI, T., AND NOBUHARA, S. 2004. Real-time 3d shape reconstruction, dynamic 3d mesh deformation, and high fidelity visualization for 3d video. *Comput. Vis. Image Underst. 96*, 3 (Dec.), 393–434.

MCGATH, G., 2013. Basics of streaming protocols. [Online] http://www.garymcgath.com/streamingprotocols.html.

MEKURIA, R., SANNA, M., ASIOLI, S., IZQUIERDO, E., BULTERMAN, D. C. A., AND CÉSAR, P. 2013. A 3d tele-immersion system based on live captured mesh geometry. In *Multimedia Systems Conference 2013, MMSys '13, Oslo, Norway, February 27 - March 01, 2013*, 24–35.

MICROSOFT CORPORATION, 2015. Network emulator for windows toolkit. [Online] https://blog.mrpol.nl/2010/01/14/network-emulator-toolkit/.

MICROSOFT CORPORATION, 2015. Ping. [Online] https://technet.microsoft.com/en-us/library/cc940091.aspx.

MURPHY, C. 2011. *Game Engine Gems 2, Believable Dead Reckoning for Networked Games*. ch. 18, 307328.

NATARAJAN, P., BAKER, F., AND AMER, P. D., 2009. Multiple TCP Connections Improve HTTP Throughput-Myth or Fact?

OPENCTM. [Online] http://openctm.sourceforge.net/.

POSTEL, J., 1980. RFC 768: User Datagram Protocol.

POUSSARD, B., RICHIR, S., VATJUS-ANTTILA, J., ASTERIADIS, S., ZARPALAS, D., AND DARAS, P. 2014. 3dlive: A multimodal sensing platform allowing tele-immersive sports applications. In *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*, 356–360.

RABBITMQ. http://www.rabbitmq.com/.

ROSSIGNAC, J. 1999. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics 5*, 1 (Jan.), 47–61.

SCHULZRINNE, H., RAO, A., AND LANPHIER, R., 1998. Real Time Streaming Protocol (RTSP).

SCHULZRINNE, H., CASNER, S., FREDERICK, R., AND JACOBSON, V., 2003. RTP: A Transport Protocol for Real-Time Applications.

SMOLIC, A. 2011. 3d video and free viewpoint video-from capture to display. *Pattern Recogn. 44*, 9 (Sept.), 1958–1968.

VASUDEVAN, R., KURILLO, G., LOBATON, E. J., BERNARDIN, T., KREYLOS, O., BAJCSY, R., AND NAHRSTEDT, K. 2011. High-quality visualization for geographically distributed 3-d teleimmersive applications. *IEEE Transactions on Multimedia 13*, 3, 573–584.

WALLACE, G. K. 1991. The jpeg still picture compression standard. *Commun. ACM 34*, 4 (Apr.), 30–44.