D3.3 3D-LIVE platform modules

Deliverable data

Deliverable no & name	***D3.3 3D-LIVE platform modules***
Main Contributors	*CERTH*
Other Contributors	*ARTS, CYBER, IT-IN*
Deliverable Nature	*P*
Dissemination level	**PU** – Public – X **~~PP~~** – Restricted to other programme participants (including the Commission Services) **~~RE~~** – Restricted to a group specified by the consortium (including the Commission Services) **~~CO~~** – Confidential, only for members of the consortium (including the Commission Services)
Date	*30/3/2015*
Status	*Final*

23/06/2015

Document history

Version	Date	Author /Reviewer	Description
V0.1	21/11/2014	S. Asteriadis D. Zarpalas	Initial Structure and Input
V0.2	01/12/2014	A.Doumanoglou	Further contributions
V0.3	12/12/2014	D.Zarpalas, S.Asteriadis, A. Doumanoglou	Further contributions
V0.4	15/12/2014	A. Doumanoglou	Further contributions.
V0.5	19/12/2014	J. Vatjus-Antilla	Cyber contributions
V1.1	19/12/2014	A. Doumanoglou	Review and edit
V1.2	8/1/2015	D. Zarpalas	Content restructuring and update
V1.3	11/2/2015	D. Zarpalas	Content restructuring
V1.4	27/02/2015	N. Vretos	Contributions in section Network monitoring and adaptive compression
V1.5	27/02/2015	M. Lazaridis, C. Athanasiadis	Adding 1st version of traceability matrix
V1.6	27/2/2015	J. Vatjus-Antilla, N. Vretos	Integrated contributions from CYBER
V1.7	28/2/2015	D. Zarpalas N. Vretos	Updated traceability matrices
V1.8	6/3/2015	N. Vretos	Updated traceability matrices
V1.9	10/3/2015	B. Pussard, M. Lazaridis	Updated traceability matrices Integrated ARTS contribution
V1.10	11/3/2015	S. Crowle, N. Vretos	Integrating ARTS and IT-innov input
V1.11	11/3/2015	N. Vretos, D. Zarpalas	Overall refinement
V1.12	13/3/2015	N. Vretos, D.Zarpalas	Added input from ARTS and IT-innov about 3D scene modelling and references about golf physics. Overall

			refinement.
V2.1	18/3/2015	N. Vretos, J. Vatjus-Antilla	Added input in traceability matrix from CYBER.
V2.2	19/03/2015	D. Zarpalas	Minor modifications towards finalization
V2.3	30/03/2015	D. Zarpalas, P. Daras	Final modifications.

Contents

1 Executive Summary

This deliverable provides a deep, technical description of the modules of the 3D-LIVE platform. Specifically, analysis is based on requirements elicited through extended experimentation procedures in the project, feedback received from end users (through the 3D-LIVE co-creation activities) and technical requirements defined by the 3D-LIVE designers. The descriptions provided in this deliverable cover the whole gamut of 3D-LIVE activities, both regarding software engineering and hardware selections. The analysis of the 3D-LIVE platform modules is intended to address the development of similar applications and, thus, use-case specific descriptions have been avoided and only added to specify the particular utility of a function or component. To that aim, a generic architecture is given and, only where necessary, use case-specific explanations are added.

1.1 Summary of 3D-LIVE Project

The 3D-LIVE project aims at developing and experimenting on a User Driven Mixed Reality and Immersive platform, connected to EXPERIMEDIA facilities (FIRE testbeds), in order to investigate the Future Internet (FI) requirements to support Real-Time immersive applications, as well as to evaluate associated services both in terms of Quality of Experience (QoE) and Quality of Services (QoS). The main objective of 3D-LIVE is to explore 3D/Media technologies and Internet of Things (IoT) in order to construct applications bringing together real and virtual environments, among distant users with varying requirements. The combination of FIRE testbeds and Living Labs will enable both researchers and users to explore Future Internet capabilities to enter the Tele-Immersive (TI) application market and to establish new requirements for Internet technology and infrastructure. It is expected that, combining FI technology and TI market, will promote and accelerate the creation and adoption of innovative applications based on FI Services.

1.2 Purpose and Scope of the document

Until today (M28), the consortium has attained a series of achievements: The first and second prototypes of the platform have been delivered and their functionalities have been described in D3.3.1 (D3.1 after DoW amendment) and D3.3.2 (D3.2 after DoW amendment), respectively. The third prototype of the platform has been finalized (M25) and, currently, it is being evaluated through LIVE 3 activities. This deliverable givesa thorough description of all building components of the platform; through this technical description, it is expected that researchers, designers and developers will be provided with an analytical, technical methodology for building tele-immersive applications, following a multi-faceted strategy, driven by technical and user-related requirements. The final prototype (3rd) will be documented in D3.4 (D3.3.4 before DoW amendment) (M30).

After all the technical details of sections 2 and 3, the traceability matrix is presented in section 4, to provide insights on how the modules were implemented based on updated requirements, gathered from technical feedback during the live experimentation. More information about this iterative development approach can also be found in D3.2 and D3.4.

1.3 Related 3D-LIVE Documents

AD(1). 3D LIVE DoW

AD(2). 3D LIVE D1.1 Investigate and Formalise an Experiential Design Process

AD(3). 3D LIVE D1.1 Study and Create the Holistic User Experience Model

AD(4). 3D LIVE D2.1 Report on the Conceptual Design of the 3D-LIVE Platform

AD(5). 3D LIVE D2.2 Report on the Needs and Requirements of the 3D-LIVE Platform

AD(6). 3D LIVE D3.3.1 First Prototype of the 3D-LIVE platform

AD(7). 3D LIVE D3.3.2 Second prototype of the 3D_LIVE platform

AD(8). 3D LIVE D4.1 First Report on the experimentation & evaluation of the 3D-LIVE Tele-Immersive Environment

1.4 Brief work packages description

Three main activities have been considered in 3D-LIVE: i) Concept Modelling (WP1), ii) Exploration and Prototyping (WP2 & WP3) and iii) Experimentation and Evaluation (WP4). WP5 and WP6 are continuously active during the project, as they support dissemination/exploitation activities (WP5) and management (WP6). 3D-LIVE is following an iterative procedure in these activities, with all three of them giving feedback to one another, in a closed-loop fashion. This has been organized into 3 prototypes of the platform (prototypes 1,2 & 3), corresponding pilot activities (LIVE 1,2 & 3).

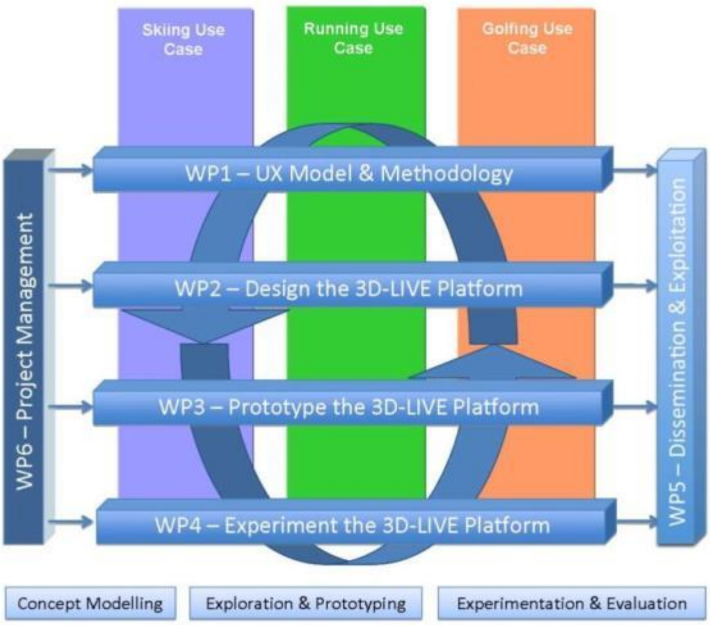


Figure 1: Functional description of work packages

Positioning in the current deliverable

Work package 3 (after contractual amendment following EC recommendations), consists of the following tasks:

- T3.1: 3D reconstruction in real time (Leader: CERTH, Partner: IT Innovation)
- T3.2: Activity Recognition, through data fusion of visual and other sensorial data (Leader: CERTH, Contributors: IT Innovation, ARTS, SportsC, CYBER)
- T3.3: Data compression and transmission (Leader: CERTH, Contributors: ARTS, SportsC, Cyber)
- T3.4: Rendering and Visualization (Leader: ARTS, Partners: CERTH)
- T3.5: Integration into the 3D-LIVE Tele-Immersive Environment (Leader: ARTS, Cyber)
- T3.6: Network monitoring and content adaptation (Leader: IT-IN, Contributors: CERTH, ARTS, SportsC, Cyber)

Tasks T3.1-4 and T3.6 (newly added task) are related to technical components that group tools, modules, functionalities, communication protocols and interfaces, which are integrated through Task 3.5. Consequently, this deliverable gives a complete description of activities performed through tasks T3.1-4 and T3.6, with D3.3.1 (D3.1 after DoW amendment) , D3.3.2 (D3.2 after DoW amendment) and D3.4 (first, second and third prototype description) focusing on activities related to T3.5, integration and functional interdependencies of the ecosystem building blocks.

Figure 2 presents the generic concept behind the 3D-LIVE deployment. The specific logic has been followed across all implementation and experimentation cycles of the project.

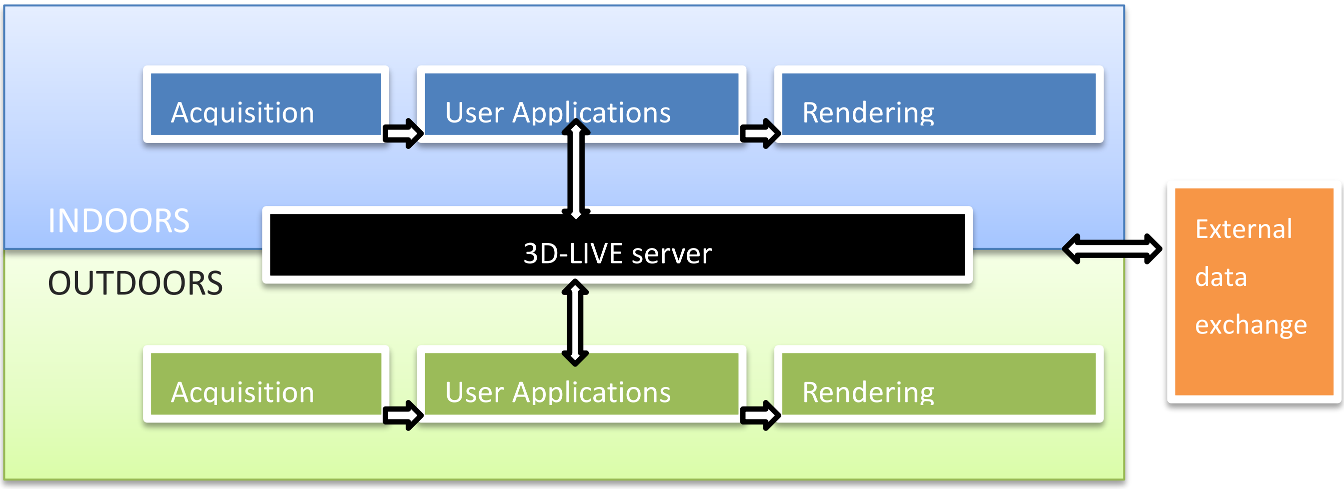


Figure 2: High level description of the 3D-LIVE platform, differentiating the modules to indoors and outdoors

As can be seen in Figure 2, the 3D-LIVE platform is deployed in 2 settings, involving completely different user and system requirements:

1. Indoor environments
2. Outdoor environments

Indoor and outdoor environments are the physical locations where the 3D-LIVE users are residing. They both consist of the following building components: Acquisition, User Applications and Rendering. Additionally to these two deployments, the external data exchange tool handles weather data fetching and includes a mechanism for sharing evaluation metrics for the modules. Central to the 3D-LIVE ecosystem is the 3D-LIVE server, responsible for handling data communications, network monitoring and game server applications.

Specifically, the building components of the 3D-LIVE platform are summarized as follows (refer to Section 1.5 for a detailed description).

Component Type	Component Name	Indoor	Outdoor	External Data Exchange
Sensor	Microsoft Kinect	X		
Sensor	Motion Sensors	X	X	
Sensor	Gyroscope	X	X	
Sensor	RGB Cameras	X		
Sensor	GPS Sensor		X	
Sensor	Microphone	X	X	
Sensor	Weather Conditions Sensor (Sensordrone)		X	
Equipment	Ski Simulator	X		
Equipment	Treadmill	X		
Equipment	Golf equipment		X	
Equipment	Mobile devices		X	
Equipment	Headsets	X	X	
Equipment	CAVE Systems, Wide displays	X		
Equipment	Sound Systems	X		
Equipment	Touch screens		X	
Equipment	Recon / AR Masks		X	
Software	3D-LIVE Capturer	X		
Software	Game Client	X	X	
Software	Game Server			X

Software	Environmental Observation Service (EOS)		X	
Software	Environment Reconstruction Service (ERS)			X
Software	RabbitMQ Server			X
Software	EXPERImonitor			X

Table 1: 3D-LIVE building components

The components mentioned above constituted the starting points of 3D-LIVE for the development of a dynamic ecosystem and the creation of novel instruments in 3D human reconstruction, compression and transmission, content adaptation to available bandwidth and activity analysis. Moreover, the 3D-LIVE consortium proposes solutions regarding environment reconstruction (weather and 3D modelling) and voice communications; all the above can be used as part of an overall methodology for highly innovative tele-immersive experiences. Specifically, in order WP3 to build the innovative 3D-LIVE tele-immersive platform, the components of Table 2 were developed, where the degree of innovation is also mentioned, associated with each of them.

Task	**Module**	**Degree of innovation**
3.1	3D modelling of the scene	INTEGRATION OF EXISTING TECHNOLOGY
	Weather data fusion	INTEGRATION OF EXISTING TECHNOLOGY & NEW METHODOLOGY
	Avatar animation	RESEARCH, INTEGRATION OF EXISTING TECHNOLOGY & NEW METHODOLOGY
	Real time, 3D reconstruction of moving humans	RESEARCH
3.2	Speed estimates	NEW METHODOLOGY
	Activity performance	RESEARCH
3.3	Voice communication	INTEGRATION OF EXISTING TECHNOLOGY

	Communication protocols	INTEGRATION OF EXISTING TECHNOLOGY
	Compression of human 3D reconstructions	RESEARCH
	Audio data compression & transmission	INTEGRATION OF EXISTING TECHNOLOGY
3.4	Interplays among input devices and sensors	INTEGRATION OF EXISTING TECHNOLOGY
	User interface	INTEGRATION OF EXISTING TECHNOLOGY
3.6	Network monitoring & Adaptive compression	RESEARCH
All WP3	Evaluation methodology of 3D-LIVE modules	NEW METHODOLOGY

Table 2: Degree of innovation for different 3D-LIVE components

In the following sections, the above building components are analysed with regards to the services offered, the supporting technologies, along with the reasoning behind the choices made in the project.

1.5 Structure of the Deliverable

Contrary to D3.1, D3.2 and D3.4, this deliverable describes the 3D-LIVE platform modules independently of its use-cases (jogging, ski, golf). However, given the different game engines used in the scenarios (jogging: RealXtend, ski/golf: Unity3D) some of the modules are described based on the game engine used. Furthermore, when certain functionality tailored to use-case specific requirements needs to be highlighted, proper explanations are given. The structure of this deliverable is as follows: Section 2 presents the technologies that constitute the basis for the 3D-LIVE deployment: it gives an analytical technical description, aligned with the rationale behind choices in software and hardware tools made by the 3D-LIVE consortium. It describes all the deployment components of the 3D-LIVE platform and the implemented modules, along with details of their implementation. In the cases where the implemented modules constitute research and technological advancements of the 3D-LIVE project, beyond existing technologies and methodologies, details of their implementation are described in Section 3 (while references are given in Section 2 wherever necessary). Section 4 explains, in the from a traceability matrix, the development's evolution based on the updated requirements gathered in the course of the 3D-LIVE's official experiments. Finally, section 5 offers the conclusions of this work.

2 3D-LIVE core technologies

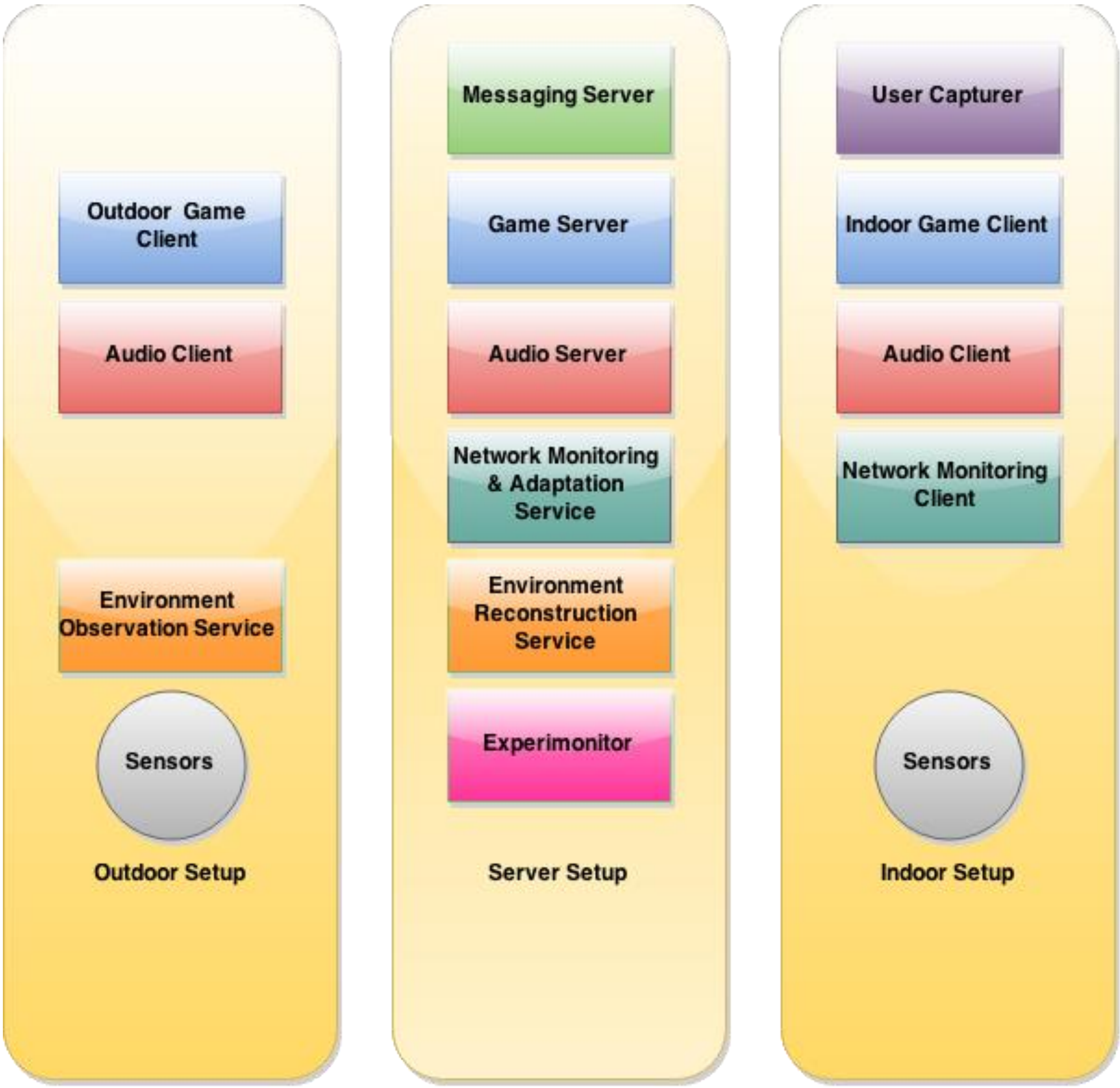


Figure 3: 3D-LIVE Platform deployment

In this section, the 3D-LIVE Platform deployment, whose conceptual model is described at the 3D-LIVE document D2.1 “Report on the conceptual model of the 3D-LIVE platform”, is discussed. In other words, what follows here is the description of how the conceptual model is actually deployed in software & hardware. The 3D-LIVE platform deployment is given in Figure 3. The different components of Figure 3 are separated in two categories: server side components (Server Setup) and client side components

(Indoor/Outdoor Setup). Server side components include the Messaging Server, the Game Server, the Audio Server, the Network Monitor and Adaptation Service, the Environment Reconstruction Service and the EXPERImonitor. For the indoor users, client side components include Sensors, the Indoor Game Client, the Audio Client and the User Capturer software, while for Outdoor users they include Sensors, the Environment Observation Service, the Audio Client and the Outdoor Game Client. One key element in Figure 3, which takes part in almost all interactions among the different components, is the Messaging Server. The Messaging Server used in 3D-LIVE is realized by RabbitMQ (section 2.4). RabbitMQ is a messaging broker, enabling easy realization of network messaging across different applications based on the publisher/subscriber model. Any application that has to publish information to others can publish on RabbitMQ and then RabbitMQ will replicate the message to its subscribers. In the beginning, it all starts with sensorial acquisition, being either motion sensing and 3D data capturing of the indoor users or GPS locations and motion sensing from the outdoor ones. Sensorial acquisition (section 2.1) is performed via User Capturer and the Game Clients (sections 2.8, 2.9), with each component being responsible for a different kind of sensors, as will be discussed later in this document. 3D Reconstruction (section 3.1.4) and activity recognition (section 3.2) is performed by the User Capturer and the information is distributed to the Game Clients via RabbitMQ messaging. The functionalities of the User Capturer software are realized by the 3D-LIVE Capturer (Section 2.8) component. Prior distributing 3D-Reconstruction data, the User Capturer software queries the Network Monitoring and Adaptation Service (section 2.12) about the data's compression level. To accomplish the latter, the Network Monitoring and Adaptation Service communicates with the Network Monitoring Clients (section 2.12) in order to determine the performance of the network. Then, the suggestion about the 3D-Reconstructed data's compression level is made available to User Capturer via the Messaging Server. The Game Server (section 2.3) is the component being responsible for maintaining the game state and syncing game clients. Game data is exchanged between the game server and game clients. In addition, a separate Audio Server (section 2.6) is responsible for audio communications. Indoor and outdoor users use specific audio clients (section 2.10) that are aligned with the audio server. This enables user interaction during gameplay. The Environment Reconstruction Service (ERS) obtains data from the Environmental Observation Service (EOS) (section 2.5) through RabbitMQ, fuses the information with the data provided from the internet and outputs to the RabbitMQ. The weather data, which is the output of ERS, is consumed directly by the game clients. As mentioned before, in 3D-LIVE, real-time monitoring of technical Quality of Service (QoS) metrics are collected along the whole game pipeline. This responsibility is carried out by EXPERImonitor (section 2.11). EXPERImonitor collects QoS metrics by the following components: 3D-LIVE Capturer, Indoor & Outdoor Game Clients, and the Game Server. The metrics are then logged and post-processed to evaluate the 3D-LIVE platform.

In the following subsections, each of the individual core components of Figure 3 is described in detail. In summary, this includes: Sensors (2.1), Game Server (2.3), RabbitMQ Server (2.4), Environment Reconstruction Service & Environment Observation Service (2.5), Audio Server (2.6), 3D-LIVE Capturer (2.7), Indoor Game Client (2.8), Outdoor Game Client (2.9), and Audio Clients (2.10), EXPERImonitor (2.11).

2.1 Sensors

Sensors, in 3D-LIVE, refer to any hardware module responsible for human sensing, location acquisition, audio input and real-time weather conditions sensing.

2.1.1 *Human Sensing in 3D-LIVE*

Microsoft Kinect Depth Sensors:

The Kinect RGB-D sensor features i) a regular RGB camera and ii) a depth sensor, consisting of a stereo pair of an InfraRed (IR) projector and an IR camera (monochrome CMOS sensor), with a baseline of approximately 7.5cm. The depth values are inferred by measuring the disparity between the received IR pattern and the emitted one, which is a fixed pattern of light and dark speckles. The Kinect driver outputs a $N_x \times N_y = 640 \times 480$ depth grid with a precision of 11 bits at 30 frames/sec. The RGB image is provided in the same resolution and frame rate with the depth data. According to the technical description in the Kinect-ROS (Robot Operating System) wiki[1], the RGB camera's horizontal Field of View (FoV) is approximately 58^{o}, while the depth camera's horizontal FOV is approximately 63^{o} and vertical FoV is

[1] http://www.ros.org/wiki/kinect_calibration/technical

approximately 45 ^{o}. Microsoft officially specifies that the Kinect's depth range is 1.2−3.5 meters, but it can be experimentally verified that the minimum distance can be as low as 0.5 meters and the maximum one can reach 4 meters. Essentially, the IR projector and camera constitute a stereo pair, hence the expected precision on Kinect's depth measurements is proportional to the square of the actual depth. The experimental data presented in the ROS wiki[2] confirm this precision model, showing a precision of approximately 3mm at a distance of 1 meter and 10mm at 2 meters. The accuracy of a calibrated Kinect sensor can be very high, of the order of ±1mm. Additionally, the Microsoft Kinect v2.0 Sensor provides Improved body tracking, Higher depth fidelity and 1080p HD video. Microsoft Kinect v2.0 was utilized in the 3rd 3D-LIVE Prototypes for improved body tracking enhancing the performance of the activity recognition module to respond to the users' feedback of erroneous speed estimates in the jogging scenario.

Suitability in 3D-LIVE: The Microsoft's Kinect sensor has attracted the attention of many researchers, as well as home enthusiasts, due to a series of advantages: accuracy, realtimeness (30Hz), reasonable range (suitable for the 3D-LIVE applications), lowcost (suitable for Low-Cost deployments in 3D-LIVE), portability. In 3D-LIVE, Microsoft Kinect has been used to perform motion capturing for avatar animation (Section 3.1.1), 3D-Reconstruction (Section 3.1.4) and Activity Recognition (Section 3.2).

Wii Balance Board by Nintendo:

The Wii Balance Board is a game controller manufactured by Nintendo, which is an accessory for the Wii and Wii U game consoles. It is shaped like a household body scale, powered by four AA batteries, and connected by Bluetooth. It contains four pressure sensors that are used to determine the user's weight and centre of balance (the projection of the centre of mass on the surface of the Balance Board).

[2] http://www.ros.org/wiki/openni_kinect/kinect_accuracy

According to the manufacturer, the system's ability to measure weight is probably more accurate than a typical bathroom scale. The pressure sensors can accurately measure up to 150kg and the physical structure can withstand a maximum force equivalent to about 300kg.

As this device is not designed for computer use, we use the GlovePIE middleware in order to read the sensor's data.

Suitability in 3D-LIVE: This device is a low-cost system (suitable for Low-Cost deployment in 3D-LIVE) that allows to measure accurately user's centre of balance. It will allow a dynamic control of the avatar for indoor user for the ski scenario, providing a much more immersive way to interplay.

Inertial Measurement Unit (IMU) EXL-S3 by EXEL:

The EXL-S3 sensor is an Inertial Measurement Unit manufactured by EXEL. It is a motion sensor whose task is to calculate the orientation and displacements of a body in space. Thanks to the Micro-Electro-Mechanical-Systems (MEMS) technology implemented in computer chips, they are small and low-energy consuming. They are not suitable for inertial navigation (including x,y,z coordinates) calculation. This would require a double integration of the accelerometer signal, so even small errors sum up very fast (few seconds). This drift could be corrected using other external data (GPS, Kinect...).

The gyroscope captures all rotations in space. Calculating the orientation is a single step of integration, generating a drift over time, but that can be controlled. The static accuracy of the gyroscope is 0.5 degrees, and the dynamic accuracy is about 2 degrees. The data transmission rate is 100Hz. Data are sent wirelessly using Bluetooth technology. Battery is internal and battery life is about 4 hours. The housing form is 54x33x14.

Suitability in 3D-LIVE: In 3D-LIVE, only the gyroscope orientation provided by the IMU is used. EXL-S3 has good orientation performance in terms of accuracy and transmission rate, suitable for the needs of 3D-LIVE. Moreover, it uses Bluetooth technology and thus it can be directly connected to a smartphone for using inside the outdoor game client application.

2.1.2 *Location sensing in 3D-LIVE*

Global Positioning System

In 3D-LIVE, the scenario needs led us towards the choice of embedded GPS systems inside Smartphones and Tablets for outdoor users. More specifically, in the outdoor jogging scenario a standard Android Smartphone GPS was used along with integrated Assisted-GPS (A-GPS) functionality. The device containing the GPS functionality was chosen to be Samsung Galaxy S2+ (SGS2+) which supports a variety of position tracking technologies, including also GLONASS. The accuracy of location tracking using the SGS2+ in city environment of Oulu was measured to be up to ±5m. For the Golfing scenario, a Sony XPeriaZ2 tablet was chosen, as it supports American and Russian GLONASS systems for a better accuracy. The location accuracy measured during experiments was about 4 meters in average. For the Skiing scenario, the Samsung Galaxy S4 was deployed, supporting GPS, A-GPS and GLONASS as well. Accuracy measured during experiments was about 3meters at low speeds and 8 meters at higher speeds.

Suitability in 3D-LIVE: The devices chosen are perfectly suited for 3D-LIVE-applications because of low-cost, reasonably accurate embedded GPS tracking, light weight and good battery life.

2.1.3 *Audio sensing*

Microphones

General-use headsets with built-in microphones have been practically used in all scenarios. Standard off-the-shelf Procaster BH-03 Bluetooth headset was used in jogging scenario both indoors and outdoors. Logitech H600 (Bluetooth wireless headset) was used in both skiing and golfing scenario by indoors players, while a wired headset was used outdoor in order to let all the Bluetooth bandwidth for motion sensors.

Suitability in 3D-LIVE: The headsets used in the project have microphones with substantially good noise-cancelation characteristics , and suit well the scenarios where people need communicate with each other in noisy environments.

2.1.4 *Environment sensing*

Two sources of environmental information are used. The wide-area sensing data is used for outdoor-situation (e.g. weather conditions) simulation in the virtual environment. This data is provided by a 3rd-party service, the Open Weather Map (OWM), which is a Web-based service and provides free weather data from more than 40,000 weather stations and from global meteorological broadcast services. The mobile sensing data are obtained via a wearable sensor device (SensorDrone) that is placed on the body of the outdoor player. Such mobile sensing data represent the most localised environmental information and has high-frequency updates.

Suitability in 3D-LIVE: OWM is a weather data and forecasting service, which provides open-access Web APIs. It offers a wide range of weather data, including precipitation, wind, clouds, temperature, etc. Most importantly, the offered APIs allows one to obtain on-demand weather data of a given point (i.e.

coordinate), which is a desired feature of 3D-LIVE Environment Reconstruction Service (ERS). SensorDrone is a good wearable environmental sensor device, given its small size, reasonable range of environmental metrics, and low costs. It has integrated sensors for measuring temperature, humidity, pressure, proximity, illumination and colour intensity. The product comes with an Android SDK allowing custom app development to feed measurement data to any Android-compatible phone via Bluetooth.

2.2 Sports Equipment

Treadmills

- Low-Cost Jogging deployment, Thessaloniki: A Treadmill 2063CA (speed 1-16km/hr and incline 0-15%) was purchased.
- High-End Jogging deployment, Oulu: A MT15 Carl Lewis fitness incline treadmill was purchased.

Suitability in 3D-LIVE: Given that there are no strict specifications about the treadmills within 3D-LIVE scenarios, the abovementioned ones are well suited for the purposes of the system.

Ski simulators

- Low Cost Ski Deployment, Thessaloniki: A Balance board (Fit Plus with Balance Board Wii)
- High End Ski Deployment. Ski simulator in LAVAL: Basic Ski Simulator by PRO SKI – Simulator :

The system has a support system and fixtures for ski-poles. The movement takes place in one plane. It is made of four wheels fixed on a pan, which roll in two curved runners. An elastic bands based system provides six intensity levels. This model is not equipped with sensors, so we used an IMU that we are already using for other purpose (EXL-S3 sensors).

Suitability in 3D-LIVE: This model of the PRO SKI-Simulator series is suitable for beginners as well as for advanced skiers. The intensity adjustment allows adapting to any type of user pattern. It provides smooth simulation of ski movements, and was used in several other virtual reality projects. It seems to be the compromise chosen by researchers for a high-end control and a reasonable price (around 800 euros), allowing sportive training as well as gaming input.

Golf equipment

- For the golfing scenario, users could use two kinds of clubs: an Iron 6 with carbon shaft and a putter. While the outdoor user hits normal golf balls, the indoor user plays with practice air balls (lighter than normal balls to prevent any accident inside the simulator). The outdoor player has a caddy, on which he can attach the tablet.

Suitability in 3D-LIVE: using clubs with carbon shaft do not disrupt motion sensors, since devices using MEMS technology can be affected by ferromagnetic materials.

2.3 Game Server

In 3D-LIVE two different game engines have been used while targeting different scenarios. The jogging scenario is supported by a game engine developed using the RealXtend platform, while the golfing and skiing scenarios are supported by a game engine developed using Unity3D. Details supporting this choice are provided in D3.2, Section 2.2. This decision affects both Game Servers and Game Clients (sections 2.8 and 2.9). Regardless of the development platform, the two Game Servers share common responsibilities: communications management, game state tracking, game client synchronization and game engine data flow. In the rest of the section, in-depth details are given for the responsibilities and the implementation of each Game Server.

RealXtend Game Server

- The RealXtend architecture is built upon client-server architectural model. In client-server model tasks are split between clients, which request services, and servers provide services assets for the clients. In a game specific scenario a service can range from simple location update of a game component or downloading an asset for example a texture or 3D-mesh. This model enables lightweight client application installer, but on the other hand it requires reliable and relatively high bandwidth network connection to the server if large amounts of graphical assets are required to load for the client. After the possibly lengthy initial asset loading, client only needs to update the user input to the server which has very miniscule bandwidth requirement. Download bandwidth requirement is also relatively low after the initial asset download, only needing to update current state of game objects in the game scenario from server to client.
  The server handles all the heavy lifting in realXtend architecture. The clients send their user input which is then used in the server to calculate the physical and graphical effects in the gameplay. This is called server authoritative gameplay mechanic. The server gathers all the input data from the clients on one frame, calculates the state of the objects and then sends the modified data back to the clients. Example data contained in the updated state sent by the server contains: objects' position/rotation/scale, physical impulses causing game object trajectory change, object velocity updates and other physical characteristic changes. The clients continuously do interpolation between

successive states and update their local gameplay status according to the server's status. This interpolation helps to smoothen the game under bad network conditions, where some network packages might arrive late or possibly never. RealXtend uses Ogre3D as its underlying render engine. Ogre3D is open sourced and updated by hundreds of community users. It is one of the most actively developed open sourced 3D engines to date.

Suitability in 3D-LIVE: RealXtend client-server architecture allows centralized asset management and lightweight client installation for the jogging scenario which enables easy and centralized modifications and deployment of new assets, as required in 3D-LIVE.

Unity Game Server

- The Unity3D server was used for the skiing and golfing scenarios. It handles various essential features: user registration, game client disconnection/reconnections, game play, game data storage and monitoring. The server is made of two main parts: "pregame" and "in game". In the "pregame" part, the server can be set up and is not running. The parameters that can be set include: the number of players, which users can join the game and data analysis activation (connection to the EXPERImonitor (section 2.11) and logging players' positions). By pressing the "Go" button, the server will be running, waiting for players to join the game. When the players are ready, the server starts synchronizing the game state between users. It will manage game state changes in accordance with users' progression or through server side control (only for debugging). The server is saving game data, thus when a player reconnects after a disconnection it will reload those data. The last feature is the monitoring of the game: choose experiment options (enable/disable full body 3D reconstruction, put gates on the slope for the ski scenario), manually change game phase in case of trouble and end the experiment.

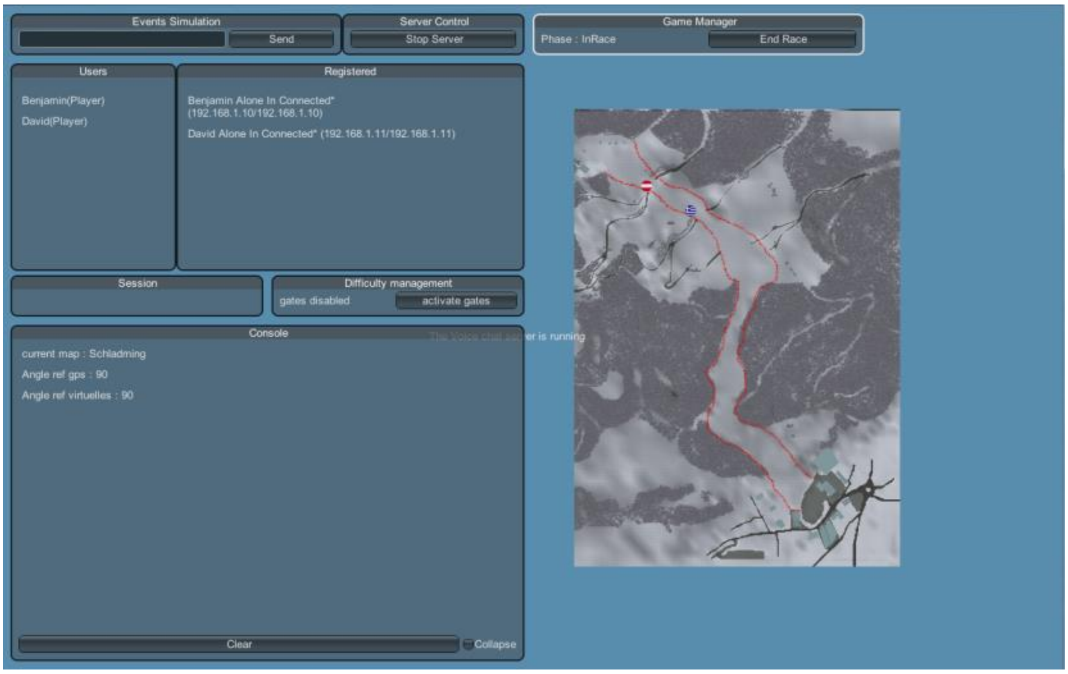


Figure 4: Ski Unity3D Game Server

Suitability in 3D-LIVE: As Unity3D is a cross platform engine, clients can be easily connected regardless of their platform. As indoor users and outdoor users are never using the same platform, (considering tablets, Smartphones and computers) this is really relevant.

2.4 RabbitMQServer

The RabbitMQ server is a message brokering system that implements the exchange and delivery of messages using the Advanced Message Queuing Protocol (AMQP). RabbitMQ is a well-established and proven AMQP message broker that provided robust messaging by offering concurrent message processing in an environment that supports clustering, federation and highly available message queues.

Suitability in 3D-LIVE: The AMQP messaging protocol and RabbitMQ server implementation was selected for use in a number of the network messaging scenarios within 3D-LIVE (for example, skeleton updates and experimental metrics delivery) where reliable and acknowledged delivery of data across the network was required. The RabbitMQ/AMQP solution was also considered to be advantageous to the 3D-LIVE as it provides network communication that is accessible from a range of technical platforms used in 3D-LIVE (including Java, C# and C++).

2.5 Environmental Observation Service & Environment Reconstruction Service

2.5.1 *Environmental Observation Service*

The Environmental Observation Service (EOS) is an Android app developed in 3D-LIVE project. EOS app is installed on out-door player's Android mobile and paired with Sensordrone[3] device via Bluetooth. The EOS app provides user interfaces to control (i.e. start/stop) Sensordrone device and measurement data. In particular the EOS app displays the real-time measurements on screen and streams them along with other sensing data from Android phone (e.g. GPS location data) to remote Environment Reconstruction Service (ERS, see below).

Suitability in 3D-LIVE: EOS app provides an easy to use interface that enables gathering and streaming environmental data from Sensordrone and integrated sensors of Android phone in real time. Ultimately it is this data (once aggregated and fused by the ERS) that provides high resolution environmental effect data for the 3D-LIVE game engine to use in the rendering of weather effects in the virtual environment.

2.5.2 *The Environment Reconstruction Service*

The Environment Reconstruction Service (ERS) is an on-line application that provides real-time environment effect data for the 3D-LIVE game engine during game play. This is achieved through the provision of a) an environmental sensor aggregation services and b) the real-time aggregation and fusion of sensor data that can be translated into effect parameters on demand, via a web based query. In the 3D-LIVE project, we acquire environmental data from two primary source types: fixed, wide area, on-line

[3] http://sensorcon.com/products/sensordrone-multisensor-tool

sources (such as Open Weather Map[4]) and dynamic, wearable environment sensors (such as the SensorDrone). This provides us with varying degrees of data 'resolution' which we aggregate into a common model and fuse using a rapid interpolation process.

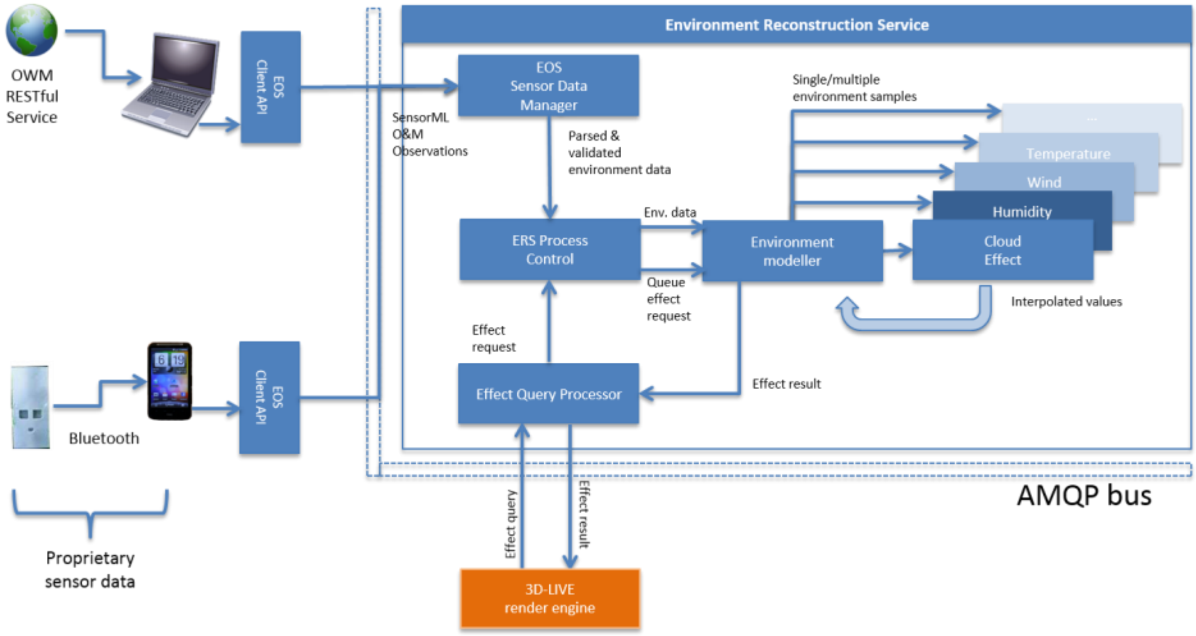


Figure 5: Environment Reconstruction Service

The ERS consumes environmental data from fixed site and dynamic locations using a common data API via a RabbitMQ/AMQP messaging server. The environmental sensor data types captured by the ERS

[4] http://openweathermap.org/

include:

Temperature	Degrees Celsius
Humidity	Relative humidity (in degrees)
Rain/snow fall	Millimetres/minute
Cloud coverage	As a percentage of the visible sky
Wind speed and direction	In miles/hour and degrees (0 to 360)
Illumination (local only)	In Lux
Wind speed and direction	In miles/hour and degrees (0 to 360)

Table 3: Environmental sensor data types captured by ERS

Fusion of all environmental data is performed using a common (real-time capable) interpolation method often used by in geo-sensor data research: the Inverse Distance Weighting (IDW) interpolation (see Resch et al., 2010[5] and Xia, Y. et al, 2011[6] for further information). Interpolated values are then used to provide effect data that includes quantitative (and qualitative) information regarding the local weather conditions

[5] Resch, B., Blaschke, T., Mittlboeck, M. (2010) Live Geography: Interoperable Geo-Sensor Webs Facilitating the Vision of Digital Earth. Int. Journal on Advances in Networks and Services. Vol. 3. No. 3 & 4. IARIA.2010.

[6] Xia, Y., Kuang, L., Li, X. (2012) Accelerating geospatial analysis on GPUs using CUDA. Journal of Zhejiang University Science Computers and Electronics. December 2011. Vol. 12. Issue 12. p990-999.

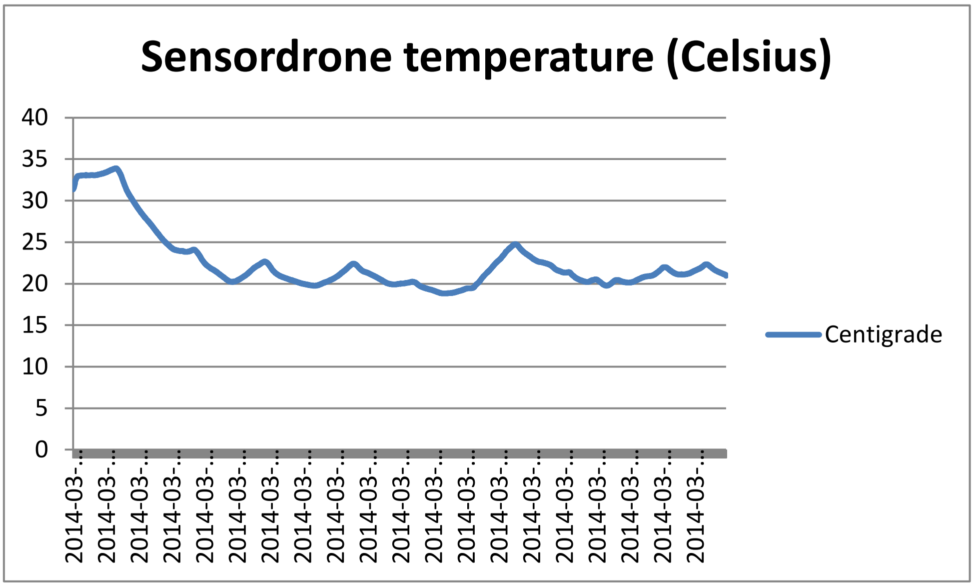
(using GPS as the point of interpolation during query).

The ERS provides a number of metrics to the EXPERImonitor (section 2.11) during 3D-LIVE run-time. These are summarised in the table below. For more detailed information on these metrics, please see the deliverable entitled D4.1: *Report on the experimentation and evaluations of the 3D-LIVE Tele-Immersive Environment.*

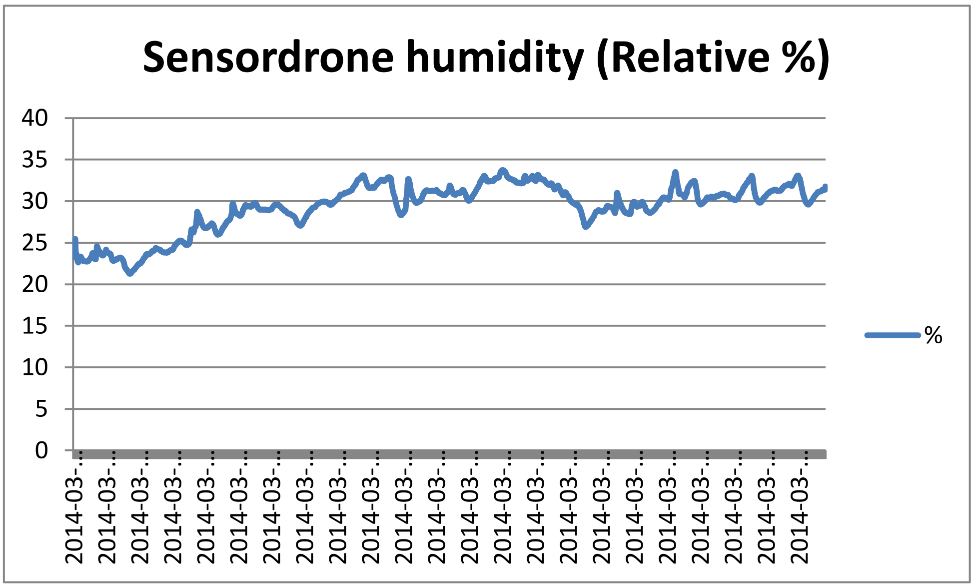
QoS metric for evaluation of *ERS*	Explanation
Effect Query Rate	Reports the rate at which the ERS receives effect queries from the 3D-LIVE rendering engine.
Effect type requests	Reports (nominally) on the effect type generated as a result of the ERS query made by the 3D-LIVE rendering engine. The difference in time between the query and the effect type result returned provides a synthesized metric indicating time required by the ERS to process the query.
Environment data sources (various)	The ERS consumes a variety of environmental measurements using a common environment metric model (see above). In LIVE 1 and LIVE 2 experiments, these values were passed on to the EXPERImonitor to allow comparison of the ‘raw’ environment samples compared with those interpolated by the modeller.
Environment modeller (interpolated) effects	The ERS interpolates all ‘raw’ environment data received from its clients using an Inverse Distance Squared function. When the 3D-LIVE rendering engine requests an effect, these interpolated values are provided both as quantitative and categorical values. The quantitative values are also passed on to the EXPERImonitor .

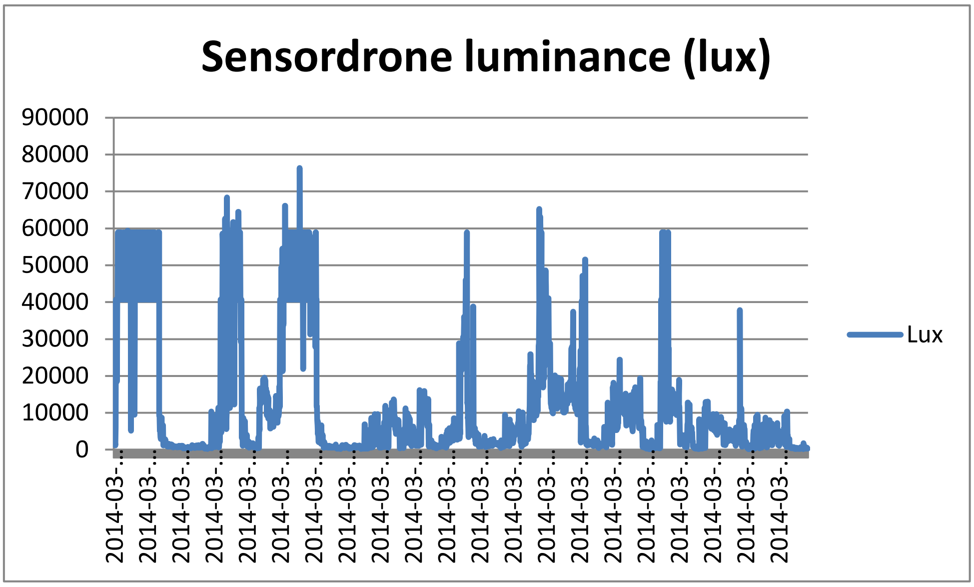
Table 4: QoS metrics for evaluation of ERS

The following figures present example data sets captured during the LIVE 2 skiing experiment using the EOS/Sensordrone client – the data was collected by the EXPERImonitor during experimentation.



Note: the relatively high temperature at the start of this series is the result of the device being held by the hand and in the sun during the initial set up of the ski run. The lower temperatures reflect the real outdoor temperature as measured by 3rd party temperature displays on the slope (it was a hot, sunny day).

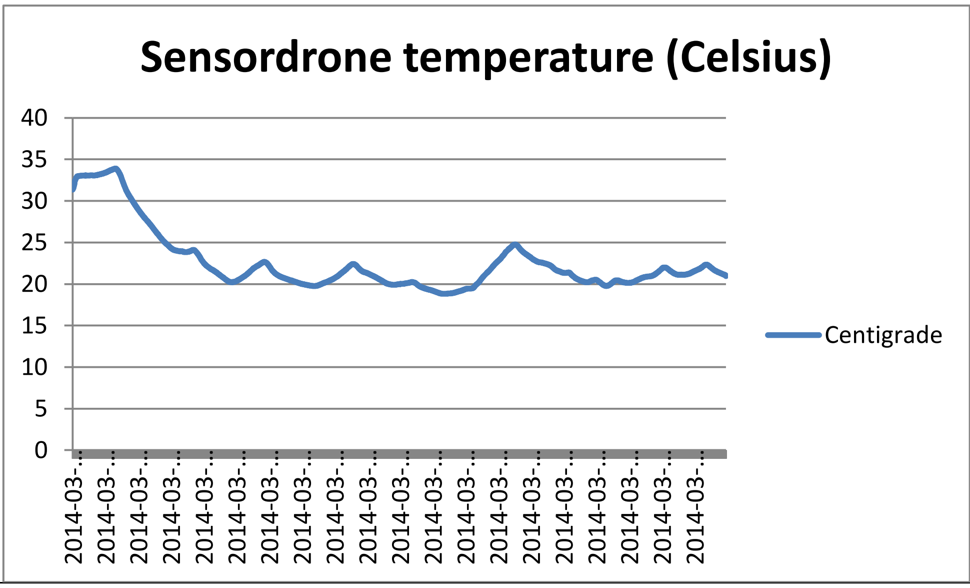




Note: in this run, the Sensordrone was not well fixed to the skier, causing it to move around and sometimes face internally towards the user (blocking access to direct sunlight). This data is later smoothed by IDW; see below.

Figure 6: Data collected by EOS/Sensordrone client

The following figures provide examples of IDW smoothing based on both OWM and Sensordrone data as it was captured and used to produce effect data.



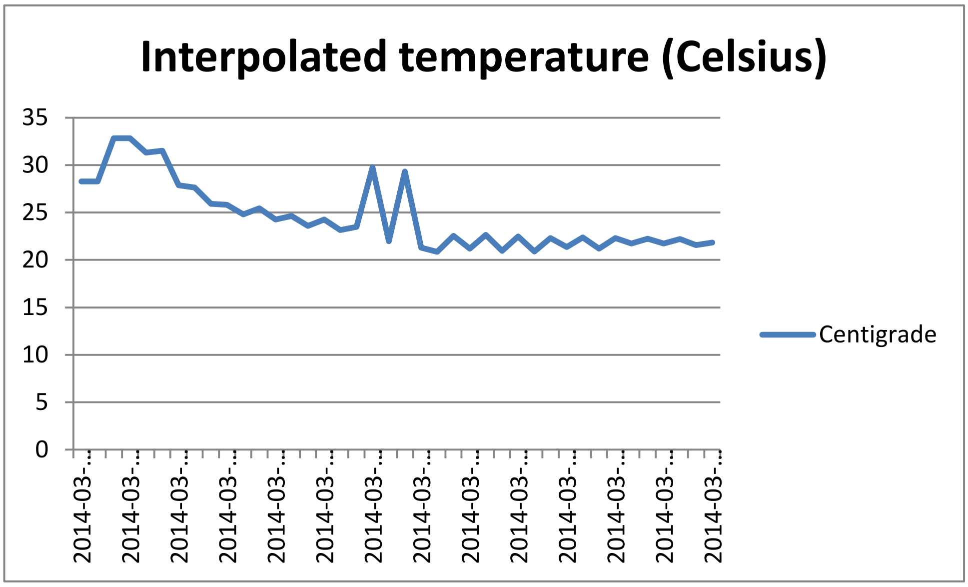
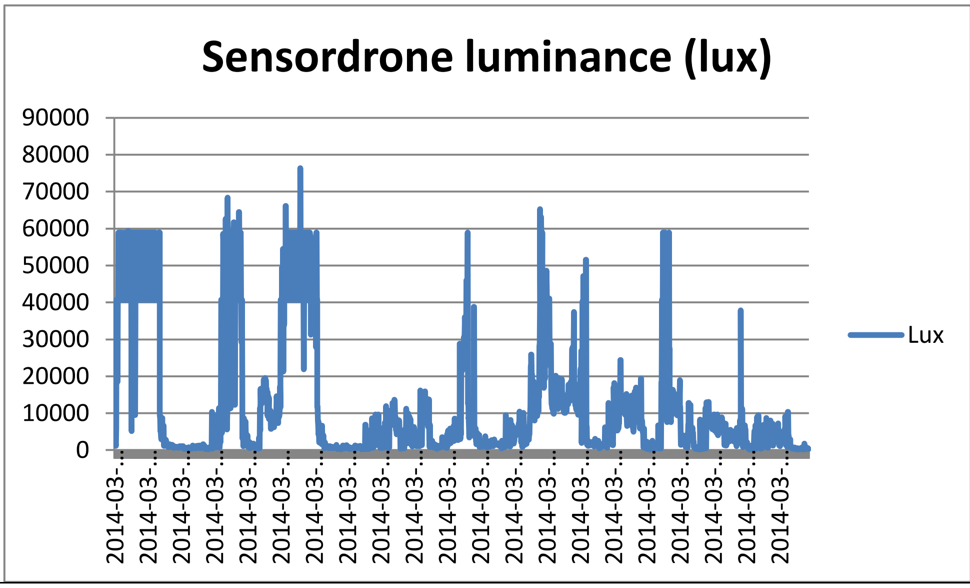


Figure 7: Sensordrone and Interpolated temperature values received by ERS at runtime

In the figures above, we see the interpolated temperature values (derived from location based queries received by the ERS at run-time) are presented. The overall series provides a smoothing of the more variable Sensordrone data (there seem to be two location based ‘hot spots’ where the query location is proximal to a density of high temperature samples).



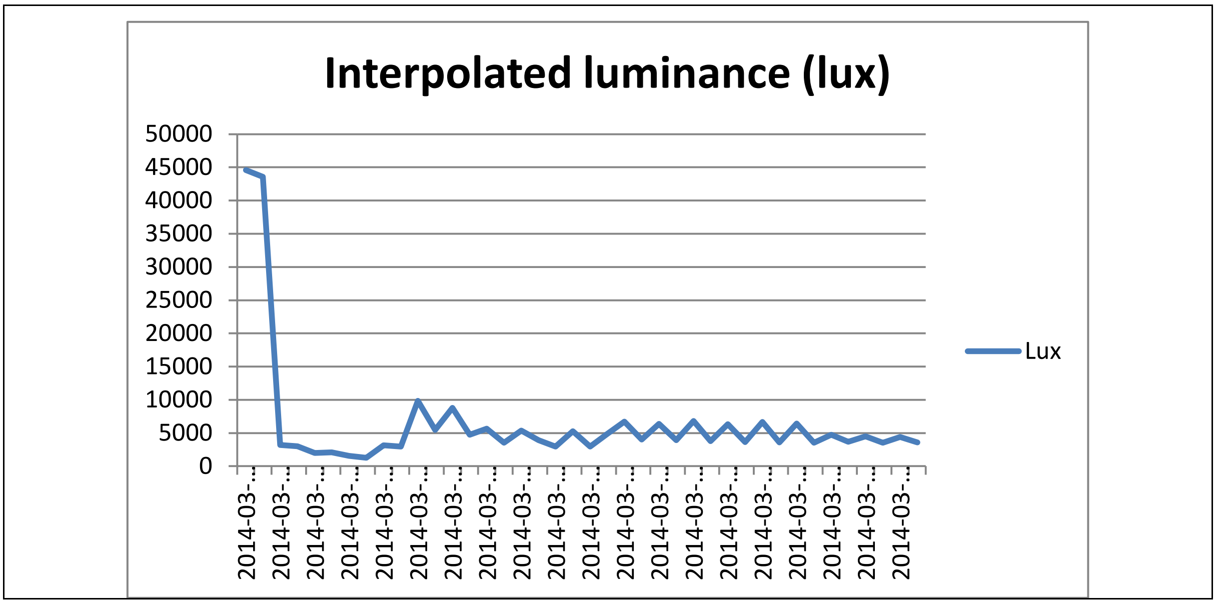


Figure 8: Sensordrone and Interpolated luminance values received by ERS at runtime

A similar smoothing can be seen with the noisy data sent from the (unsecured) Sensordrone during an experimental run. Whilst the local luminance data did indicate sunny day conditions (as high peeks) it was clear that the light sensor was pointing inwards a considerable amount of the time. This result provided us with important guidance relating to securing the Sensordrone to the user for future experiments.

Suitability in 3D-LIVE:

The ERS provides a light-weight and well aligned approach to delivering real-time environmental effect information to the 3D-LIVE gaming environment. Its design was developed with a view that multiple environment data sources (of varying implementation) could send data using a common data format that can be swiftly and reliably connected. Here we leveraged the RabbitMQ/AMQP communication service (already used for other 3D-LIVE communication) for this purpose. Consumers of the ERS effect data also use the RabbitMQ/AMQP API to easily access the latest environment effects using a location-based query.

2.6 Audio Server

Initially various Voice Over Internet Protocol (VoIP) applications were considered; Teamspeak, Ventrilo and Mumble. RealXtend (section 2.3) already contained mediocre support for mumble audio protocol when the decisions were made. This, in conjunction with Mumble being the only open-source software with permissive licensing, had a heavy impact on the resulting decision. Eventually, Mumble was chosen for voice transmission between indoor and outdoor clients. Mumble is a low latency and low bandwidth audio protocol which supports positional audio via Celt, Speex and Opus codecs. Mumble's network bandwidth requirements depend greatly on the audio quality settings. With no positional audio data mumble consumes about 15.8kbit/s on average. The default quality with positional audio takes up to 60kbit/s of bandwidth which is still acceptable for an average 3G mobile connection used in 3D-LIVE project. Mumble uses client-server architecture, which means it might end up bandwidth bound with extensive amount of users on one channel compared to peer to peer competitors but considering the scenario where concurrent users on same audio channel will remain under few hundred users, this is not a problem. In the Skiing and Golfing scenarios, initially, another solution was experimented (i.e. the Unity3D VoiceChat plugin), but it was found to cause trouble in terms of latency and Framerate to the mobile outdoor game client. Finally, the common technical choice was to use Mumble in all scenarios.

Suitability in 3D-LIVE: Easy to use, open-source and permissive licensing is well suited for project like 3D-LIVE. Protocol also works on mobile devices and desktops and scalability allows mobile usage on 3G- and 4G-networks.

2.7 3D-LIVE Capturer

The 3D-LIVE Capturer application is the realization of the user capturer software (section 2) in the indoor setup. The application comes in two versions: the full and the stripped down, targeting different cost solutions. The full version targets the low end setup that allows for 3D-Reconstructions of the users. On the other hand, the stripped down version targets the high end setup that supports user representation via avatar animation. 3D-LIVE Capturer captures user skeleton information using one or more Microsoft Kinect Sensors. When using more than one Kinects, the application will fuse the skeleton information obtained from the multiple Kinects into one final skeleton using the skeleton merging algorithm described in section 3.1.2. Skeletal information is used afterwards by the indoor game clients (section 2.8) in order to perform animation of the users' avatars. Special skeleton filter algorithms were developed and integrated in 3D-LIVE Capturer in order to further enhance the avatar's animation quality (section 3.1.1). Furthermore, the application is able to perform Activity Recognition for the use cases of Jogging and Skiing evaluating the user performance. The activity recognition algorithms used are described in detail in section 3.2. As described earlier, what the full version includes in addition, is the ability to capture and reconstructs 3D full-body representations of users, in real-time (section 3.1.4). These 3D reconstructions are then compressed and transmitted over the network to the game clients (section 3.3). The user skeletal data and the Activity Recognition Results are also transmitted in real-time to the game clients.

EXPERImonitor Quality metrics are also calculated upon request and are streamed to the provided EXPERImonitor Server. Particularly, the reconstruction's visual quality is evaluated with the method described in section 3.1.4 in real-time. The compressed mesh is also evaluated for its visual quality with the same method. All the parameters of the 3D-Reconstruction process and mesh compression are configurable within the application's user interface. The 3D-LIVE Capturer, while running, provides visual inspection of the Kinect depth and color streams along with skeletal information. Moreover, full-body 3D reconstruction visualization is provided in the full version. Activity recognition results are also displayed on-screen during the live capture. The performance of 3D-LIVE Capturer is critical for real-time immersive interactions and thus it is highly optimized for speed. It makes heavy use of multi-core CPU multithreading and algorithm parallelization for nVIDIA GPUs via CUDA. This enables for high reconstruction frame rates (up to 8 fps), high reconstruction streaming framerates (up to 8 fps) and minimum network delays. All the above is achieved via parallelizing the whole pipeline of capturing-reconstruction-compression-transmission, activity recognition as well as QoS metric calculation. Below, in Figure 9, two sample screenshots from the 3D-LIVE capturer are depicted.

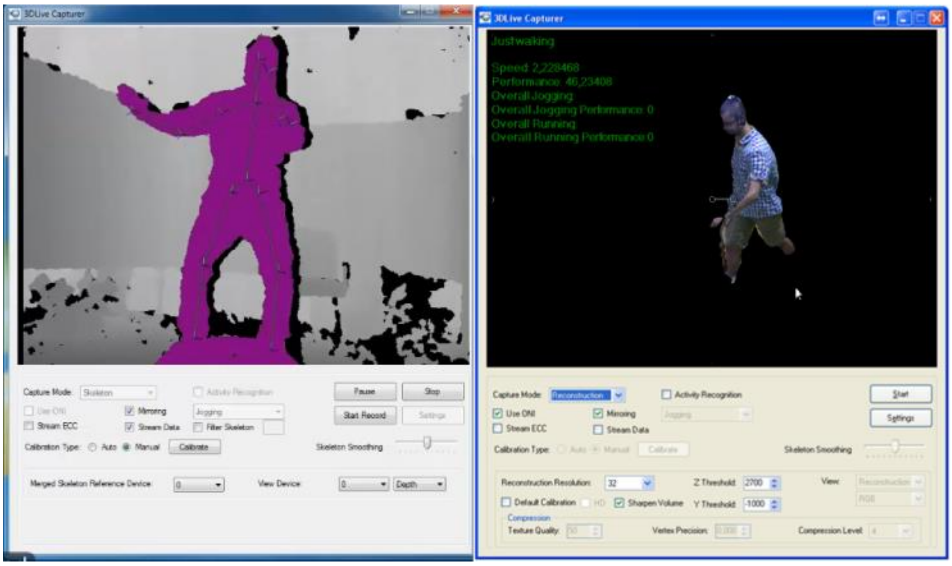


Figure 9: 3D-LIVE Capturer. Left: Skeleton tracking. Right: Full 3D reconstruction of moving human

Early versions of the 3D-LIVE Capturer utilized Primesense's OpenNI[7] and NITE to interface with Microsoft Kinects. During the early stages of the 3D-LIVE project, Microsoft Kinect SDK was rather immature (or even nonexistent) and OpenNI was the only, open and free, available option. After the first 3D-LIVE experiments took place, users reported low avatar animation quality, which was due to OpenNI skeleton capture and tracking quality. When the project advanced, the Microsoft Kinect SDK was mature enough and had already started to show skeleton tracking results of interesting quality in other applications. Consequently, CERTH, in order to address the users' feedback of low animation quality decided to replace the OpenNI/NiTE Kinect interface of the stripped down 3D-LIVE Capturer with

[7] Primesense was recently bought by Apple and an official link of OpenNi is, thus, not currently available.

Microsoft SDK, before the LIVE 3 experimentation cycle. Indeed, the latest version of the stripped-down 3D-LIVE Capturer that targets the high end setup, utilizes Microsoft Kinect SDK v1.8[8] which, along with the overall avatar animation quality, is by far superior to what was obtained by the previous interface.

In order to verify the appropriateness of this transition, CERTH conducted an objective evaluation of the two SDKs (MS SDK vs OpenNI), through EXPERImonitor metric analysis. This analysis is presented in Appendix B. Apart from the objective evaluation presented in Appendix B that vote in favour of the Microsoft SDK, using MS SDK in general it has additional benefits:

- is free to download & use,
- has both native C++ interface as well as .NET interface to use with C# and other .NET languages,
- recognizes and tracks moving people using skeletal tracking,
- determines the distance between an object and the sensor camera using depth data.
- captures audio using noise and echo cancellation or finds the location of the audio source,
- enables voice-activated applications by programming a grammar for use with a speech recognition engine.

Additionally, in the LIVE 3 experimentation cycle for the low-end setup, CERTH added the newly released Kinect v2.0, to support the full version of the 3D-LIVE Capturer with improved body tracking (thus allow for the enhancement of the avatar's motion animation of the high end users), which enhanced further the performance of the activity recognition module relying on skeletal data. This was done in order to respond to the users' feedback of erroneous speed estimates in the jogging scenario.

[8] http://www.microsoft.com/en-us/download/details.aspx?id=40278

2.8 Indoor Game Client

The indoor game client is one of the most important components of the 3D-LIVE platform deployment. This is due to the fact that the main responsibility of the indoor game client is the visualization of the virtual environment and the users' avatar or 3D reconstructed representation. All the users experience the sports activity and interact in a shared virtual space realized by the game client. Game clients integrate features such as environment reconstruction, avatar animation, 3D reconstruction rendering, game play mechanics via the activity recognition module's results, voice communication (for the skiing and golfing scenarios in early prototypes), user input via sensors (e.g. Wii, Pro ski simulator) and visualization of the weather and activity recognition results. Overall, for all the 3D-LIVE scenarios the whole game scenario is realized by the game clients.

2.8.1 *RealXtend Game Client*

For the indoor game clients of the jogging scenario the open source RealXtend virtual world platform was chosen. RealXtend speeds up the development of the global standardized 3D internet of virtual worlds by making the best technology available to everyone. With RealXtend networked multiuser 3D applications can be created, as the RealXtend SDK is a complete 3D multiuser application development kit. It is a modular framework that can be used both to run standalone applications (for example a single player game or a CAVE installation) and networked worlds. The same codebase is used both for server and client which are just different configurations. RealXtend uses Ogre3d for rendering and Qt for UI and several other things. It has a mature built-in development and design environment and API, allowing the developers to implement required extensions for 3D-LIVE. RealXtend already contains a powerful avatar animation system and game state synchronization as a baseline. The open source nature of the platform, along with its advanced network synchronization technology, allowed 3D-LIVE for extending the platform's functionality in order to support synchronized 3D reconstruction rendering in real-time. Moreover, the well-defined user interface developing tools, which are also included, allowed to build easy to use and responsive UI frontends for the 3D-LIVE application.

The RealXtend game client mainly handles the visualization and input handling of the game application. The user is introduced to the jogging scenario by the main user interface which indicates proper ways to interact with the program. Even though all the large game assets are loaded from the server, all the user interface elements and gameplay logic is locally available. Two different modes were implemented in RealXtend Game Client: a) indoor user mode b) observer mode. Therefore, the user decides if his/her client acts as indoor runner application or as an observer who just observers other runners without interaction via clicking the appropriate button. The application has been designed so that all the initializations for avatar animation, connection to the server, downloading of all the required game assets and starting positions for each jogger are done automatically (R0-32). User body animations are captured when user decides to start the course on a treadmill.

When the connection and initialization is made to the game server, client application starts the gameplay scripts, which update the user interface with speed (R0-11) and distance travelled information, create weather queries and update the user position on the race score-table. Since collaborative running is emphasized, people are encouraged to talk to other runners and inform them about when to start running through the course. Indoor users can see other users animating realistically in real-time either through avatar representation, predefined animations (R0-5) or through full human 3D-reconstruction (R0-9) (See also sections 3.1.1, 3.1.3, 3.1.4 for technical details). Finally, when users have agreed to start the collaborative running and finished the course, every user receives a final qualitative analysis of their running performance, based on a set of rules calculated by an algorithm using the captured skeleton data as source of information (R0-12) (See section 3.2.2). Users can then compare their “quality” of running with other users through voice over IP (R0-25).

Suitability in 3D-LIVE: RealXtend can be setup both as server and client in one executable and all new features are synchronized properly across the server and the clients without separate testing. RealXtend is open source and free to use for everybody. A wide variety of C++ libraries are supported and fast user interface prototyping is made possible by using internal scripting language (javascript).

3D modelling of the scene

The jogging-scenario venue is located in Finland, in the city of Oulu. A nine block 3D-model of the venue was created accurately to immerse indoor users to the virtual Oulu scenery. The Blender3D modelling software was used to model the venue to resemble the real life counterpart. Blueprints of the city area were used with extensive photographing and height data to achieve the correct scale (R0-6). Final touches were made in the rendering engine to match the scale with the avatars. The venue contains city hall, green park and central meeting point of Oulu pedestrians, which was also used as a starting and finishing point of the jogging scenario run. With accurate mapping and realistic scale, indoor runners get better immersed in the twilight zone. An image of the virtual venue is shown in Figure 10 and a real life comparison in Figure 11.

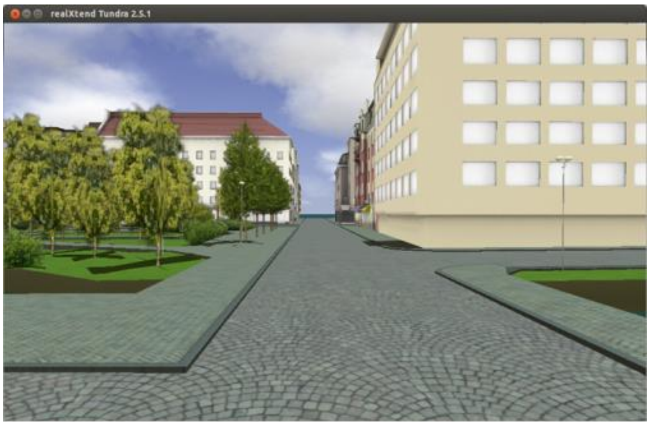


Figure 10: Image of the virtual venue.

Figure 11: Image of the real life venue

Interplay among input devices in 3D-LIVE

At the start of the scenario, where the avatar has been spawned on the virtual location at the start of the course, avatar controlling is very straightforward. For the indoor user, Microsoft Kinect(section 2.1.1) has a leading role in controlling the user's avatar. The application receives the Kinect skeleton data captured by 3D-LIVE Capturer (section 2.7) via the RabbitMQ Server (section 2.4). The skeleton data are then used to animate the user's avatar using the animation algorithm described in section 3.1.1. Moreover, the application also receives the user's speed estimation via RabbitMQ as it was calculated by the 3D-LIVE Capturer using the speed estimation algorithm described in section 3.2.1. Avatar then proceeds to run through predefined route at the virtual city of Oulu at the speeds reported by the speed estimation module while being animated via the skeleton data captured by the kinect. If any problematic situations occur throughout the course, backup controllers were implemented by using keyboard and mouse. WASD-buttons from the keyboard control the direction where avatar moves, spacebar makes the avatar jump over obstacles and pressing the right mouse button activates mouselook controller mode, which allows changing the direction that the avatar is facing. F-Button enables and disables the flight mode.

Controlling the outdoor jogger avatar can only be done by connecting the Android smartphone with 3D-LIVE application. GPS position is transmitted to the centralized service which is used by other realXtend clients to update the outdoor avatar location in the virtual world seen by indoor joggers.

User interface in 3D-LIVE

The indoor game client is designed to be handled with mouse and keyboard found in all desktop computers but interactions are kept minimal. The majority of the input required is at the start of the application and in the main menu. Main menu requires basic input of server address, port and protocol but also the address of the RabbitMQ Server (section 2.4) where Kinect data is read when captured. After these basic input fields are set, the user selects if he is going to join the jogging experiment as observer or as jogger. These options are shown in Figure 12 and Figure 13.

Figure 12: Main menu with address input fields

Figure 13 Main menu with capturer input fields.

At the start of the experiment, when users have entered the virtual world, user interface is a noninteractive data display. This is because the user has no chance to interact with the user interface when running on the treadmill.

Rendering and Visualization

3D-scene is rendered using open source Ogre3D rendering engine which is incorporated as core renderer in RealXtend platform. It can visualize lots of dynamic 3D graphical effects in high framerates and it can overlay the graphics with UI elements defined for the jogging scenario. The Graphical effects implemented include: dynamic day-night cycles, dynamic shadows according to sun position and volumetric clouds, which visualize cloud coverage retrieved from the weather data server. The Rain effect was implemented as a post-processing effect. Visualizing the weather conditions indoors as part of the 3D-scene immerse users better to the scenario, than reading all the data from 2D user interface elements. Some of the weather data are displayed on the UI if the user is interested in them. Data elements in the user interface include: weather data from the actual venue, distance travelled, user's position and speed. User interface elements from the actual racing conditions are shown in Figure 14.

Figure 14: User interface showing data for user.

The rendering algorithms implemented in the RealXtend indoor game client are able to render and animate pre-created avatar assets as well as render time-varying textured human 3D-reconstructed meshes. Indoor users can see their own avatar representation performing accurate animations of their movements from a 3rd person view. This is accomplished with the aid of 3D-LIVE Capturer (section 2.7), which streams user skeleton data, 3D reconstructions and activity evaluation results to the network via RabbitMQ. The users' competitor's avatars are also rendered inside the virtual environment. For the competitor outdoor jogger this corresponds to an avatar that performs predefined animations while for the competitor indoor jogger it may be either an avatar that closely resembles the movements of the actual jogger or a 3D-reconstruction that realistically depicts him/her.

Pathfinding (jogging route calculation)

In the jogging scenario Android GPS location data is already heavily filtered to acquire the most precise location currently available when queried from the Android system, therefore no additional GPS filtering was done client side to improve location accuracy. Setting the avatar on the virtual world on the other hand was not directly matched with a simple translation from GPS coordinates to virtual world XZ-coordinates. Location reported by android system was not always aligned with the roads, therefore simple closest match algorithm was implemented in order to place avatar on top of the road in virtual world. The Closest-match algorithm is visualized in Figure 15.

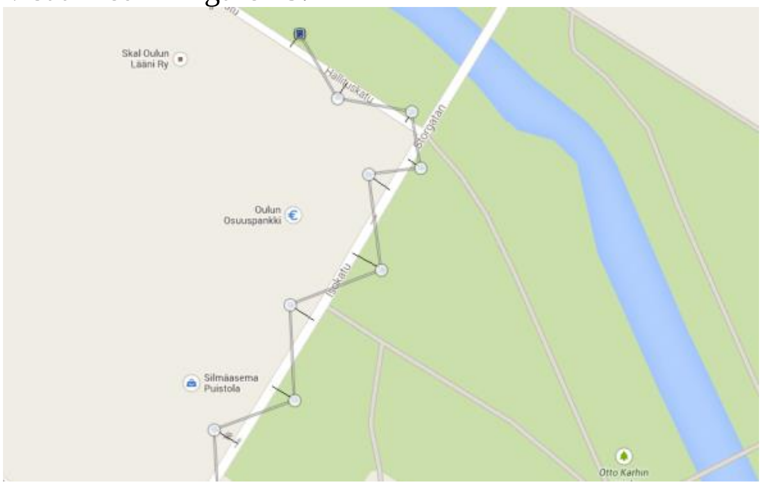


Figure 15: Closest-match location algorithm visualized.

Pathfinding for indoor user is much simpler due to absence of GPS inaccuracies. Indoor joggers followed the predefined path created by checkpoints therefore indoor jogger position was changed solely according to speed parameter captured by 3D-LIVE Capturer (section 2.7). Following strictly the predefined path from one checkpoint to another, initially generated large amounts of collisions between avatars, therefore we introduced little error to the path, which is explained in more detail in the next section.

Collision avoidance

Collision avoidance was used in jogging scenario to make jogger avatars run on less crowded paths from checkpoint to checkpoint. This was accomplished by introducing little bit of randomness to the chosen path where avatars were destined to run through the course. Course was generated by placing a checkpoint throughout the course and avatars found their way through the course by sequentially running through these checkpoints. Changing the actual checkpoints to so called checkareas, where avatars did not focus on one point but more like focused to find their own way to the random spot in this specific checkarea, made avatars to differ in their paths just enough to minimize collision count with each other. This fairly easy method is computationally invisible and caused no lag, nor slowdowns for the rest of the system. Route selection system with collision avoidance using check areas is shown in Figure 16.

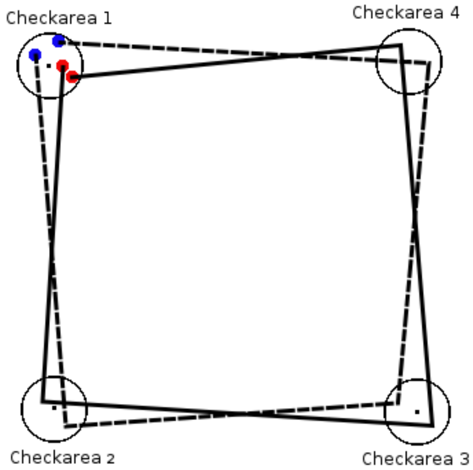


Figure 16: Route selection for two avatars using checkareas.

Deployment - Visualization

For the deployment of the indoor game client of the jogging scenario, apart from typical displays that were used in the Low-Cost indoor cases, 3D-LIVE also utilised:

- **CAVE system**

The RealXtend indoor game client was deployed in Oulu University labs of Center of Internet Excellence (CIE) cave. Cave was created using three 1280x800 video projectors for wide field of view, and virtualized as one big display. This setup allowed user to immerse to the virtual world with close to 180 degree horizontal field of vision. Picture of the CIE cave is shown in Figure 17.

Figure 17: CIE cave used in jogging scenario.

- **Wide displays**

The 3D-LIVE jogging scenario has also been deployed to normal wide screen.

2.8.2 *Unity 3D Game Client*

The indoor game client applications for golfing and skiing scenarios have been created using Unity3D, a cross-platform game engine. The Unity3D indoor game client has been developed as a single application that is able to support both scenarios via configuration. The indoor game client's features can be divided in two main parts: pre-game features and in-game features. Pregame features include the settings menu, user instructions and pregame room. In the settings menu, the user can set his profile data, select the sensory devices that will be used, and enter the IP address of the server he wants to join. The user can read some instructions before to join the pregame room, where he will be waiting for other players to join the game, until they are all there to start. In game features include everything that makes the game. The user will see a virtual representation of each player in the virtual environment. Through the user interface, he can see additional information about the game (e.g.: the length of his last shot in the golf scenario), environmental data or the positioning of other players. The application will use the sensors that have been chosen in the settings menu to interact with the virtual environment. The rendering will be done in accordance with the choices made in the settings menu. Finally, the indoor application communicates with Unity3D server in order to synchronize all game data (player positions, game state, game phases) and with the RabbitMQ server for everything related to external modules (see section 2.4).

Audio

In the indoor game client application, there are two uses of audio: player communication and ambient sounds.

For player communications, the indoor users use a separate application as described in sections 2.6 and 2.10.

Additionally, ambient sounds are also implemented in order to enhance the immersion. For the ski scenario, those sounds are essential as the users are using immersive display device. Those devices can bring some cyber sickness. Generate appropriate stimulus for the other sense tend to decrease the sickness.

Interplay among input devices in 3D-LIVE

As already mentioned in the beginning of the section, the indoor users can use various devices to interplay. Those controls can be chosen in the setting menu of the indoor game client.

- Ski scenario :

For the skiing scenario, indoor users can interplay by controlling the direction and the speed of their avatar. Four different ways of avatar direction control were developed:

- When the user did not select any device, she will control her avatar's direction using the keyboard. By pressing A or D the player can respectively make a turn to the left or to the right.
- If no device is selected and a game controller is plugged (such as an Xbox controller) she will be able to use the joystick to turn left or right.
- The Wii Board can be used by selecting it in the settings menu. The controller must be connected by Bluetooth to the controller, and the GlovePIE middleware running with a Wii Board configuration. By switching her weight to the left or right, she can respectively make a turn to the left or to the right.
- The Pro Ski Simulator can be used by selecting it in the settings menu. The EXLs sensor must be connected by Bluetooth to the controller, and the EXLs_datasender middleware running.

The avatar's speed control follows a solely physics based implementation influenced by metrics and data captured by the Microsoft Kinect (section 2.1.1). Thereby, the speed can be controlled only if a kinect is used. The first way to control the speed of the avatar is by doing the movement of pushing on the ski poles. As the avatar is animated, the implementation of the internal physics engine detects when the poles are hitting the ground. When it does, a force is added to the avatar, making it go faster. The second way is by using the Activity Recognition (section 3.2) feature of the 3D-LIVE Capturer (section 2.7). It calculates the bending ratio of the user. As the skier controller behaviour is physic-based, when the user bends, it decreases the frontal area of the skier[9] and therefore also the aerodynamic drag force. This results into accelerating the skier and the maximum speed that can be achieved is increased.

- Golfing scenario:

For the golfing scenario, indoor users can interplay by hitting the ball and by making game related choices in the indoor game client application.

Indoor users can use two types of a golf club: Iron 6 and putter. Both of them are equipped with an IMU sensor (section 2.1.1) of which the data are used to control a shot during the strike phase. The strike phase can be divided in three parts. First the player can target the direction where he wants to shot by rotating the club around the shaft axis. Then he can do free shots to train until he stops moving when he is ready. Finally he can perform his "real" shot in which the sensor will measure the speed of the club head at impact in order to simulate the shot in the virtual world.

During the game, indoor and outdoor players need to interact at several moments: when they are ready to play, when they must choose the best between their two golf balls and when their ball is out of bounds or lost. This is accomplished via on-screen buttons. An indoor user can choose to use a support screen or not. If she uses it, the main screen will only display the 3D environment and the support screen will display interactive GUI (e.g.: minimap or buttons). If he didn't choose to use a support screen, both 3D environment and interactive GUI will be displayed on his single wide screen.

Values found from Sighard F.Hoerner, *Fluid-Dynamic Drag : practical information on Aerodynamic drag and hydrodynamic resistance*, 1965[9]

3D modeling the scene:

Ski

a) Low poly mesh of the slope

In order to make consistent the wide area environment in the virtual scene, we had two opportunities: integrating a mountain sky box in the scene or create a low resolution textured mesh of the mountains surrounding our slope. Using the first solution, it would have increased the performances in-game, but unfortunately the skybox of the real mountain was very hard to acquire, and it would have been impossible to create consistent weather effects over a skybox. Consequently, a 3D model has been created thanks to Google Earth and Sketchup's low resolution model extraction.

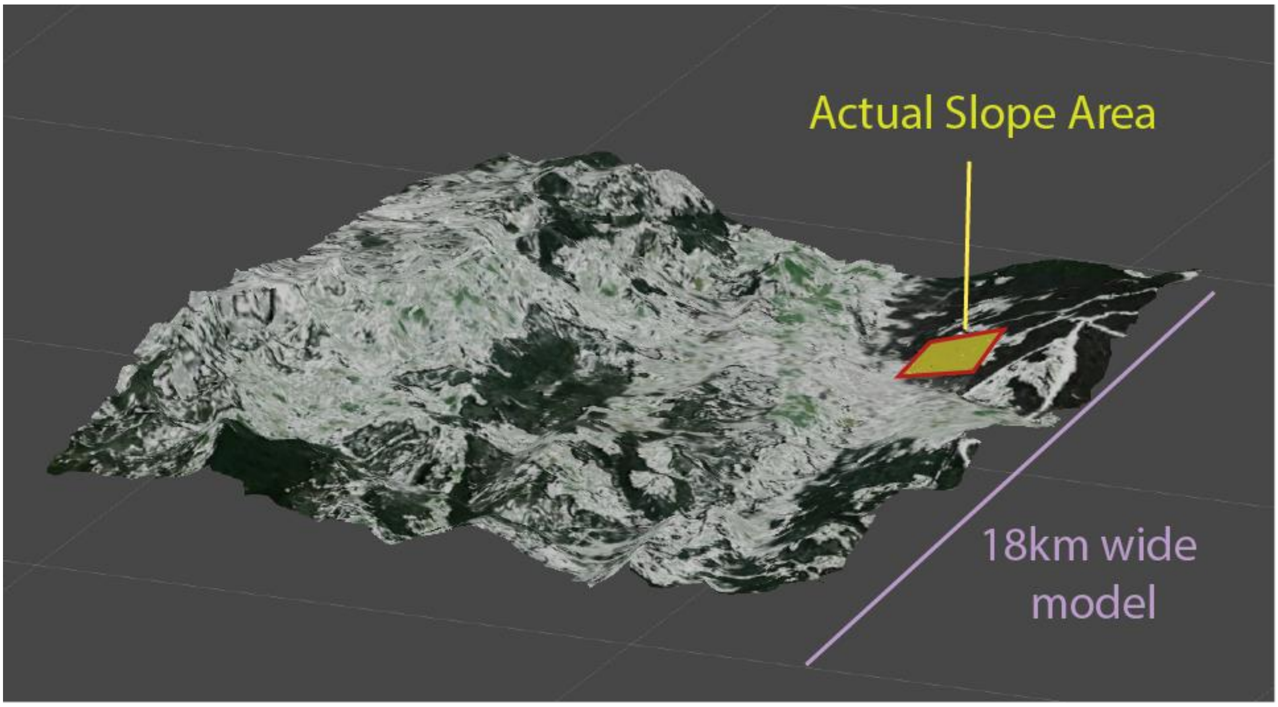


Figure 18: Wide area model and actual slope representation integrated

b) High poly mesh of the slope.

In order to accurately reproduce the Schladming Planai I black run, we secured use of 1x1m resolution 3D geometry of the mountain range from and with the agreement of the governmental authority of Steirmark[10]. In its raw form, the data we were provided was too large and not in the correct format to use directly within the Unity3D.

[10] http://www.verwaltung.steiermark.at/cms/ziel/74837988/DE

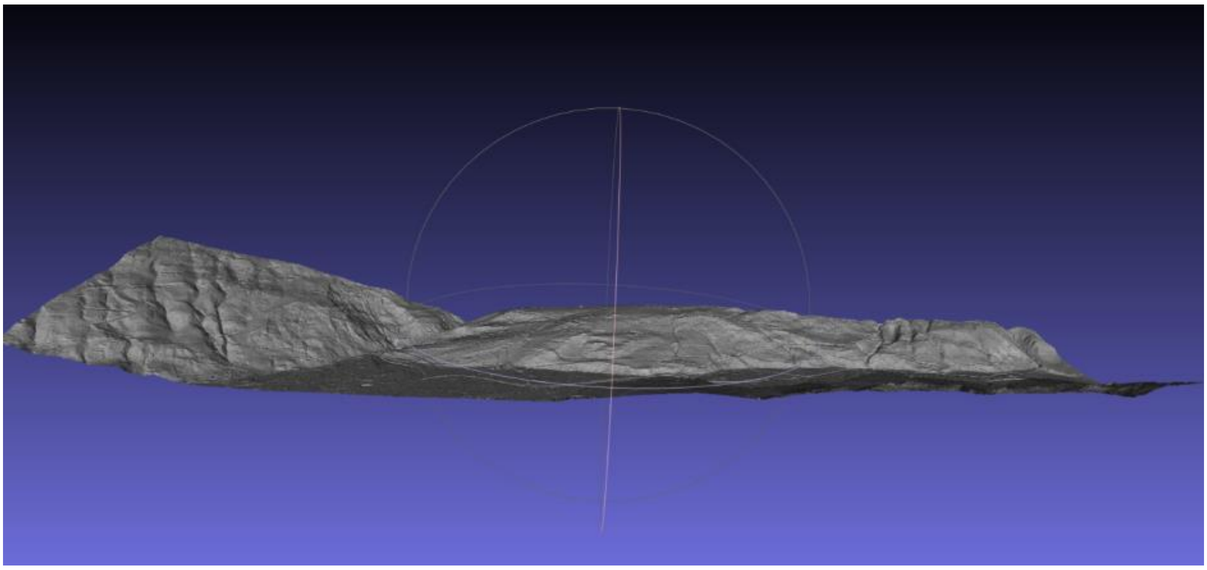


Figure 19: Editing Schladming Planai I 3D Geometry with MeshLab

A pre-processing phase that included clipping and point cloud data conversion was therefore required. We used MeshLab (http://meshlab.sourceforge.net/) to clip the data set (6.7 million points initially) and then compute normal and reconstruction polygons from the point cloud.

Then, using Blender graphics software (http://www.blender.org/), a 1024x1024 height-map of the model was created. To do this, we put an orthographic camera on the top of the model with a resolution of 1024x1024, and set its orthographic scale to fit the model's limits. A shadeless material with a texture was used and set its type to blend. Its colour band was set from black to white and completely opaque. All mapping axis were set to "Z" and the projection was set to "flat". We changed the colour management settings to "Linear" in order to have a linear relation between height and greyscale. The final step was to set the output format to TIFF.

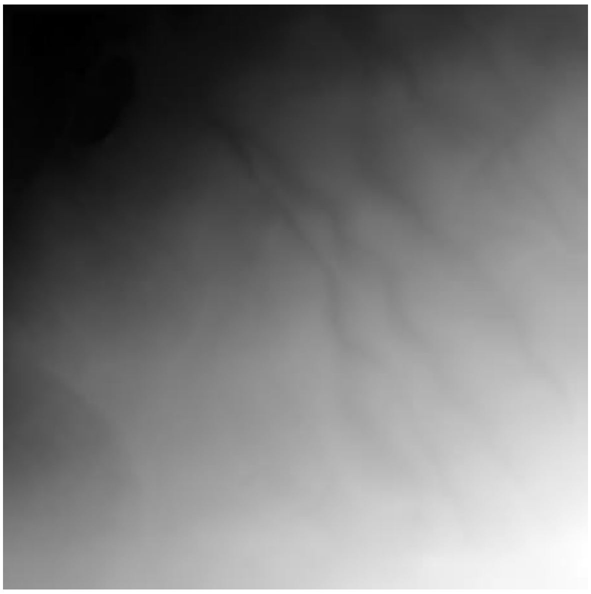


Figure 20: Height map of the high poly model of the slope

Then we used the terrain creation built-in Unity3D module in order to import the image and then get a good quality terrain. As the model is 1300 meters wide, 1 pixel represents 1.27 meters which fits to the accuracy of the measurement of the slope. Unity3D provides some tools in its terrain module, which allowed erasing some defects (mostly local irrelevant heights) by smoothing. Trees were also manually textured and added using those tools. Then we added the snow effect thanks to the “ATS Snow Suite” free package, which affects both terrains and 3D models.

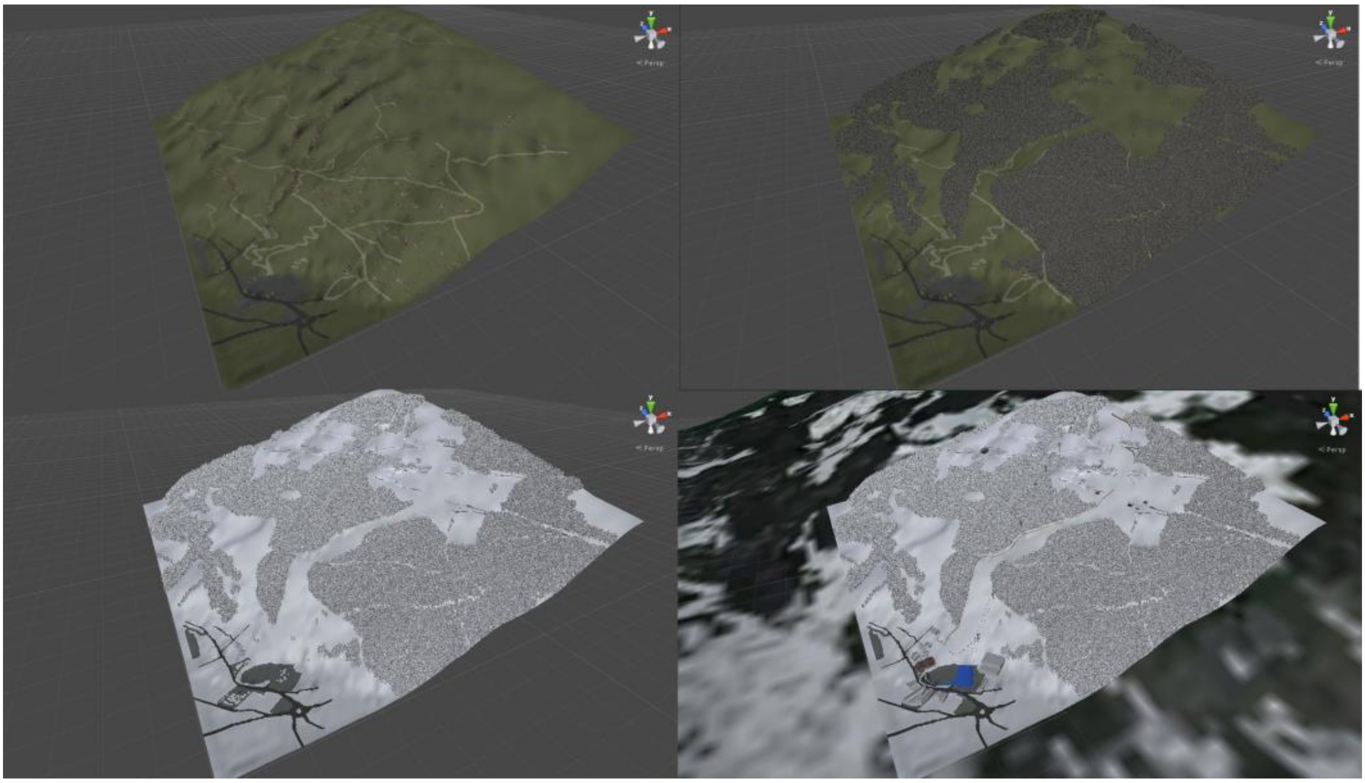


Figure 21: From top left to bottom right: Texturing – Adding trees – Adding snow – Full model

The final step was the addition of several objects in order to enrich the environment. Indeed, even if the natural environment was real-based, it was not sufficient to provide a good user experience. Users need visual information that will make them feel immersed in the virtual world and in the game. For this purpose, three types of elements were created:

- Real elements: By using Google Earth and on the site pictures we were able to design buildings or other elements that are present along the ski course: houses, snow making machine, telecabin, etc.
- Game play related: Checkpoints, safety net to define slope limits and gates.
- Promotion: banners were also added with the logos of partners. It was considered important to provide visibility for all participants of the project during public presentations.

Figure 22: Houses along the ski course

Golf:

The golf scenario took place on the golf of Laval. Course number one is used for the experiments, as it is an easy one close to the clubhouse.

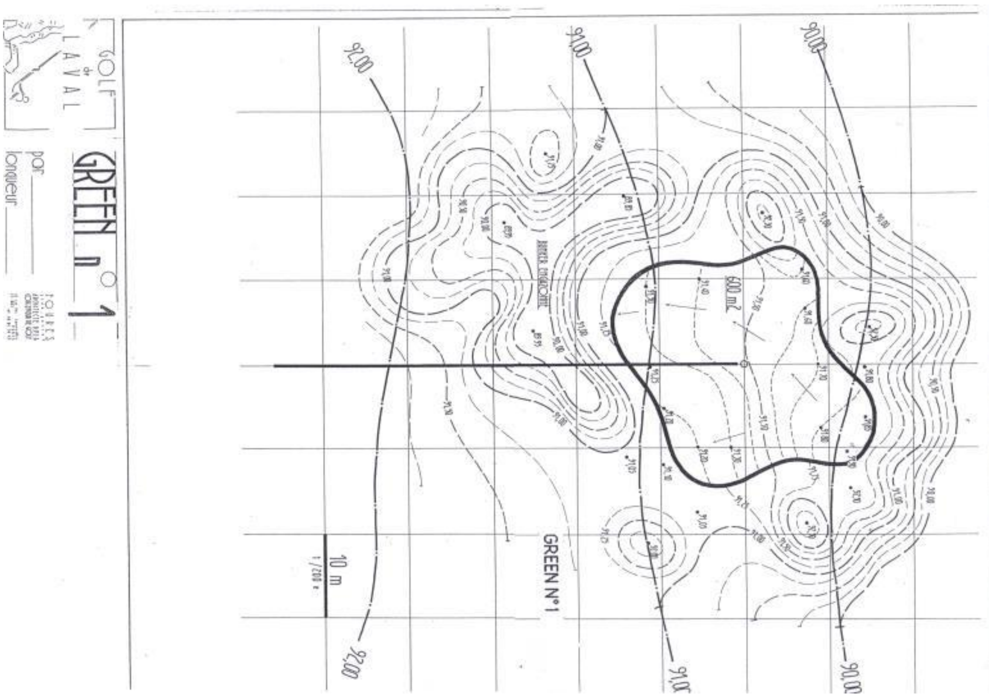


Figure 23: Topographical survey of the green in Laval golf course

Using Google SketchUp and an accurate topographical survey of the green provided by the staff of the golf of Laval, we created a first model. Despite the efforts, the model wasn't clear enough.

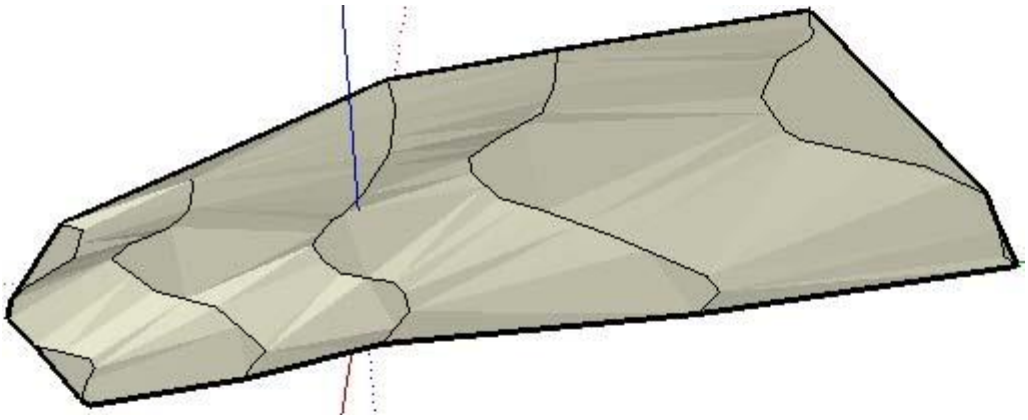


Figure 24: From topographical survey to 3D Model using Google Sketchup

Some faces were not set in a proper way, which could create inconsistent behaviour when it comes to physics simulation. Those errors were removed manually using Blender.

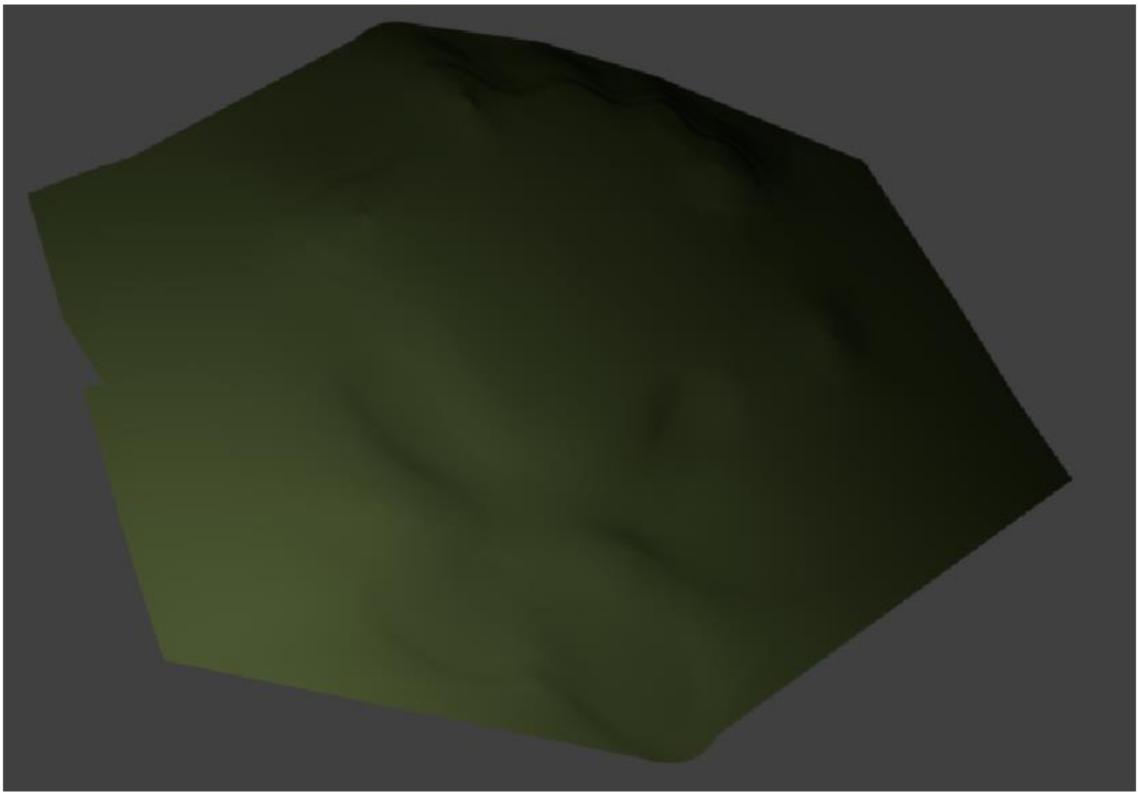


Figure 25: Final model of the green.

Modelling of the fairway was more complicated because of only one available bad quality image of the course, without any data (scale, heights).

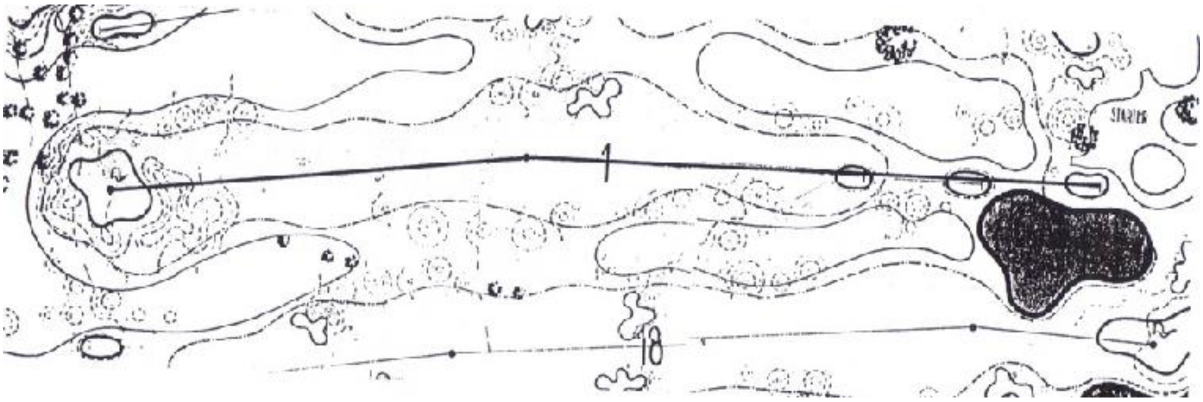


Figure 26: Approximates contour level lines of the course

The fairway was modelled using Blender, trying to have as few polygons as possible. Indeed, this is a large area where a highly accurate model is not required for simulation. On the contrary, the green model should be pretty accurate, because the putting physics require higher precision to get a good simulation.

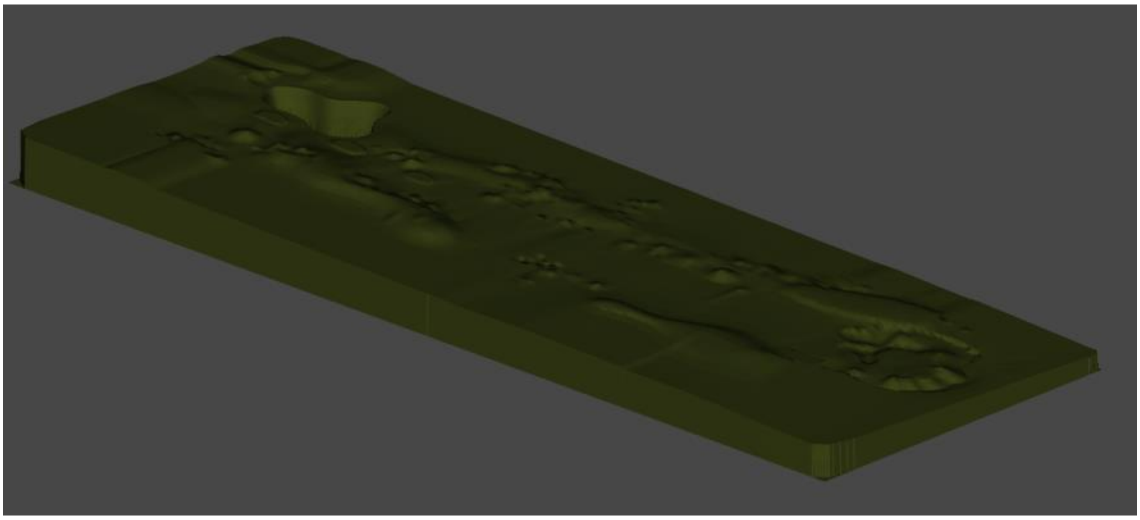


Figure 27: Final model of the course n°1.

Physics simulation

Unity3D has a built-in physics engine that calculates several things: collision, friction, forces, velocity and others. Physics is calculated in a separate thread and updated at a fixed rate. This allows a precise simulation but impose not overloading the thread. On each object that we want to have physics simulation, we add a "rigidbody" Unity3D component, which allows controlling its speed or adding forces. In our applications, there are two main uses of the physic engine: contact with the ground and aerodynamics. The aim is not having perfect realistic behaviour, as the number of parameters to take into account is too high. Instead, a conceiving simulation is enough. The only requirement is that the users believe it is real and feel engaged in the game as if they were really playing on the course. Empirical measurements allowed us to refine our algorithms to get a simulation as close as possible to reality. As a result, during Live 3, none of our golfers noticed an inconsistent behaviour of their virtual strikes, while for ski the users reported that the runs were smooth and enjoyable.

Skier (Skiing):

For each avatar a "controller" script was developed that performs the physics simulation, in order to provide a control as realistic as possible.

Wheel collider (Skiing):

The simulation of the contact between skis and the snow is made using a Unity3D component called "WheelCollider". It is normally used in driving games to simulate the behaviour of cars. It allows to easily simulate sliding. A full realistic simulation was impossible because the hardware used does not measure how the player is putting her weight on her feet. Indeed, on the pro ski simulator this data is not measured and on the Wii Balance Board it would have no meaning as users can't reproduce the ski movements. WheelColliders were set up experimentally in order to get a behaviour close to reality while remaining easy to handle for a gaming purpose.

Speed estimation (Skiing):

The speed of the skier (when she doesn't use her poles) is the result of the gravity, the friction between the skis and the snow and friction between the air and the skier. Gravity will boost the skier while both frictions will slow him down. The equations describing the forces applied to the skier are given below:

Snow/Ski friction intensity: $F_{friction} = \mu \times m \times V_{skier}$

Aerodynamic friction intensity : $F_{aero} = \frac{1}{2} A \times C_d \times \rho \times V_{skier}{}^2$

where:

- μ is the kinetic coefficient of friction between snow and waxed ski. Its value is about 0.1[11].
- m is the mass of the skier with his equipment
- V_{skier} is the speed of the skier.
- A x Cd is the frontal area times drag coefficient. According to already mentioned "Fluid Dynamic Drag" study, this product goes from 0.84 for a person standing up and 0.46 for the tuck position.

The "controller" script implements speed estimation by using the abovementioned physics equations. The tests performed on the slope showed that the simulation is relevant: an indoor user can perform a downhill on our 1.2km slope in 1 minute 7 seconds and a professional skier on the slope achieves a 1 minute and 10 seconds time. Considering that the user in the virtual world has better conditions, we can confirm that the simulation is accurate.

Golf ball (Golfing):

For golf, an appropriate script was developed to handle strike simulation and ball physics taking into account the following concepts from physics:

Bounciness

In order to handle the contact between the ground and the ball, the Unity3D components "rigidbody" and "physics material" were employed. Physics material permits to control the bounciness of the ball: 80% of the kinetic energy of the ball is absorbed when it touches the ground. The "rigidbody" allows setting the

[11] Raymond A. Serway, & Jerry S. Faughn. *College Physics Sixth Edition*. Pacific Grove: Thomson Learning Inc, 2003: 102.

friction of the ball when rolling on the grass of the fairway or the green.

Flight

The golf ball's flight is not a simple ballistic trajectory. Due to the shape of the golf ball and its movements during the flight, the trajectory is more complex.

When a golf ball is hit, the speed of the ball, the shot angle and speed of rotation are determined. Those parameters influence the ball's trajectory as well as the behaviour of the ball when it falls to the ground. The first phenomenon that applies is the drag and lift. Drag is a force opposite to the ball's motion while the lift force acts perpendicular to the path. They satisfy the following equations:

$$F_{drag} = -\frac{1}{2}\rho_{air} \times S \times C_d \times V^2$$

$$F_{lift} = -\frac{1}{2}\rho_{air} \times S \times C_l \times V^2$$

Where S is the reference area (projected area), C_d and C_l are the coefficients of drag and lift, ρ_{air} the density of air and V is the velocity of the ball. As the golf ball is a sphere, the lift is not due to the geometry of the ball (as it would be the case with an airplane wing), but to the Magnus effect.

Obviously, the ball undergoes the laws of gravity as well. When a spinning ball moves through the air, it will change the speed of the airflow around it asymmetrically. On the one side, it accelerates the air by pushing it away resulting into a local pressure decrease. On the other side, it is opposed to the movement of air resulting into local pressure decrease. Consequently, as the pressure becomes higher on the one side of the ball when compared to the other, a force called lift appears. The higher the speed of rotation, the more significant this force becomes.

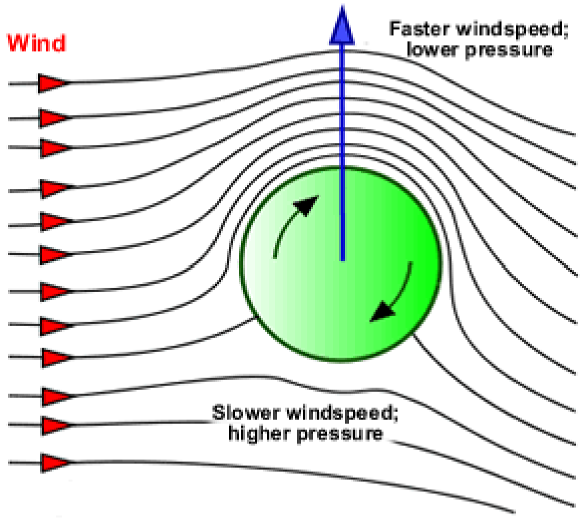


Figure 28: Magnus effect representation

The direction of the lift force depends on the orientation of the ball's rotation. For almost all the shots, the rotation is from the front to the rear (due to the inclination of the club) generating an upward effort that will blow up the ball. If the ball also rotates along a vertical axis, the trajectory of the ball is changed to one of the sides: we then say that the ball has an effect (e.g.: hook, slice...).

This is where dimples on golf balls come into account. Those dimples create micro-disruption generating a turbulent flow. It increases a bit drag force and significantly lift forces. So even if the ball is slowed, it is "flying" more and thus goes further thanks to the dimples.

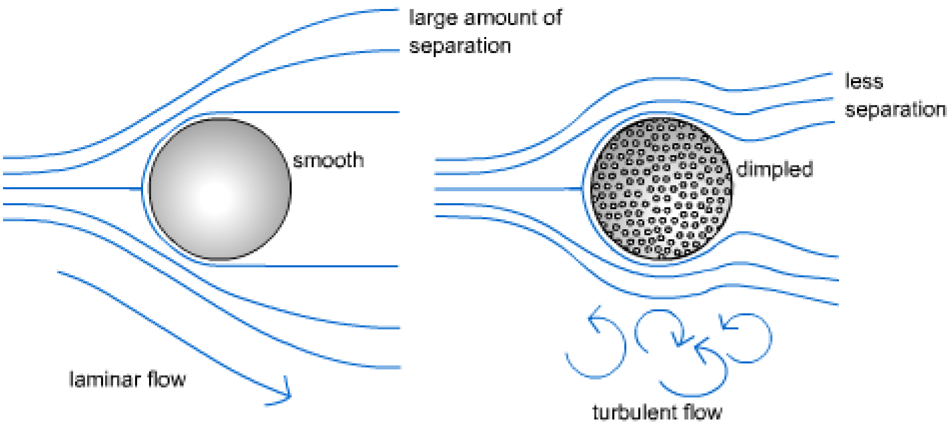


Figure 29: Effect of dimples on the air flow[12]

The complex part of the simulation is to determine the equation of both drag and lift coefficients. Indeed, there is no theoretical equation, but only laws based on experimental measurement. From various studies[13], we pulled out models giving the drag and lift coefficients:

$$C_d = 0.23 + 0.35W$$

$$C_l = 0.4W^{0.4}$$

Where W is the rotation rate: $W = \frac{2\pi\omega r}{V_{ball}}$

[12] http://guernseydonkey.com/?p=7438

[13] P.W.Bearman, J.K.Harvey, Imperial College of Science and Technology, Canada – *Golf Ball Aerodynamics*, The Aeronautical Quaterly, 1976 ; K.Maehara, K.Ihara, H.Shimosaka, M.Inoue, A.Kasashima, Bridgestone Sports, Japan - *Golf Ball*, United States Patent n°05916044, 1999

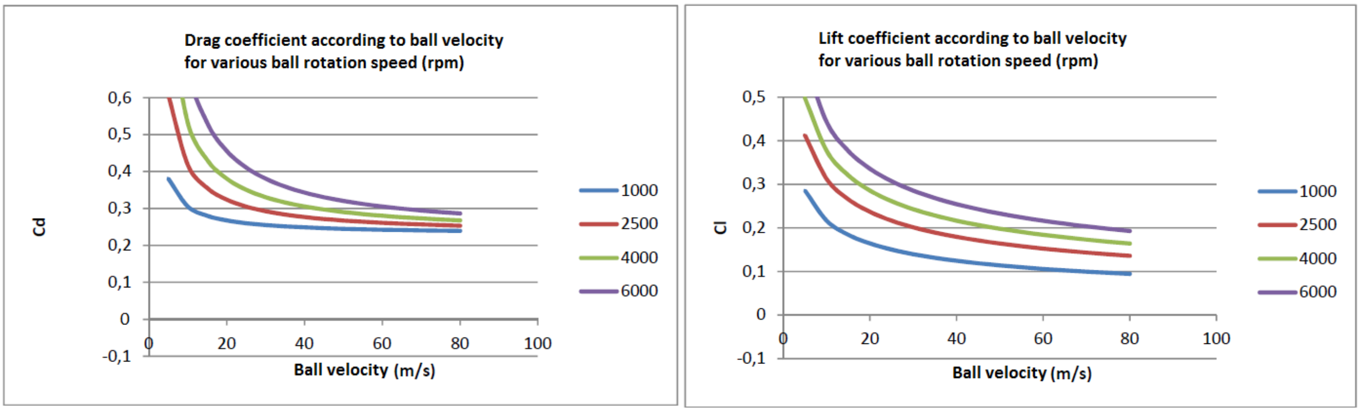


Figure 30: Drag and lift coefficient models

In the application the golf ball's flight is simulated by applying all forces to the ball regarding its velocity and rotation.

User Interface

As indicated above, the game application is divided in two main parts: pre-game and in-game.

The pre-game part contains three things: settings menu, instructions and the "GO" button that start the Game.

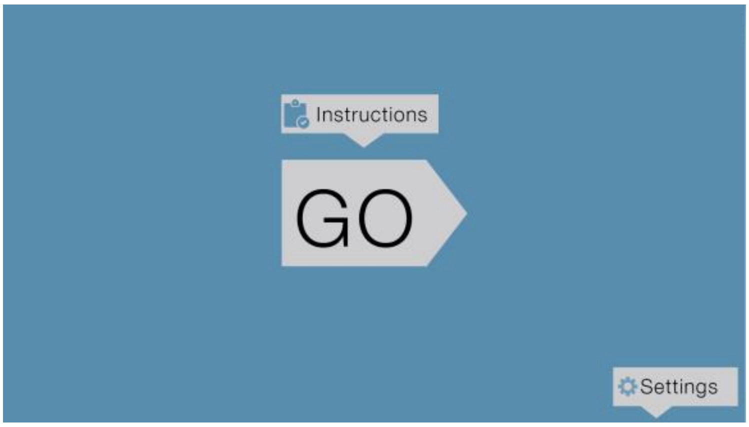


Figure 31: Start screen (Home) of the Unity3D indoor game client application

By clicking on the "Settings" button, the user gains access to all the available application settings that configure the underlying mechanisms developed in the indoor game client application. Most of the settings are the same between both scenarios, except for scenario-dependent devices. The navigation mechanism implemented inside the settings menu is kept as user friendly as possible. The "Back" button brings the user back to the previous window, while the "Home" button redirects the user to the start screen.

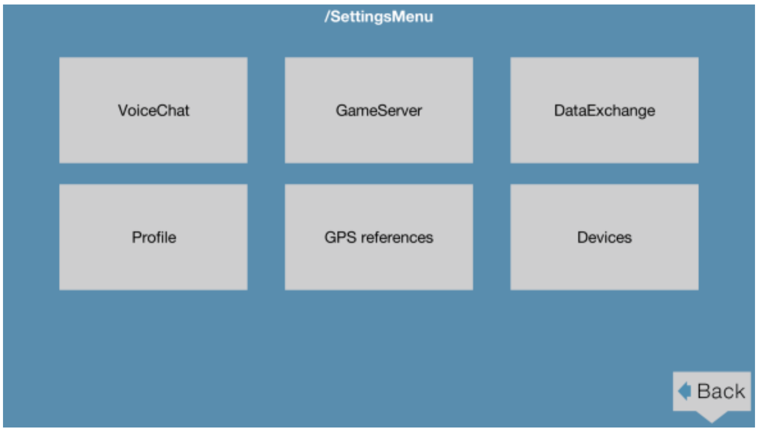


Figure 32: The settings menu of the Unity3D indoor game client application

The devices to be used in-game are chosen from the “Devices” menu. The available devices depend on the profile selection being either “high end” or “low end” which can be set from the “Profile” menu. All the buttons are designed to be easily readable and clickable, as those screens are also the same on tablets and smartphones.

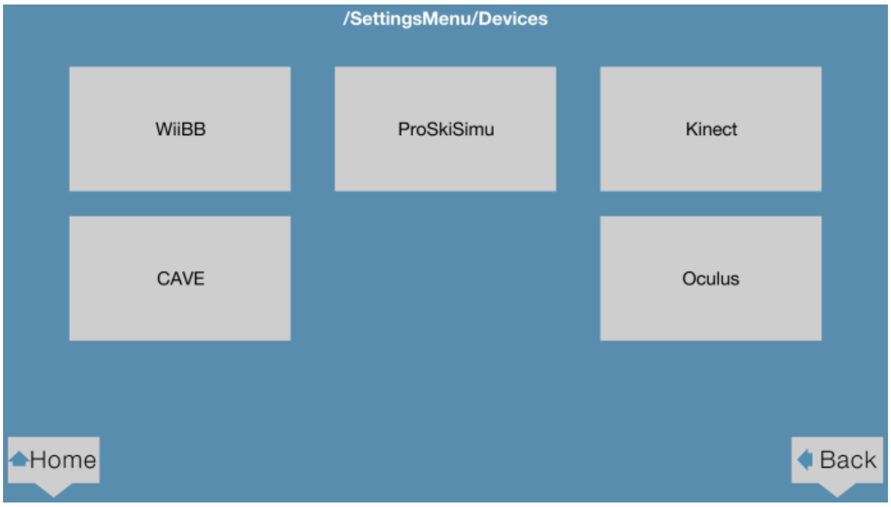


Figure 33: Devices menu on Unity3D Indoor game client application for a High-End user

A device is enabled/disabled by turning it on or off as shown in Figure 34.

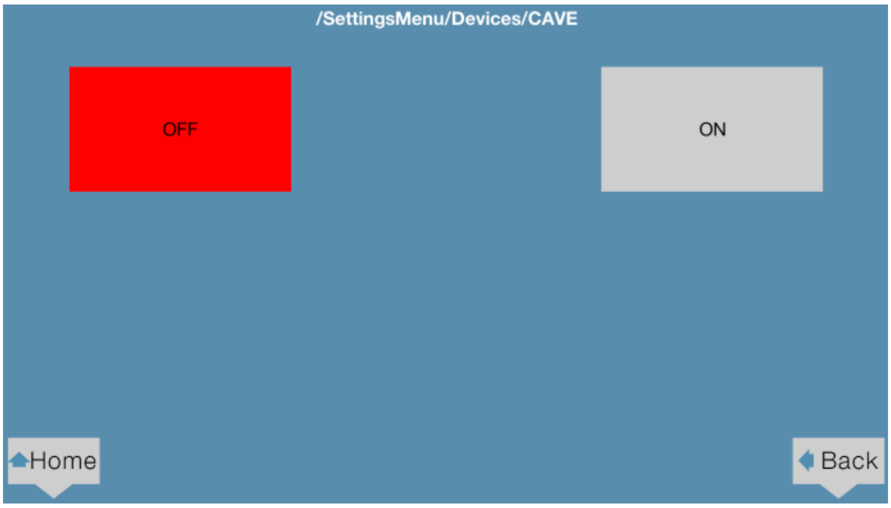


Figure 34: Turning Cave Device Support ON or OFF in Unity3D indoor game client application

When clicking the "GO" button in the home page, the player joins the game room, where (s)he will be waiting other players.

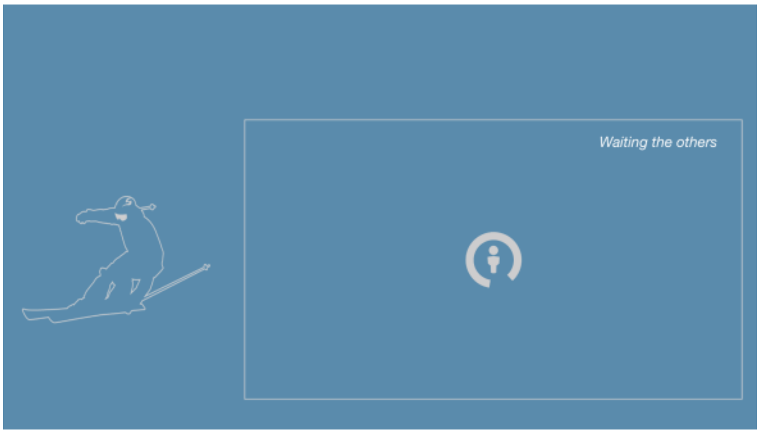


Figure 35: Game room of the Unity3D Indoor game client application

When all players have joined the game room, the server will load the game level, reaching the second part of the application which contains in-game features. As each in-game user interface is different regarding the scenario, they are described separately below.

Ski:

As indoor users are on a ski controller (Wii Balance Board or Pro Ski Simulator), there is no interactive elements on the user interface. During game phases where players can just talk, weather information acquired from the ERS (section 2.5) is displayed. Moreover, the player's name is displayed on top of each avatar.

Figure 36: Unity3D ski indoor game client application with weather information

During the run, all the players' path trajectories along with timestamps are saved in-memory. Upon finishing the run the see the results of the race, with their rank position, nationality flag and time to finish the track.

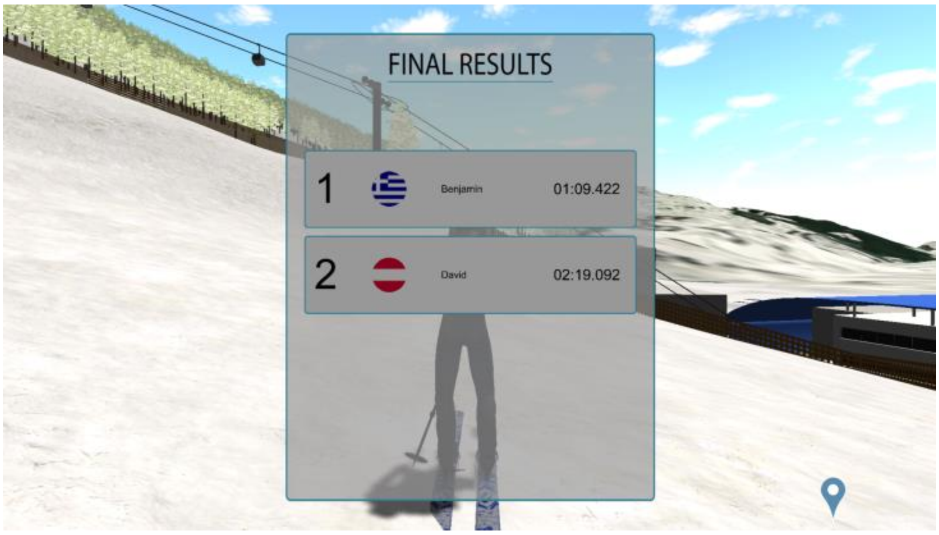


Figure 37: Unity3D ski indoor game client application with ranking information

After the ranking being displayed, a 2D replay of the race is shown based on the recorded data gathered during the race. Each player is represented by the flag of his nationality.

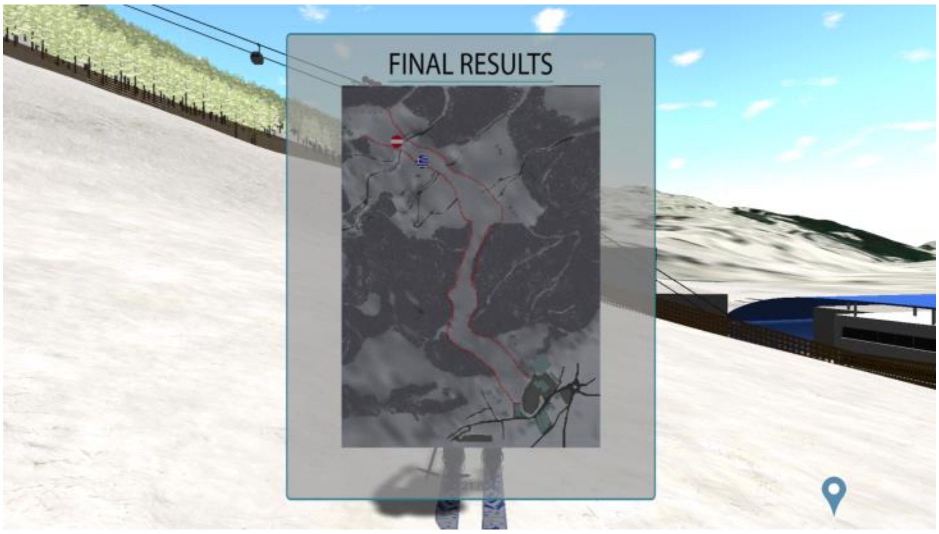


Figure 38: Unity3D ski Indoor game client showing a replay of the race

During gameplay, in order to display short information or instructions (e.g.: countdown), pop-up messages are displayed for a short time. During the race, the player can also see various pieces of information: speed, position, compass and delay when passing checkpoints.

Figure 39: Unity3D ski indoor game client application with countdown pop-up message and race UI

Since for the ski scenario a player can use various types of displays, we created a customizable User Interface compatible with all types of displays: Oculus DK1 or 2, CAVE, SAS3+ (by CLARTE) and standard monitors. This module allows us to quickly adapt the UI to a new device by changing positions, scales, and the camera used for display. To handle this adaptive interface, an architecture of basic 3D UI elements has been defined. Each basic 3D element is attached to a specific camera or game object at a predefined distance. Thanks to that process a UI element can be displayed on any device. The parameters to set for those elements are: Texture, Position, Rotation, Object Parent, Text and they can be updated dynamically. When we deploy on different rendering devices, we just instantiate the same elements, but change the initial position or parent object. The fact our User Interface is made with 3D elements strengthen the feeling of immersion in the application as we feel the UI objects in the virtual world.

Figure 40: Unity3D ski indoor game client's customized User Interface: CAVE (left), simple monitor (top right), Oculus Rift (bottom right)

Golf:

In the golf scenario, an indoor user can optionally use a support screen. This means that all 2D user interface items (text, map, buttons) will be displayed on the support sreen, and the main screen will display all the 3D content. In the case that the support screen is not used, both 2D and 3D content is displayed on the unique monitor.

For the indoor player the visualization of the following features was developed:

- The score with scramble choices
- History of previous events like shots or ball choices
- User names on top of avatars

- Pop-up messages that give instructions and information
- The position of the hole and the balls thanks to 3D indicators
- The club the user is playing with

Figure 41: Unity3D golf indoor game client

When a player strikes the ball, the ball creates a trail that permits to view its trajectory based on the physics described in section 0. All the players can see it, so that even the remote players know where their mates shot.

Figure 42: Unity3D indoor game client: Ball trajectory after a ball shot

When the player’s ball has stopped, the map pops for all users, showing where the ball is on the course as well as the remaining distance of the ball to the hole. The player finishes her turn by pressing the button

"Next".

Figure 43: Unity3D golf indoor game client: choosing the ball to start the next shot

When both players have finished their turn, they are asked to choose the ball, from which they will start the next shot. They see the map with each ball's position and remaining distance to the hole. They ought to click on the same button to start the next shot. If a player shot out of bounds, the other ball is automatically chosen, and if both are, the players start from the previous shot location.

Figure 44: Unity3D golf indoor game client: choosing the ball to start the next shot

When a player puts the ball in the hole, the game is finished. A final screen shows all important data relative to their performance as well as a mini map showing the trajectory of the scramble ball.

Figure 45: Unity3D golf indoor game client: Final Results

Suitability in 3D-LIVE: Unity3D is a powerful engine that has efficient built-in tools. The physics engine enables accurate simulations, the rendering engine is reliable and the networking module is easy to use and fit real-time conditions. Implementing a custom feature is also simple if needed. And all of those capabilities are available regardless of the platform.

2.9 3D-LIVE Outdoor Game Client

2.9.1 *Outdoor Game Client with Java and Eclipse*

For the outdoor jogging scenario a custom Android application was developed and deployed. The application was kept simple and the main focus was on reliability of the data transmission from the smartphone to the indoor clients. The outdoor jogger game client application's user interface has a basic interface for the user to see the route and his/her position on the map. The GPS data is transmitted to the indoor game clients. The corresponding outdoor jogger's avatar is then appropriately placed in the virtual world according to the algorithm described in 0 in order to be seen by the indoor joggers. The outdoor game client application only has one large button on the main view which enables the GPS location reporting and activates the map shown on the UI. Figure 46 shows the main elements of the outdoor application's user interface.

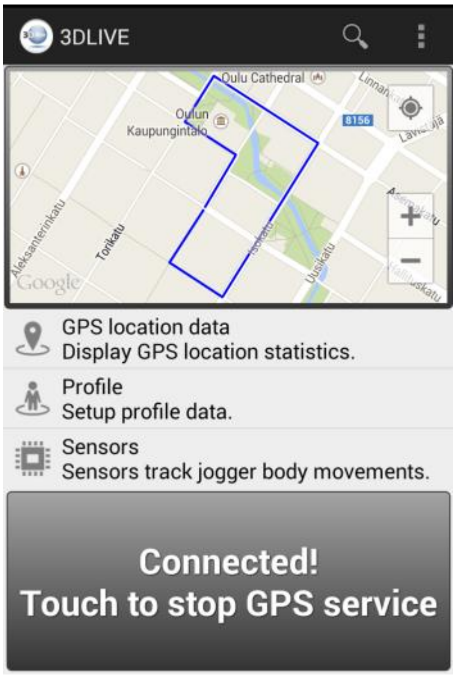


Figure 46: Mobile application used in jogging scenario.

2.9.2 *Outdoor Game Client with Unity3D*

Exploiting Unity3D's cross platform capabilities, the mobile outdoor game client applications for golfing and skiing were created with it, also following the same structure as the respective indoor game clients. Similarly, the outdoor game client for golfing and skiing is made of two parts: pre-game and in-game. As the pre-game features are the same with the indoor game clients, we will only focus on the in-game features (see section 2.8.2 for Unity3D indoor game client details) of the outdoor game client.

Audio

The audio system is only used for communication, using the separate Mumble application connected to the Mumble Server (see section 2.10).

Interplay among input devices in 3D-LIVE

Mobile applications are using their internal GPS with high accuracy mode to send the user's position to the server. Then, the outdoor avatar is positioned at the corresponding location in the virtual world. In the ski scenario, the outdoor avatar's position (and therefore the GPS coordinates) is used to trigger game phase changes, such as the end of the player's meeting before the race. For the golfing scenario, the outdoor users use a golf club with an EXL-S1 IMU on (prototype version of the EXL-S3, see section 2.1.1) to acquire motion data. This is working the same way as the indoor one (see section 0).

As the outdoor ski game client application is connected to a HUD with an on-board Android system, it receives inputs made on the wrist wireless controller from the HUD application. This is used to start the race when the players are ready.

User Interface

Ski

The outdoor application for the ski scenario doesn't have a user interface during the game. Indeed, the skier is using a HUD in which the user interface is displayed. The main duty of the outdoor game client application is to handle features such as networking or GPS location. The only visible information on the phone is the replay of the race. Implementing the view of the replay on the phone involves interaction of the outdoor game client with the server in-order to retrieve the recorded race data and render the replay on-screen.

During the experience useful information about the game is displayed in the HUD respecting the same game flow:

First the user must enter a game area, corresponding to the zone surrounding the slope (it can be at the bottom or at the top of the slope). This will automatically trigger the meeting phase, where remote skiers are represented and animated.

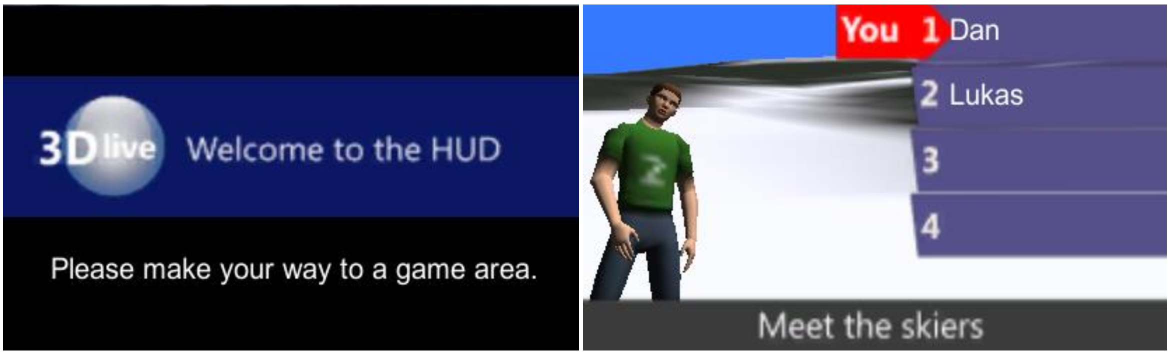


Figure 47: Welcome screen and Meeting screen of the HUD

Once the meeting is over, navigation directions to the starting gate are provided to the outdoor skier. As long as he skies down he can get closer to the gate and to the other players.

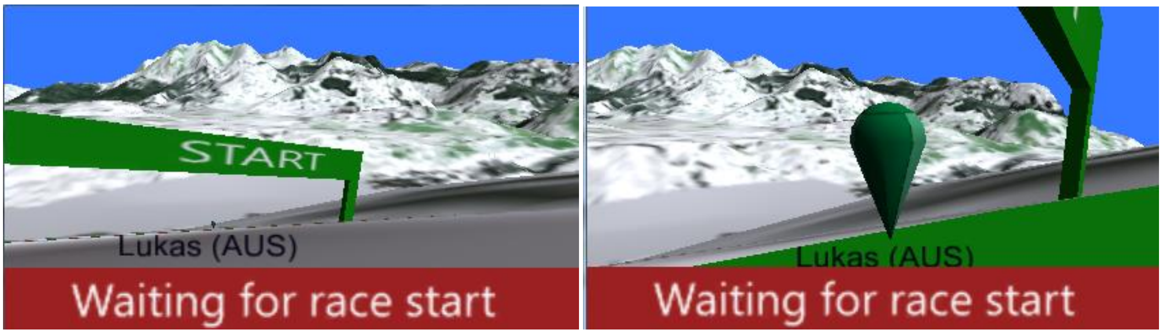


Figure 48: Starting Gate with pins of other players visible in the HUD

Once the outdoor user is at the starting gate, she can ask her competitors whether they are ready and trigger the countdown thanks to her wrist controller.

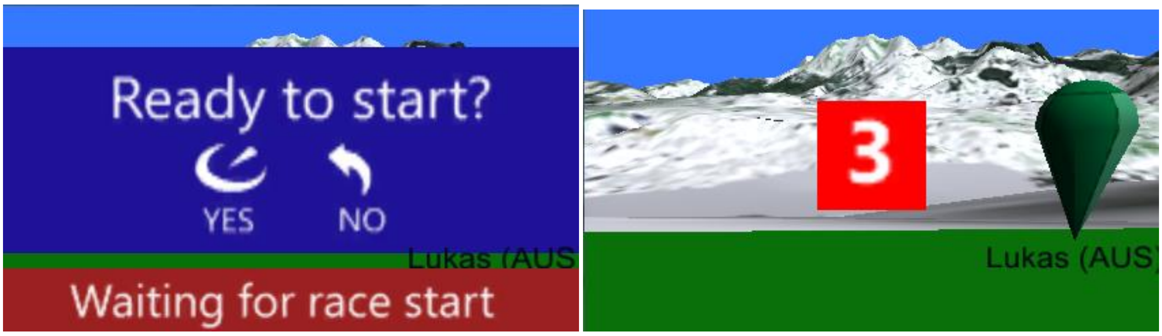


Figure 49: Triggering the start of the race thanks to wrist controller

The race can start. The screen can either be hidden for safety reasons or a bird's eye view can be displayed at lower speeds to see the progress of other racers on the slope, until everyone reaches the finish line.

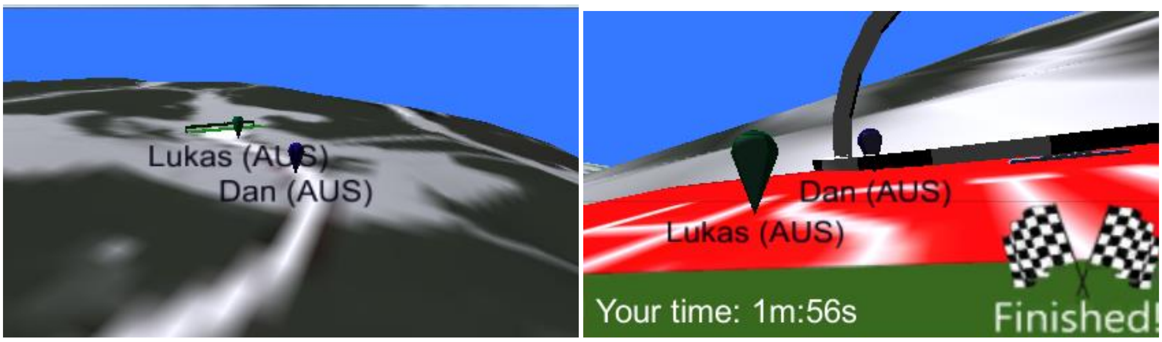


Figure 50: Bird's eye view and finsih line during a race

At the bottom of the slope, a podium is shown with remote users on it. The scores can be displayed or hidden thanks to the wrist controller. By turning her head around the user's HUD will automatically adjust virtual world's render view in order to always be consistent with real mountains locations.

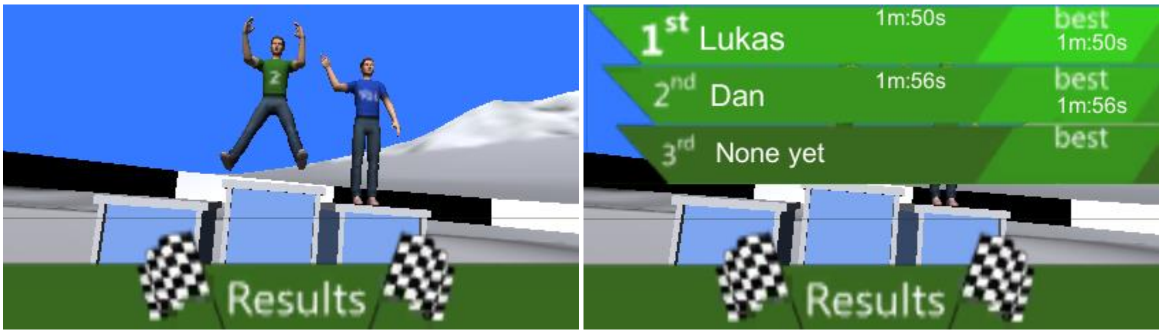


Figure 51: Final results. The remote player happy on the podium

Golfing:

The mobile application for golfing is pretty similar to the indoor one. The main difference is that the outdoor doesn't see the virtual environment but has an augmented view (augmented reality): (s)he can see the indoor player's avatar, golf balls and their trajectories.

Regarding 2D user interface elements, it contains the same content as the indoor game client application. In addition, two other features were implemented to the outdoor game client:

- Locating the ball after a shot: when she performs her shot, the outdoor player has to go to the ball's landing spot in order to get its precise position. He can press the "Lost ball" button if the ball is lost, or press the "In the hole" button if she succeeds to put the ball in the hole.
- Repositioning the ball if the indoor ball is chosen: in this case the player will see an augmented target on the ground where she needs to put the ball for the next shot.

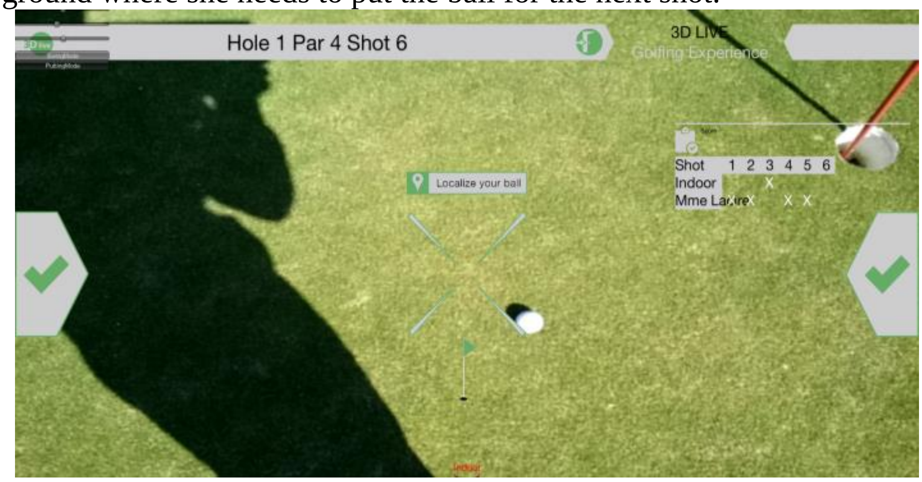


Figure 52: Unity3D outdoor app for golfing – measuring ball position

Suitability in 3D-LIVE: Using Unity3D for the outdoor mobile applications was pretty reliable. Indeed, as it is a cross platform engine, we saved a lot of time by using modules we already implemented for the indoor apps. Moreover we didn't need any transition module to connect the application to the server.

Masks

In the Live 3 Skiing prototype the Recon Snow 2 smart goggles were deployed. This device is running a custom Android OS and applications developed for Android are directly compatible with the goggles. Consequently using Unity3D to develop a custom HUD application was an obvious decision.

To handle the important amount of data to transmit in real time between the HUD and the phone, a WiFi pairing solution has been chosen. Information, Scores, Positions, Orientations and skeletons of the players, are shared using JSON formatting and UDP socket communications. In the Live 3 Skiing prototype we deployed the Recon Snow 2 smart goggles. This device is running a custom Android OS and applications developed for Android are directly compatible with the goggles. Consequently using Unity3D to develop a custom HUD application was an obvious decision.

Figure 53: HUD + wrist controller

To handle the important amount of data to transmit in real time between the HUD and the phone, a WiFi pairing solution has been chosen. Information, Scores, Positions, Orientations and skeletons of the players, are shared using JSON formatting and UDP socket communications.

2.10 Audio Client

As described in section 2, for audio communications in the Jogging scenario and in the last prototypes of

Skiing and Golfing an external audio server, called mumble has been used. The usage of this server requires the indoor/outdoor users to have a separate application installed in their hardware platform, called the Mumble Application. This application serves communications.

For indoor users the mumble application has deployment binaries for desktop computers while for the outdoor users that use a smartphone, the android mumble application is used. In both cases, either desktop deployment or smartphone deployment, the Mumble application runs in the background thread of the computer/smartphone and captures voice data when specific threshold of sound amplitude is exceeded.

Even though mumble is light weight this previously mentioned method conserves energy and bandwidth when using mobile devices.

2.11 Experiments Monitoring (EXPERImonitor)

The EXPERImonitor is a framework focused on the management of experiment content that allows developers to explore the relationship between QoS and QoE in complex distributed multimedia systems. The tool is specifically designed to support the observation of systems where user-centricity, mobility, ad hoc participation and real-time access to information are critical to success.EXPERImonitor uses a hybrid data model that combines formal reporting of QoS and QoE data types. The hybrid approach provides the ability to collect large quantities of measurement data (e.g. service response times, network latency, user satisfaction, etc) whilst allowing for exploration of causation between observations within such data (e.g. user satisfaction in relation to service response time).

The ability to efficiently traverse experiment content between QoS and QoE is an essential capability for evaluation of complex socio-technical systems. Data exploration can provide indications of factors that influence each other and is used to segment data for further investigation and analysis. With ever growing big data sets generated by Internet systems, EXPERImonitor can significantly reduce the time from observation to insight.

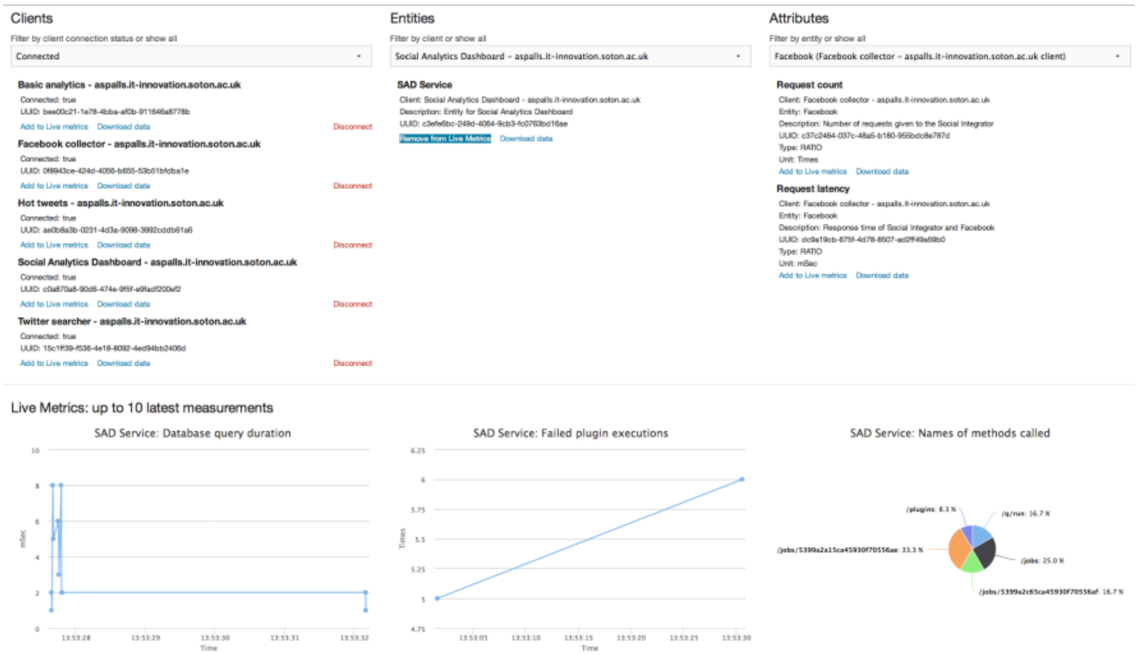


Figure 54: Example screenshot from the EXPERImonitor web dashboard

EXPERImonitor is a web service with a web-based admin interface and a REST API which connects to clients via RabbitMQ in order to receive high-volume monitoring data. Client APIs are available in multiple languages (Java, Android, C#, C++, Ruby). The web interface offers a live view of incoming metric data and a data explorer view for completed experiments. Data may also be exported to CSV files for further analysis in more specialised tools.

The software was developed primarily in the EXPERIMEDIA project but has been deployed and technically enhanced in collaboration with the 3D-LIVE project.

Suitability in 3D-LIVE: The EXPERImonitor was selected as the experimental management system as it is well aligned with the procedural and technical experimental requirements of the 3D-LIVE project. Specifically, the EXPERImonitor provides experimental monitoring service that offers a machine processable method for formally specifying QoS and QoE metrics data that is compatible with the data capture requirements of the 3D-LIVE UX model. Additionally, the EXPERImonitor provides a real-time experiment monitoring dashboard that allows the experimenter to view experimental data live as the experiment progresses, allowing for rapid verification of experiment data. Finally, the EXPERImonitor provides experiment data aggregation and export so that data collection for analysis is expedited.

2.12 Network Monitoring and Adaptation Service & Network Monitoring Client

As briefly described in section 2, the Network Monitoring and Adaptation Service has a twofold role in the deployment of the 3D-LIVE platform. Firstly, to monitor the network performance of the 3D-LIVE's infrastructure. Secondly, to provide suggestions about the compression level of the 3D-Reconstruction data by taking into account the network performance measurements. The suggestions are based in a rule-set that aims to provide the best user immersion possible under the restrictions imposed by the network. The suggestions may favour visual quality or frame update rate (later, in section 3.4, being referred as speed). Whatever the preference is, the Network Monitoring and Adaptation Service will never propose compression levels that result into extremely low update rates of the 3D-Reconstructed data at the Game Clients. The suggested compression levels are finally consumed by 3D-LIVE Capturer (section 2.7) in which they are used to determine the compression parameters to be fed into the Mesh Compression algorithm for compressing 3D-Reconstructions. This component constitutes innovative 3D-LIVE technology, which has never been studied before in the context of Tele-Immersion systems. Vital details about the internals of the module are given in section 3.4.

3 3D-LIVE innovative technologies

In the previous section all the deployment components of the 3D-LIVE platform were described along with details of their implementation in the case where they constitute application and integration of existing state-of-the-art methodologies. On the other hand, in the hereby section the modules that constitute research and technological advancements of the 3D-LIVE project, beyond existing technologies and methodologies, are described with in-depth details. Advances beyond existing technologies and/or adaptation strategies to the 3D-LIVE needs are thoroughly given, along with the technical metrics (Quality of Service) that were defined to evaluate the performance of the 3D-LIVE modules. For the relationship between tracked QoS metrics and the related QoE metrics, please, see D4.1.

3D-LIVE advanced the state of the art in many areas related to tele-immersion technologies, evident from the following publications:

- S. Asteriadis, A. Chatzitofis, D. Zarpalas, D. Alexiadis, P. Daras, "Estimating human motion from multiple Kinect Sensors", Mirage 2013, 6th International Conference on Computer Vision / Computer Graphics Collaboration Techniques and Applications, June 6–7, 2013, Berlin, Germany
- A. Doumanoglou, S. Asteriadis, D. Alexiadis, D. Zarpalas, P. Daras, "A Dataset of kinect-based 3D scans", 11th IEEE IVMSP Workshop: 3D Image/Video Technologies and Applications, Yonsei University, Seoul, Korea, 10-12 June 2013
- 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, issue 99, 2014
- A. Doumanoglou D. Alexiadis, S. Asteriadis, D. Zarpalas, P. Daras, "On human Time-Varying Mesh compression exploiting activity-related characteristics", IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2014. 4-9 May, Florence, Italy
- B. Poussard, S. Richir, J. Vatjus-Anttila, S. Asteriadis, D. Zarpalas, P. Daras, "3DLIVE: A Multi-Modal Sensing Platform Allowing Tele-Immersive Sports Applications", 22nd European Signal Processing Conference (EUSIPCO 2014), Lisbon, Portugal, 2-5 Sept. 2014
- D. Alexiadis, D. Zarpalas, P. Daras, "Fast and smooth 3D reconstruction using multiple RGB-Depth sensors", IEEE International Conference on Visual Communications and Image Processing, VCIP 2014, Dec 7-10, Valletta, Malta
- D. Alexiadis, A. Doumanoglou, D. Zarpalas, P. Daras, "A case study for tele-immersion communication applications: from 3D capturing to rendering", IEEE International Conference on

Visual Communications and Image Processing, VCIP 2014, Dec 7-10, Valletta, Malta

- S. Crowle, A. Doumanoglou, B. Poussard, M. Boniface, D. Zarpalas, P. Daras "Dynamic Adaptive Mesh Streaming for Real-time 3D Teleimmersion", submitted in the 20th International Conference on 3D Web Technology, (Web3D 2015), June 18-21, Heraklion, Greece.

3.1 Human reconstruction and animation

For the indoor users, depending on the available number of Kinect depth sensors and the quality of network (see section3.4) two different methods for user representation in the Tele-Immersive environment have been used in 3D-LIVE: Simple avatar animation, replicating human movements and full body 3D-Reconstructions of humans. The first case will be explained in the following subsections: avatar animation using the Kinect skeleton information and fusion of multiple skeletons captured from different Kinect sensors to handle occlusion issues and offer more accurate motion information, which will also be augmented with the process for animating the avatar of the outdoor users. The 3D-Reconstruction case that has been employed in the low-cost indoor case, will be presented in detail in section 3.1.4.

The human reconstruction and animation module uses Microsoft Kinect Depth Sensors (Section 2.1) and, for the outdoor runners, GPS data (2.1), while implementation is integrated in 3D-LIVE Capturer (2.7) and processing is made in Indoor Game Clients (2.8).

3.1.1 *Avatar animation*

As already discussed in section 2.7, for raw joint positions and orientation extraction, the Microsoft SDK framework has been used for the 3rd prototype. The Kinect offers skeleton data at 30 frames per second. Each skeleton frame contains pose and orientation information for all the joints of the user. For each joint orientation, a quaternion in the global coordinate system is used. From human skeleton extraction, avatar animation is performed via skeletal animation retargeting (avateering). Animation Retargeting refers to animating 3D characters using the reference Kinect motion capture data. The Kinect's joint orientations are applied to the avatar's bones in the global coordinate system taking into account the hierarchical placement of the bones. Thus, the order at which the skeleton joints are updated matters, depending on the avatar's hierarchical construction. Bones that are lower in the hierarchy are updated first due to the fact that the orientations are given in absolute coordinate system. Otherwise, the child bones would be transformed because of the rotation of the parent bones. To accomplish skeletal animation retargeting, the initial position of the avatar is set to T-pose, as depicted in Figure 55.

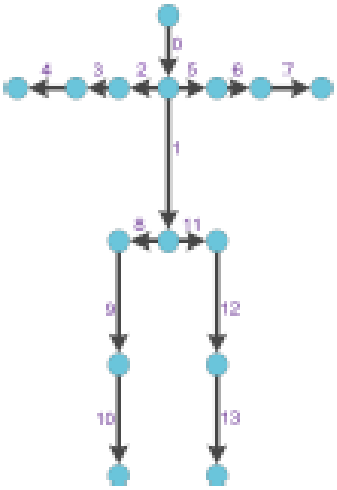


Figure 55: Skeleton in T-Pose

In this pose, skeleton orientations provided by Microsoft Kinect SDK are considered to be reference quaternions. Let those quaternions be denoted as $q_{KinectInit}$. Moreover, on the one hand, let the quaternion q_{init} describe the initial bone orientation for an arbitrary bone of the 3D avatar in its initial pose, i.e. in T-pose. On the other hand, let q be MS SDK's quaternion describing the orientation of the corresponding joint at an arbitrary time-point. Then, applying skeletal animation retargeting is equivalent to setting the orientation of each bone of the avatar to $q * q_{KinectInit}^{-1} * q_{init}$.. See also [F. Destelle, 2014] and [K. Apostolakis, 2013] for similar techniques in skeletal animation retargeting.

In order to improve the animation quality, CERTH implemented a set of skeleton filters, aiming to aid the low quality of the captured skeletons offered from the raw capturer. The skeleton filters implemented inside 3D-LIVE Capturer are the following:

1. *Flipped Orientation Filter:* it corrects any 180° wrong orientation estimation especially on the legs.
2. *Jitter Filter:* Filters out any joint position/orientation jitter.
3. *Calibration Filter:* The filter aims to provide a more accurate mapping between the skeleton provided by the kinect and the animated avatar itself. The users are required to stand in a T-Pose and this is automatically detected by the application. Then, the filter would transform the kinect skeleton appropriately, so that a T-Pose captured by the kinect matches a T-Pose of the avatar.
4. *Anthropometric Filter:* Some times, the captured skeleton's legs would be turned inwards resulting into a totally unnatural position of the avatar. This would prevent this effect from happening.
5. *Smooth Orientation Tracking Filter:* This will make the avatar not follow the user's movement by 100%. On the contrary it would only follow user's movement up to a certain degree resulting into

smoother ending avatar poses.
6. *Yaw Cancel Filter:* This filter cancels any yaw bone rotation.
7. *Natural Pose Filter:* This filter smoothes the overall pose to a certain degree. Very awkward and unnatural poses get smoothed to less awkward and more natural poses.

3.1.2 *Skeleton merging*

One of the core components of the 3D-LIVE architecture involves accurate extraction of human motion parameters. One of the toughest bottlenecks, related to non-wearable sensors, however, has to do with their significant sensitivity to occlusions or self-occlusions (Figure 56). For the above reason, in 3D-LIVE, CERTH investigated a multiple Kinect sensor approach, where each user is being tracked by more than one Kinect sensor, each having a different point of view, to allow for more robust skeleton extraction and tracking, allowing for a more pleasant animation of the user's avatar. Initially, the skeletons obtained by the different Kinects are filtered to isolate only the ones that correspond to an existing human. This is done in order to eliminate potential falsely detected skeletons in the background. A population of n candidate positions $p_{j,n,k}$ around each joint j, for every Kinect sensor k is initialized, and each of these units is translated accordingly in the 3D space. In subsequent frames, each point $p_{i,n,k}$ is assigned a weight $w_{j,n,k}$, which depends on two factors (equation 1): the distance of each candidate position from the corresponding observation $s_{j,k}$ (the actual measurement of Kinect sensor k for joint j) and a series of confidence values, modelled as non-linear functions of heuristics, namely the joints' expressivity, their distances on the z-axis from the corresponding values of sensor's depth map and the possibility that $s_{j,k}$ belongs to a limb with expected posture. These factors are non-linearly fused by following a Mamdani Fuzzy-Inference-Scheme [Mamdani, 1975] delivering an overall confidence equal to $\lambda_{j,k}$. . Confidence and distance values define weight $w_{j,n,k}$:

$$w_{j,n,k} = \sum_k \{\lambda_{j,k}\}^{-1} e^{-|s_{j,k} - p_{j,n,k}|} \quad (1)$$

After each iteration t, a roulette wheel selection scheme [Chipperfield, 1995] is followed, encouraging more promising positions. The algorithm converges to a few candidates after a number of iterations, which give the final joint position. More details can be found in the 3D-LIVE paper "Estimating Human Motion from multiple kinect sensors" [Asteriadis, 2013], that was presented in Mirage 2013, 6th International Conference on Computer Vision / Computer Graphics Collaboration Techniques and Applications by CERTH.

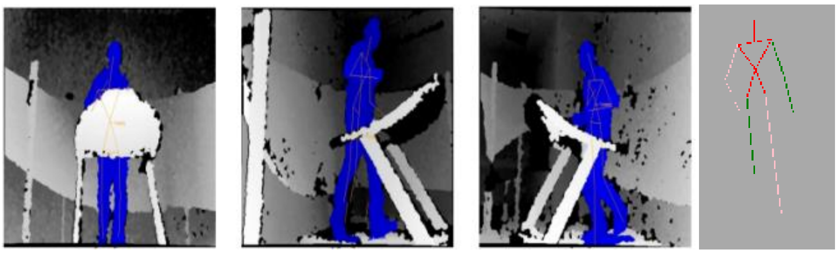


Figure 56: Human Skeleton from 3 different angles with occlusion, and the resulting skeleton, using joints from proper sensors.

3.1.3 *Outdoor user avatar animation*

In the outdoor user case when no other sensors but GPS-sensor was available, a predefined animation was used to move avatar around the virtual venue. Rendering engines contain some basic walking and running and idle animations, which were blended and interpolated to avatar GPS position changes to smoothly animate the avatar in the virtual world.

Pre-recorded animations were used in different ways depending on the scenarios. For instance in the Golfing and Jogging scenarios the animation is set function of the GPS inputs. The user speed influences the reading speed of the animation file and the speed of relocating the avatar on teh ground. For the ski scenario, animations of turns are available. So regardless the user is indoors or outdoors, the trajectory of the avatars' reference point is studied and, when a turn is detected, the proper animation is triggered depending on the amplitude of the turn.

About the animation based on wearable sensors, we can distinguish two cases:

- The Golfing swing animation
- The Avatar fused animation

The Golfing swing animation is the reading of a predefined swing animation based on real actual activity of the golfer. Indeed while taking a strike, the golfer's movement is recorded in real time by the inertial sensor attached to his golf club. This data provides us with a consistent value to use to animate a swing movement on the avatar. Thanks to this technique a user can see exactly when the remote player is taking his strike or training in real time.

The avatar fused animation is the combination of bones rotations coming from inertial sensors attached to the user's body and predefined animations. Within 3D-LIVE we experienced attaching four sensors to the body of one user to allow basic interactions. We proposed to attach one sensor on the torso, which is the reference sensor. In fact all the other orientations will be compared to this one in order to animate the bones of the avatar in local referential, allowing animations of the user in any position in the real world. One sensor is attached to the head, one other on the shoulder and the last one on the arm. Our fusion algorithm allows us to animate an avatar with predefined animations apart from the bones described above.

Consequently it allows a player running the 3D-LIVE mobile application to get his avatar animated in remote applications. One part of the skeleton animated with predefined animations function of the current activity of the player, and another part animated thanks to the real actual movements of the player.

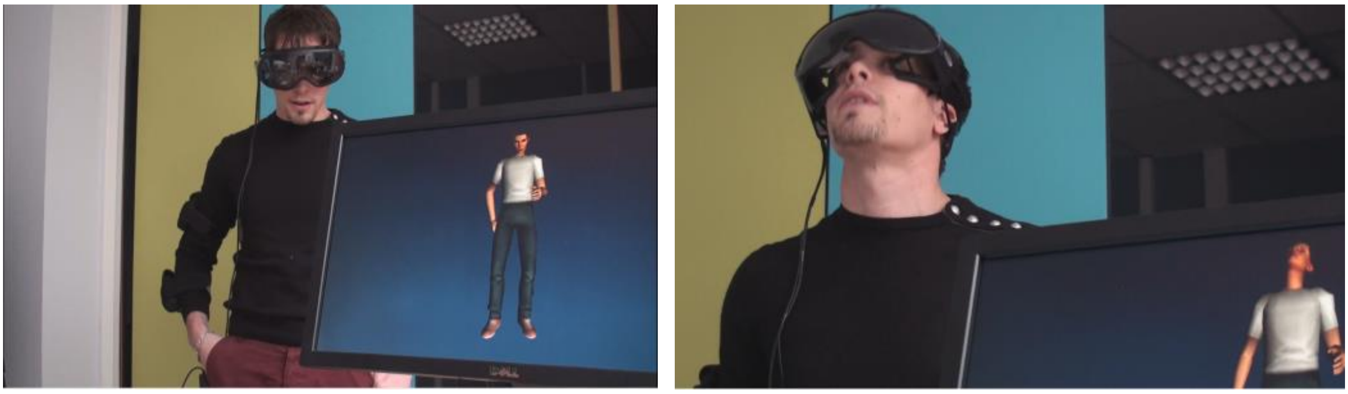


Figure 57: Head and arm animated with live sensors' data. Other parts of the avatar reading predefined animations

Based on the above technical description and the elicited user requirements (see D3.2 & D3.4), a series of technical Quality of Service metrics have been extracted in order to individually assess the usefulness and performance of the human 3D-Reconstruction & animation components (Table 5):

QoS metric for evaluation of *Avatar Animation*	**Explanation**
Skeleton Confidence (%)	Reports on confidence regarding whether 3D posture of human body is a replica of the actual one. This is inferred through confidence values delivered through the Capturer Application and is an inherent part of Microsoft SDK and OpenNi.
Skeleton quality (mm)	Reports on quality regarding whether avatar parts correspond to the actual point cloud of the user. This is measured through comparing Capturer’s inferred skeletal z-positions with the

	corresponding z-values in the same (x,y) position.
Skeleton Jerkiness (mm)	Metric that uses motion for motion-related metrics extraction. The more natural a motion is (less jerky), the more reliable it is that it follows real motion of the user. It is measured as the variance of motion within a small time window.
Avatar virtual X and Z-location (3D world coordinates)	Reports avatar location in virtual world for tracking and replication purposes. This metric can be translated to GPS position and can be compared to outdoor jogger actual GPS location to know how much virtual location differs from actual location.
Avatar virtual longitude and latitude location (digital degrees)	Actual latitude and longitude of the outdoor avatar in the virtual world, to be compared with the positions provided by the GPS of the mobile devices. This provides some indication about the path accuracy of outdoor users rendered in the game.

Table 5: QoS metrics for evaluating animation

3.1.4 *Real-time, 3D-Reconstructions of moving humans*

A major sense of tele-immersion is offered in 3D-LIVE, through the real-time visualization of the actual appearance (i.e. 3D-Reconstruction) of the indoor users. Thus, the very challenging problem of real-time robust 3D-Reconstruction of humans has been thoroughly investigated within 3D-LIVE, as this is a major scientific focus of CERTH.

Although accurate 3D-Reconstruction methods from passive RGB cameras can be found in the literature [Vogiatzis, 2007][Furukawa, 2009], they are not applicable in TI applications, since they require a processing time of several minutes per frame. Other, mainly visual hull-based methods [Franco, 2009], are quite fast, yet, they lack the ability to reconstruct concavities. Regarding methods that use active direct-ranging sensors [Curless, 1996], they have not been used in real-time applications; instead, they are applied off-line to combine range data captured by a single sensor.

Most of the relevant real-time TI-oriented approaches [Vasudevan, 2011] [Maimone, 2011] fuse partial 3D data only at the rendering stage, in order to synthesize intermediate 2D views. In [Vasudevan, 2011], a high quality TI system is described, including an accurate method for the generation of multiple depth-maps from stereo clusters, which are then combined at the rendering stage to synthesize views for given viewpoints. A similar TI system is described in [Maimone, 2011], where the depth-maps are captured by multiple Microsoft Kinect sensors. On the other hand, a Kinect-based 3D-Reconstruction system is presented in CERTH's previous work in [Alexiadis, 2013] that produces a single full 3D mesh, independently to the rendering stage. However, the explicit mesh "zippering" technique of that method may reject a significant amount of the captured information during the described "hard" procedure of decimating overlapping mesh regions. Moreover, it does not produce watertight and manifold meshes, which is a pre-requisite in 3D-LIVE applications. In the following, the steps followed for 3D-Reconstruction of moving humans in the 3D-LIVE platform, are presented. The proposed methodology has a number of innovations that were derived from the needs of the project, they have been implemented and are now used as part of the 3D-LIVE ecosystem. For a full analysis of the method, the reader is referred to 3D-LIVE's paper [Alexiadis, 2014], that was presented in the IEEE Visual Communications and Image Processing Conference (VCIP), 2014 and was selected among the conference's best papers.

1. *Capturing setup and calibration:* The RGB-Depth data are acquired by 4 Kinect sensors, placed in a circular spatial arrangement that provide full-body 360^{o} coverage of a human. In order to accurately estimate the internal parameters of each single Kinect, the method of [Herrera, 2012] is used. For the external calibration of the system, a simple, yet efficient method has been proposed in 3D-LIVE, that makes use of a large planar chessboard surface. The idea is based on the detection of a large number of 3D point correspondences in sets of sensors. Then, the extrinsic parameters are estimated for all cameras in an all-to-all manner.
2. *Raw point reconstruction:* Thresholding follows for foreground (human) extraction. Each foreground pixel of each depth map is reconstructed in 3D and its raw 3D normal is calculated through Step Discontinuity Constraint Triangulation (SDCT) [Alexiadis, 2013].
3. *Confidence-based 3D smoothing:* Instead of using the zippering method applied in CERTH's previous work [Alexiadis, 2013], which resulted to the loss of a significant portion of information and did not make use of measurements' quality criteria, a weighted smoothing approach for dealing with overlapping (coming from neighbouring sensors) adjacent surface regions has been investigated within the frame of 3D-LIVE. In particular, confidence values for each point were defined and a novel technique for smoothing (using surface normal) was introduced.
4. *Scalable volumetric reconstruction*: The characteristic volumetric function of the surface is extracted, implicitly defined as the iso-surface at an appropriate level. For this purpose, a Fourier Transform (FT)-based approach was followed [Kazhdan, 2005].
5. *Texture-mapping:* In 3D-LIVE, a confidence scheme was employed for 'colouring' the extracted iso-surfaces, using RGB information from every sensor's colour camera.

The major differences between CERTH's previous work on this subject (not within 3D-LIVE) with the 3D-LIVE method can be summarized as:

- In [Alexiadis, 2013], the explicit mesh “zippering” method may reject a significant portion of the captured information during the “hard” procedure of decimating overlapping mesh regions. Moreover, it does not produce watertight and manifold meshes.
- However in [Alexiadis, 2014], the captured data are fused in a weighted manner, based on the confidence of the corresponding depth measurements, and data are implicitly fused to generate manifold watertight meshes. The above improvements result in superior visual quality, at higher frame rates, both significant factors of tele-immersive experience.

Experimental Results

The experiments were ran on a PC with an i7 processor (3.2GHz), 8GB RAM and a CUDA – enabled NVidia GTX 560. When performing full-body 3D-Reconstruction, skeleton fusion can only be performed from kinects placed in horizontal orientation, since skeleton capture cannot work when the kinects are placed vertically. During 3D-LIVE experiments, 4 Microsoft kinects were used for 3D-Reconstruction from which only one is placed horizontally. The rest would be placed vertically in order to be able to sufficiently cover the human space. Thus, skeleton fusion in this case was not utilized. Please note that skeleton information in the case of 3D-Reconstruction is only required for the activity recognition module (thus, not for avatar animation), and that in order to offer visually pleasant 3D-Reconstructions of the users, we have avoided to capture the user with occlusions (i.e. in the jogging scenario we have removed the panel of the treadmill). In the 3rd prototype the skeleton information would be acquired by a Microsoft Kinect 2 as discussed in section 2.1.

“Skiing” sequence of 3D-LIVE: Reconstruction results for a specific frame of a person on a ski simulator are given in Figure 58. This is a frame of the sequence that 3D-LIVE through CERTH has contributed to MPEG-3DGC database, available at: `https://www.gti.ssr.upm.es/~mpeg/3dgc/3Dmodels/`. In this figure, the reconstructed raw points are illustrated, rendered using triangles generated by SDCT [Alexiadis, 2013] (left image), along with the output of the confidence-based smoothing operation (middle) and the volumetric reconstruction (right). The noise in the raw point-set (left image), as well as the effect of “Z-fighting” surfaces is obvious. On the other hand, the smoothed point-sets (middle) define a smooth surface. However, the triangular surface obtained by SDCT, although visually more pleasant, is non- manifold and contains cracks and holes. Therefore, a final watertight and manifold surface (right) is reconstructed using the volumetric method.

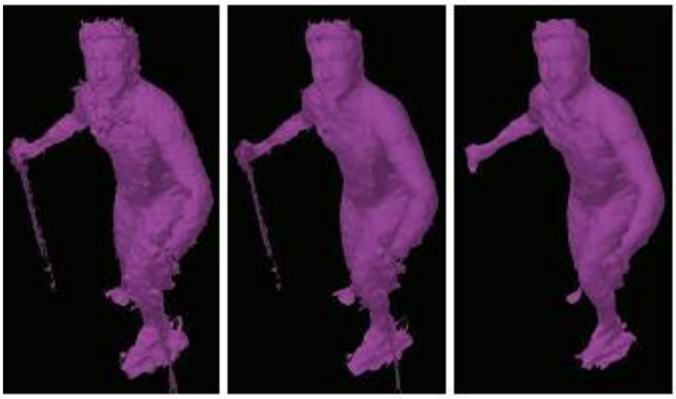


Figure 58: Raw, stitched and final reconstruction

Textured reconstruction results are also given in Figure 59. It is clear that the final reconstruction results contain much less artifacts and are visually superior.

Figure 59: Raw, smoothed and final reconstruction

Using the same sequence, Figure 60 provides some comparative results with the "zippering" method of [Alexiadis, 2013]. One can verify that a) the proposed method produces reconstructions with fewer artifacts and moreover, b) the method of [Alexiadis, 2013] can benefit from the proposed weighted smoothing method. In terms of time requirements, high reconstruction rates, close to 10 fps, can be achieved.

Figure 60: From left to right a) Raw SDCT reconstruction, b) Output of CERTH's previous work [Alexiadis, 2013], c) Output of [Alexiadis, 2013] after applying the proposed weighted smoothing method, d) watertight volumetric reconstruction [Alexiadis, 2014]

Experimental results using the Huawei/3DLife dataset (http://mmv.eecs.qmul.ac.uk/mmgc2013/) are reported in the following figures. Figure 61 presents results for a specific frame of the "Xenia" sequence. The raw textured point-set is presented on the left, followed by the output of the smoothing operation and the final, confidence-based watertight reconstructed mesh.

Figure 61: Raw point set, smoothing, final reconstruction

Figure 62: a) Raw SDCT reconstruction, b) Output of [Alexiadis, 2013], c) Output of [Alexiadis, 2013] after applying the proposed weighted smoothing method, d) watertight volumetric reconstruction

Figure 63: Results for two frames and two point-views

Figure 62 provides comparative results with the "zippering" method of [Alexiadis, 2013]. Finally,

Figure 63 presents the reconstruction results for two frames of the “Stavroula” sequence.

3.1.5 *Evaluation of 3D-Reconstruction*

Given that the evaluation of 3D-Reconstruction has not been addressed before, CERTH has formalized this by introducing a method that compares the visual quality of the reconstructed mesh with a high quality capture of the subject via a high quality camera. Comparison will be based on the visual quality of the 3D-Reconstruction from an arbitrary viewpoint, i.e. a viewpoint different from the ones of the Kinects used for the creation of the 3D-Reconstruction. The metric is based on Peak Signal To Noise Ratio (PSNR). Apart from the 4 Microsoft Kinects being used for 3D-Reconstruction, the capture setup is equipped with an additional RGB camera. This RGB camera is calibrated in its intrinsic parameters. Additionally, it is calibrated in its extrinsic parameters with respect to the rest of the Microsoft Kinects. In the 3D-LIVE Capturer software, the 3D reconstructed mesh is rendered from a viewpoint defined by the camera’s extrinsic parameters and projected to the image plane via the camera’s intrinsic parameters. This operation is hardware accelerated via OpenGL rendering in an off-screen buffer. The resulted rendered image of the 3D-Reconstruction is then used to compute its PSNR with respect to the reference image obtained by the camera itself. The PSNR computation is evaluated in two forms. The first form is instantaneous, i.e. independently per frame, as if computing PSNR between two images. On the other hand, the second form is sequential. The PSNR is computed between two video sequences (i.e. the PSNR is computed over the whole sequence of frames like in a video). This evaluation was introduced into QoS metrics, as described in Table 6.

QoS metric for evaluation of *Real time, 3D-Reconstructions of moving humans*	**Explanation**
Reconstruction Instantaneous PSNR (dB)	This is a way to evaluate aesthetic quality of the reconstructed mesh. An extra, calibrated camera is placed as part of the setup. Human’s 3D-Reconstructions from kinect sensors (depth + RGB), are virtually projected on the corresponding 2-D plane and compared against RGB values returned by this camera.
Reconstruction Sequence PSNR (dB)	Gives an overall aesthetic result of a whole session of interaction, by computing the PSNR as in video sequences.

Table 6: QoS metrics for 3D-Reconstruction

3.2 Activity Recognition in 3D-LIVE

3D-LIVE adopts a multi-modal approach for Activity Recognition. Wearable and non-wearable technologies are interchanged, in order to give a full framework of activity reports. These are further exploited by the 3D-LIVE ecosystem in various ways: Feedback to the user, input to the rendering engine, scenario evolution. The Figure 64 below is illustrative of the 3D-LIVE approach:

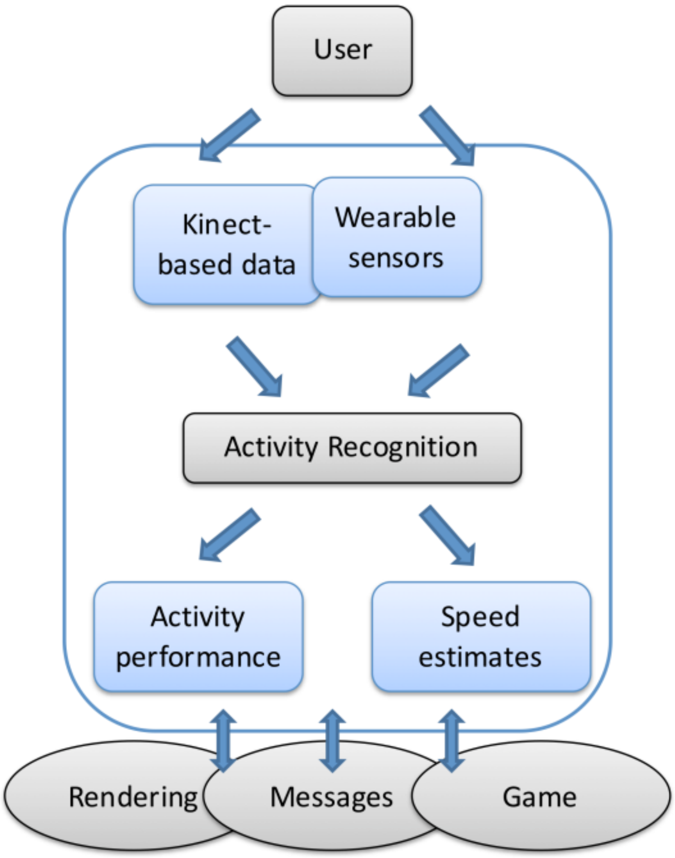


Figure 64: Activity Recognition in 3D-LIVE

This module uses Microsoft Kinect Depth Sensors (Section 2.1), wearable sensors (2.1) and, for the outdoor runners, GPS data (2.1) while implementation is integrated in 3D-LIVE Capturer (2.7) and processing is done in Indoor Game Clients (2.82.8).

3.2.1 *Speed estimates*

Using Kinect depth sensors: Using three-dimensional information retrieved with the use of the Kinect sensor, it is possible to extract information related to user's speed, either through exact calculations (jogging) or through indicates correlated to speed (ski). Below is 3D-LIVE's approach conducted by CERTH to speed estimation through Kinect-based information.

Jogging: Prior to evaluating a jogger's activity, the type of activity has to be defined. In 3D-LIVE, we make an automatic distinction among four basic categories, which require a speed estimation module: 1. Running, 2. Jogging, 3. Walking, 4. Standing still. To this aim, a training set of known pairs of parameters (z_f, u) has been recorded, with z_f being feet's average depth (z) position, and u corresponding to actual treadmill speeds. A first-order regression model, mapping first order positive derivatives dz/dt to u is found using least-squares and this is subsequently used, for unknown data, in order to calculate user speed in a generic and easy to implement way and, thus, avoid complex solutions for extracting it from the treadmill itself. Thresholds are defined in order to infer whether user is standing still, walking, jogging or running and, for the two last cases, proper heuristics and fuzzy machines are triggered for activity evaluation.

Skiing For the outdoor user's avatar, its speed estimation is calculated from the distance and time difference of two sequential GPS points. The avatar's running speed and the predefined animation's speed are modified accordingly to match real life motion.

The problem in the ski scenario is that GPS positions are updated on average every second (with differences of up to 4 seconds). With the skier reaching speeds of 28m / s we are obliged to predict the path trajectory of the skier and correct it as a function of GPS data received. Not predicting the trajectory would result an error of about 20 meters, which would be considered too much. The objective is to obtain an avatar in the virtual world that keeps a minimal error with respect to the actual position of the skier while having a behaviour that seems realistic for indoor users.

To improve the accuracy of the signals received by the Smartphone, we integrated a Kalman filter. Its prediction model is based on simple kinematics equations and the observation model is fed by new incoming GPS data, in order to reduce the noise of the raw signal.

Each time GPS data is received, an invisible virtual ski controller (the same used to simulate indoors players' avatars), called "Ghost" is teleported at the coordinate received. It is allocated a speed (corresponding to the speed to get to the point of the previous point received) and a direction (corresponding to the direction described by the GPS coordinate received and the previous one). The following positions of the "ghost" are then calculated in the virtual world in the same way that is simulated for the behaviour of indoor player.

The avatar of the outdoor user in the virtual world seeks to achieve the position and velocity (direction and intensity) of the "ghost". When the "ghost" and the avatar are not in the same place at the same speed, the avatar will change its speed to reach the “ghost”. In order to have a plausible behaviour for indoor players who will ski with him, the maximum acceleration of the controller has been limited. This way, if the avatar and the "ghost" are found to have a significant position or speed difference, the avatar will take time to recover the delay. But the experience of players using simulators will be good since they will not see any strange behaviour.

To further increase the accuracy of speed estimation for the outdoor user, we used an IMU (section 2.1.1) that is attached to the torso of the skier. The orientation of the "ghost" is then defined by the orientation of the IMU. Unfortunately, we were not able to test and validate the outdoor usage of the IMU, as it was very difficult to set up for experiments in the Schladming site.

3.2.2 *Activity performance*

Kinect-based data: A set of metrics describing body posture was first defined, in order to assist in the implementation of fuzzy and crisp rules that were used for indoor activity performance evaluation. The feedback of this procedure is given in the form of messages, delivered, on the fly, to the indoor user whose activity is being tracked by Kinect Sensors. Figure 65 and Table 7 show the joints extracted by the Kinect sensor, while Table 8 and Table 9 are illustrative of the metrics used.

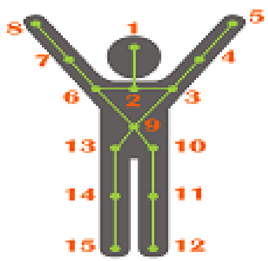


Figure 65 Joints tracked by the Kinect sensor

ID	Joint
P_1	Head

P_2	Neck
P_3	Left Shoulder
P_4	Left Elbow
P_5	Left Hand
P_6	Right Shoulder
P_7	Right Elbow
P_8	Right Hand
P_9	Torso
P_{10}	Left Hip
P_{11}	Left Knee
P_{12}	Left Foot
P_{13}	Right Hip
P_{14}	Right Knee
P_{15}	Right Foot

Table 7 Joint IDs tracked by the Kinect sensor

Metric	Calculation
HipsAngle	Angle formed by line segment $P_{10}P_{13}$ with the horizontal axis
ArmBones	Average of angles formed by vectors P_3P_4 - P_4P_5 and vectors $P_6P_7 – P_7P_8$
ArmSide	Average of angles formed by vectors $P_3P_4 – P_2P_9$ and vectors $P_6P_7 – P_2P_9$
KneeAngle	Average of angles formed by vectors $P_{13}P_{14} – P_{14}P_{15}$ and vectors $P_{10}P_{11} – P_{11}P_{12}$
LeanForw	Angle formed by vector P_2P_9 with the vertical axis

Table 8: Calculation of body metrics for Jogging activity recognition

Metric	Calculation
ArmSide	Average of angles formed by vectors $P_3P_4 – P_2P_9$ and vectors $P_6P_7 – P_2P_9$
KneeAngle	Average of angles formed by vectors $P_{14}P_{13} – P_{14}P_{15}$ and vectors $P_{11}P_{10} – P_{11}P_{12}$
Bending	Average of angles formed by vectors $P_{10}P_2 – P_{10}P_{11}$ and vectors $P_{10}P_2 – P_{13}P_{14}$
TorsoUpright	Angle of P_2P_9 with the horizontal axis
LegsLean	Average of angles of $P_{11}P_{10}$ and $P_{14}P_{13}$ with the horizontal axis
KneeDist	Distance between P_{14} and P_{11} normalized with the average leg length

Table 9: Calculation of body metrics for Skiing activity recognition

By knowing the relevant positions of body joints, CERTH proposed the use of fuzzy logic[14] for estimating performance scores of sports activities; In particular, we defined variables of body metrics and we defined two or three fuzzy sets for each: *LOW*, *MED(IUM)* and *HIGH*, *with MED being optional* (see Appendix A for the corresponding values). Similar, performance (output) of the Fuzzy Inference System was modelled by two fuzzy sets (*LOW and HIGH*) from 0 (bad performance) to 100 (excellent performance). In the jogging scenario, fuzzy sets for extremes (LOW,HIGH) were modelled by ramp membership functions (MFs), while Triangles were used for MED sets. On the other hand, in the skiing scenario MEDIUM fuzzy sets were modelled by Gaussian membership functions (MFs), while Sigmoidals were used for low and high sets. Below are the rules used in the module:

Jogging:

RULE 1: if **HipsAngle** is LOW then **JogEvaluation** is HIGH

RULE 2: if **HipsAngle** is HIGH then **JogEvaluation** is LOW

[14] The FuzzyLite library V5.0 was used: http://www.fuzzylite.com/

RULE 3: if **ArmBones** is LOW then **JogEvaluation** is LOW

RULE 4: if **ArmBones** is HIGH then **JogEvaluation** is LOW

RULE 5: if **ArmBones** is MED then **JogEvaluation** is HIGH

RULE 6: if **ArmSide** is LOW then **JogEvaluation** is HIGH

RULE 7: if **ArmSide** is HIGH then **JogEvaluation** is LOW

RULE 8: if **LeanForw** is LOW then **JogEvaluation** is HIGH

RULE 9: if **LeanForw** is HIGH then **JogEvaluation** is LOW

RULE 10: if **KneeAngle** is LOW then **JogEvaluation** is LOW

RULE 11: if **KneeAngle** is HIGH then **JogEvaluation** is HIGH

Walking: speed between 1,5 km/h and 5.5 km/h

Jogging: speed between 5.5 km/h and 7.5 km/h

Running: speed above 7.5 km/h

Ski:

RULE 1: "if **KneeAngle** is MEDIUM and **Bending** is MEDIUM and **ArmSide** is MEDIUM and **TorsoUpright** is LOW and **LegsLean** is HIGH and **KneeDist** is HIGH then **EvalSki** is HIGH"

RULE 2: "if **KneeAngle** is HIGH and **Bending** is HIGH and (**ArmSide** is LOW or **ArmSide** is HIGH) and **TorsoUpright** is HIGH and **LegsLean** is LOW and **KneeDist** is LOW then **EvalSki** is LOW"

The system for Jogging is 'Takagi-Sugeno-Kang' while for Skiing is of type 'Mamdani', with min and max *t*-norm and *s*-norm used for the AND and OR operators. For every Rule, implication is done using the 'min' operator, while the 'max' operator is employed for aggregation. Finally, defuzzification for the 'Mamdani' model is done using the Centre of Gravity of the output while for the 'Takagi-Sugeno-Kang' is done using the Weighted Average of the output. The models and their parameters were chosen after extensive experimentation with a sufficient amount of pre-recorded motion capture data.

Figure 66: Demo of Activity Recognition for Jogging

Figure 67: 3D-LIVE Capturer performing Jogging Activity Recognition

However, apart from activity scores, the 3D-LIVE activity recognition component also gives feedback to the user with regards to *what must be improved* to increase his/her performance. To that aim, CERTH employed a series of anthropometric rules, based on expected body posture, depending on the activity. Apart from the fuzzy reasoner described above, different types of feedback are also given to the user for different cases:

Jogging, Case 1: speeds between 5.5 and 7.5 klm/hour

Warnings are issued if:

-**HipsAngle** is too large

-**ArmBones** do not form an angle close to 90°

-**ArmSide** angle is too big (arms should be as close to the torso as possible)

-**LeanForw** angle is too big (the torso must be upright while running)

-Maximum value of ***KneeAngle*,** over a period of 30 frames, is too large (feet should be close to the ground to save energy)

Jogging, Case 2: speeds between 5.5 and 7.5 klm/hour

Similar rules with Case 1, with the exception of the maximum value of **KneeLift,** over a period of 30 frames, that should be larger than in case 1.

Ski:

-**ArmSide** angle must be within certain limits

-TorsoUpright angle should be small enough.

-KneeAngle angle should not exceed a certain value

-Bending angle should not exceed a certain value (the skier' waist must be in a bending position)

-Minimum value of **LegsLean** over a period of 30 frames should be small enough as legs should be leaning sideways while skiing.

-KneeDist should also be large enough, since knees should have a certain distance with each other while skiing.

At the end of each session, an average performance score, as well as a list of suggestions to the skier are presented as feedback for better performance at the next session.

QoS metric for evaluation of *Activity Analysis based on Kinect data*	**Explanation**
Skeleton Confidence (%)	Reports on confidence regarding whether 3D posture of human body is a replica of the actual one. This is inferred through confidence values delivered through the Capturer Application and is an inherent part of Microsoft SDK and OpenNi.
Skeleton quality (mm)	Reports on quality regarding whether avatar parts correspond to the actual point cloud of the user. This is measured through comparing Capturer's inferred skeletal z-positions with the corresponding z-values in the same (x,y) position.
Skeleton Jerkiness (mm)	Metric that uses motion for motion-related metrics extraction. The more natural a motion is (less jerky), the more reliable it is that it follows real motion of the user. It is measured as the variance of motion within a small time window.

Table 10: Activity analysis QoS

For a technical explanation of the above metrics, please, see D4.1 (report on the experimentation & evaluations of the 3D-LIVE Tele-Immersive Environment).

3.3 Data compression and transmission

Compression of Human 3D-Reconstructions

Given that the "heaviest" data to be transmitted is the human 3D-Reconstruction data, CERTH has conducted research in 3 different techniques for time-varying mesh (TVM) compression, a specific topic that is largely unexplored by the research community so far. In the first one, entitled "Towards Real-Time and Efficient Compression of Human Time-Varying Meshes" (published in IEEE's TCSVT, [Doumanoglou 2014 (A)]) a novel skeleton-based approach to Human Time-Varying Mesh (TVM) compression is presented. We attempt to address the Human TVM compression problem inspired from video coding by using different types of frames and trying to efficiently remove inter-frame geometric redundancy utilizing the recent advances in human skeleton tracking. The overall approach focuses on compression efficiency, low distortion and low computation time enabling for real time transmission of Human TVMs. It efficiently compresses geometry and vertex attributes of TVMs. Moreover, this work is the first to provide an efficient method for connectivity coding of TVMs, by introducing a modification to the state-of-the-art MPEG-4 TFAN [K. Mammou 2009] algorithm. This work also proposes a method for motion-based coding of Human TVMs that can further enhance the overall experience when Human TVM compression is used in a Tele-Immersion (TI) scenario. This work, given its high novelty has been presented in the 105th MPEG Meeting, and more specifically in the 3D Graphics Group ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio with input document m30537[15]Jul-Aug 2013, Vienna.

In the second one, entitled "ON HUMAN TIME-VARYING MESH COMPRESSION EXPLOITING ACTIVITY-RELATED CHARACTERISTICS", presented at IEEE's ICASSP conference,

[15] http://wg11.sc29.org/doc_end_user/documents/105_Vienna/wg11/

[Doumanoglou 2014 (B)] the potential of exploiting activity related global features in order to improve the performance of the existing human Time-Varying Mesh (TVM) compression scheme of [Doumanoglou 2014 (A)] is presented. The paper introduces a strategy for selecting the most appropriate I-Frame that will serve as a reference frame for the encoding of EP-Frames, exploiting activity-related characteristics, given that in 3D-LIVE the users perform specific activities (i.e. ski and jogging). Two different strategies are presented, using a skeleton-matching criterion and a periodicity measurement metric based on human skeleton. Results show that the concept is sound, but they also reveal the sensitivity of the proposed methods to the skeleton quality (which is offered by the available Kinect SDKs).

Lastly, in the paper entitled "A case study for tele-immersion communication applications: from 3D capturing to rendering" key aspects related to next-generation tele-immersion applications are analyzed, studying the end-to-end chain from 3D capturing of remote users to rendering. Key modules include 3D-Reconstruction and mesh compression. In this paper the reconstructed geometry, along with the vertex attributes, is compressed using the open source static mesh compression scheme OpenCTM. The texture images of the mesh are separately compressed using H.264 video coding. A wrapper for the openH264 (http://www.openh264.org/) library was developed and used for that purpose. The tele-immersion pipeline performance with respect to the various parameters was in detail evaluated with respect to visual quality and real-timeness. An advantage of the system is its scalability, since both the 3D-Reconstruction and compression modules are scalable and adaptable to the characteristics of the users' workstations and the network. This work was presented at IEEE's VCIP conference.

The main differences between the different compression schemes are highlighted (with green color the advantages and with red the disadvantages) in Table 11:

TCSVT / ICASSP methods	VCIP method
Highly novel: Introduces a chain of innovative ideas, giving a practical proof of concept that efficient inter-frame redundancy exploitation in TVMs is possible and **establishing ground for further improvements**.	Efficient combination of robust and established methods.
Exploits temporal redundancy fully[1] → **Better compression ratio / distortion** performance	Exploits temporal redundancy only for textures[2]
• Relies on robust extraction of high-level features (i.e. Kinect based skeleton information).[3] • Only for moving humans	• Does not rely on the extraction of high-level features. • More generic
More complex and computationally expensive[4]	Simpler → **Faster**
Highly novel	Not novel but mature and stable.

Table 11: Comparison between TCSVT/ICASSP and VCIP compression method

[1] For both geometry and color

[2] Via established video coding schemes

[3] Current technology does not guarantee robust human skeleton extraction

[4] **Near** real-time with current technology

In 3D-LIVE live experiments, the compression method presented at VCIP was favored against the one of TCSVT/ICASSP for the following reasons:

- It is an efficient **combination** of **robust** and **established methods**
- It is **simpler** and **faster** compared to the TCSVT/ICASSP method

However, on bad network conditions, where **packet loss** can occur, **both methods suffer** from the fact that the decoding of a frame relies on the successful decoding of previous frames, since they both exploit inter-frame redundancy. This problem is **non-trivial** and a **variant of the VCIP** method was developed to face the new challenge. The new method **replaces** the **H264 video codec** with a static **JPEG** image compressor, in an effort to reduce the fragility of the algorithm on bad network quality.

The implementation of the compressor side of this module is integrated in 3D-LIVE Capturer (section 2.7) while the decompressor is implemented and deployed to the Indoor Game Clients (section 2.8).

QoS metric for evaluation of *3D-Reconstruction Data Compression*	**Explanation**
Reconstruction Compression FrameRate (FPS)	Reports on the rate at which reconstructed frames are compressed. The value is measured in frames per second.
Reconstruction Average Compression Ratio	Reports on the compression ratio of the compressed mesh. This is a natural number calculated as (uncompressed size) / (compressed size). This metric is calculated as a running average since the beginning of the experiment.
Average Reconstruction Compression Time (sec)	Reports on the time it takes to compress a 3D-reconstructed mesh. It is measured in seconds and is calculated as a running average since the beginning of the experiment.

Compressed Reconstruction vs Reference RGB Camera Instantaneous PSNR	Reports on the quality of the compressed reconstruction. Refer to section 3.1.5 for details.
Compressed Reconstruction vs Reference RGB Camera Sequence PSNR	Reports on the quality of the compressed reconstruction. Refer to section 3.1.5 for details.

Table 12: QoS metrics for Compression of 3D-Reconstructions

3.4 Network monitoring and adaptive compression

A network monitoring module has been specified for the purpose of optimising the transmission of the most intensive network data used in the 3D-LIVE system: the full body reconstruction data. Before transmitting the data to the receiving user on the network, the data compressor reduces the overall size of the frame by compressing the data is a lossy fashion; the higher the compression level, the lower the quality of the reproduced full body reconstruction viewed by the receiver. However, the smaller the size of the frame data sent to the end user, the greater the potential for higher frame rates. Typical payloads of a compressed frame, compressed at different levels, is described in Table 13:

Full body compression level	**Typical payload (in bytes)**
High	~163k/frame
Medium	~192k/frame
Low	~244k/frame

Table 13: Typical payloads of a compressed frame at different compression levels

The network monitoring module attempts to optimise the rate at which frames can be sent and received during a 3D-LIVE game by capturing network conditions between the sender and receiver and selecting the appropriate level of data compression without causing an unnecessary drop in overall visual quality. This is done by monitoring the two primary factors that impact frame throughput: first network latency; second: bandwidth (for further information on network monitoring, see *D4.2: Second report on the experimentations and evaluations of the 3D-LIVE Tele-Immersive Environment*).

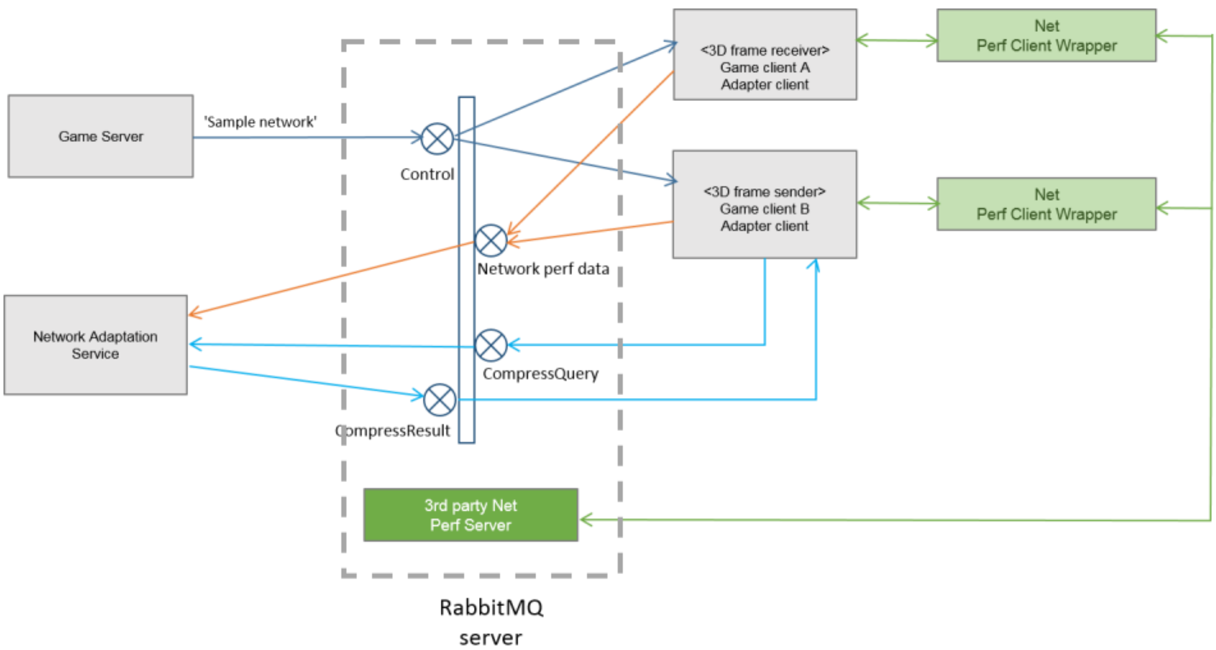


Figure 68: Network monitoring architecture

Additionally, in the context of T3.6, the compressor employed in 3D-LIVE was equipped with adaptive capabilities by varying its parameters according to the feedback obtained though network monitoring. The compressor's parameters are controlled via network messages emitted by the network monitoring module. The compressor has 3 different pre-saved configurations for 3 different levels of compression / visual quality modes (namely LOWQ, MEDQ, HIGHQ). The names of the levels refer to the corresponding visual quality of the compressed mesh. Thus, HIGHQ refers to a compressed reconstructed mesh of HIGH visual quality and low compression ratio. Similarly, LOWQ refers to a compressed mesh of LOW visual quality, thus high compression ratio. The compressor's performance is controlled through 2 different parameters: JPEG Image Quality and Geometry Vertex Precision. The JPEG Image Quality factor is a percent factor that controls the visual quality of the compressed textures of the reconstructed mesh and is proportional to the resulted compressed texture size, i.e. the higher the factor the larger the size. Geometry vertex precision is measured in millimeters and controls the precision at which the reconstructed mesh's vertex coordinates are encoded. Higher precision results into better visual quality and higher filesize. Content adaptation for the case of 3D reconstructed time-varying meshes is achieved by varying the parameters of the new compressor according to Table 14: Compressor Parameters:

Parameter	Description
JPEG Image Quality (%)	Directly affects the quality of the texture proportional to the compressed size.
Geometry Vertex Precision (mm)	Directly affects the quality of the mesh. Implicitly has impact on compressed size. The higher the precision the greater the data size.

Table 14: Compressor Parameters

Experimental results for different compression parameters are presented at Table 15 for 4 different sequences. The results depict: PSNR, Average Compression Ratio (CR) per frame and Average FileSize (in KB) per frame. The different configurations (compression parameters) are depicted in Table 16.

	CONFIG1			CONFIG2			CONFIG3			CONFIG4			CONFIG5			CONFIG6		
SEQUENCE	PSNR	CR	SIZE	PSNR	CR	SIZE	PSNR	CR	SIZE	PSNR	CR	SIZE	PSNR	CR	SIZE	PSNR	CR	SIZE
aldoum-Ski	16,47	63,81	59,55	16,76	51,75	73,42	16,84	44,81	84,79	16,91	39,90	95,21	16,96	33,37	113,84	17,05	28,45	133,55
aldoum-Ski2	17,05	63,31	60,02	17,38	51,26	74,12	17,53	44,35	85,67	17,66	39,60	95,94	17,67	33,02	115,07	17,77	28,24	134,53
dalexiad-Ski	15,08	71,37	53,23	15,41	57,11	66,53	15,67	49,23	77,17	15,77	43,67	87,01	15,78	36,19	104,98	15,94	31,10	122,16
dalexiad-Ski2	16,12	67,31	56,45	16,45	54,99	69,09	16,54	47,96	79,23	16,61	43,33	87,68	16,72	36,00	105,53	16,73	31,16	121,93
AVERAGE	**16,18**	66,45	57,31	**16,50**	53,78	70,79	16,65	46,59	81,72	16,74	41,63	91,46	16,78	34,65	109,86	**16,87**	29,74	128,04

Table 15 Compression Experimental Results

	JPEG Image Quality (%)	Geometry Vertex Precision (d) (mm)
CONFIG1	5	0.015
CONFIG2	10	0.012
CONFIG3	15	0.012
CONFIG4	20	0.010
CONFIG5	30	0.008
CONFIG6	40	0.001

Table 16: Compression Configurations

The PSNR metric presented at Table 15 was computed with the same methodology as described in section 3.1.5. The compressed mesh was rendered in an off-screen buffer according to an external RGB camera's calibration parameters and the resulted image was compared against the original captured image from the camera via PSNR. From the above experimental table we chose to assign LOWQ to CONFIG1, MEDQ to CONFIG2 and HIGHQ to CONFIG6, mainly taking into account the visual impact of the compression parameters to the reconstructed mesh. Below, screenshots for the sequences "dalexiad-ski2" and "aldoum-ski2" are presented for subjective visual quality evaluation among with the corresponding compression level, the PSNR metric measured for the whole sequence and the average Compression Ratio (CR) per frame for the sequence.

HIGHQ PSNR: 16.7 dB / CR: 31   MEDQ PSNR: 16.4 dB / CR: 55   LOWQ PSNR: 16.1 dB / CR: 67

Figure 69: Compressed reconstructions from the `dalexiad-Ski2` sequence

HIGHQ PSNR: 17.7 dB, CR: 28 MEDQ PSNR: 17.3 dB, CR: 51 LOWQ PSNR: 17.0 dB, CR: 63

Figure 70: Compressed reconstructions from the `aldoum-Ski2` sequence

In conclusion, we see that the results of the adaptive compressor vary in both compression ratio and visual quality terms, enabling for real-time tele-immersion experience under varying network quality conditions. Thus, the requirement for content adaptation in 3D-LIVE is fulfilled. Finally, all the work in 3D-LIVE for T3.6 has been documented in detail and submitted for publication in the 20th International Conference on 3D Web Technology, (Web3D 2015: http://web3d2015.web3d.org/):

S. Crowle, A. Doumanoglou, B. Poussard, M. Boniface, D. Zarpalas, P. Daras “Dynamic Adaptive Mesh Streaming for Real-time 3D Teleimmersion”.

3.5 Evaluation metrics

A wide range of metrics have been captured during the course of the 3D-LIVE experimentation. These metrics were collected from many of the technical modules that execute at run-time whilst running the

3D-LIVE system. The logical metric model used to represent this data is depicted in the entity relationship diagram of Figure 71.

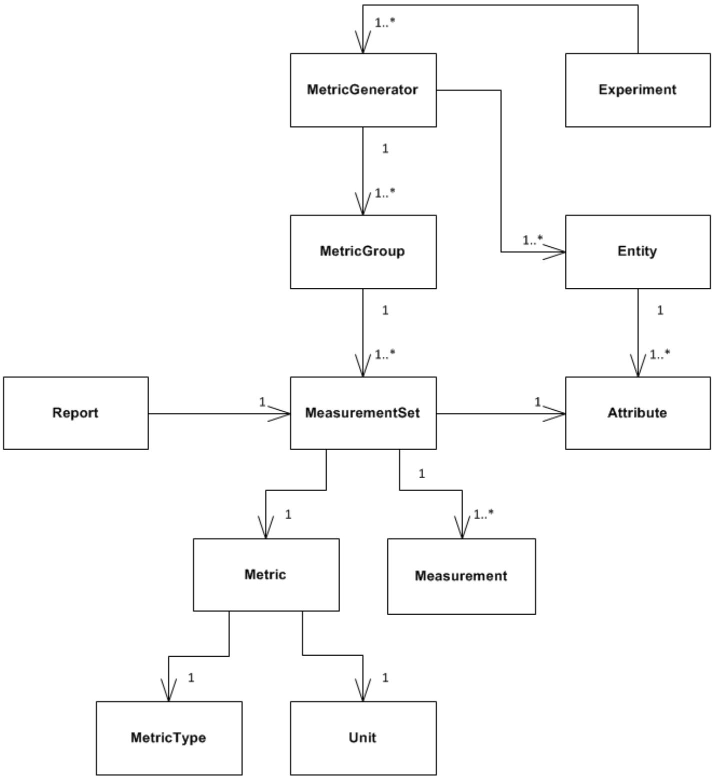


Figure 71: Logical metric model

In this model, the objects of experimental observation (referred to as ‘Entities’) are de-coupled from the agent (the EXPERImonitor software client) making the observations. *Entities* themselves must contain one or more *Attributes* that are the subject of actual instrumentation and measurement activity. We can consider the data management structures that support the collection of data representing attributes from either a ‘top-down’ perspective (starting from Metric Generators) or from a ‘bottom-up’ view point, starting with a data collection type (the Measurement Set type) that is mapped directly to an attribute of interest. Here we will take the latter approach and start by directly linking data sets to an attribute.

The *Measurement Set* type holds a set of measurements that specifically relate to an attribute and in addition has associated with it a metric meta-data indicating its Metric Type (nominal; ordinal; interval or ratio) and its Unit of measure. In the diagram above, we see two instances of Measurement Sets (each uniquely identified by a UUID value) which are mapped directly to the attributes of interest.

Moving up the data hierarchy, the next level of logical organisation is the *Metric Group* – a container used to perform one level of partitioning for collections of measurements that relate (for example, video rendering metrics). *Metric Groups* themselves are collected together by the top level data organisation, the *Metric Generator*. The *Metric Generator* represents system-level components that generate metrics, for example it may be useful to differentiate server and client based metric generators. An additional mapping, similar to that used to link measurement data sets to attributes is specified linking metric generators to entities under observation since it is likely that individual systems will be deployed to observe different entity types. EXPERImonitor client software must send their specification of the metrics they are going to provide the EXPERImonitor in this way. Using this approach, the experimenter has a means by which to understand which clients are performing what kind of measurements, and what they relate to within the experimental venue.

3D-LIVE software components formally described their metrics using this model and transmit the data, via a RabbitMQ server, to the EXPERImonitor during the experiment. Metrics reported by the various 3D-LIVE system components are described throughout this document in the related QoS metric tables. Readers interested in further details of these metrics are referred to the 3D-LIVE UX model and reports on the 3D-LIVE experiments, as described in deliverable D4.1: *Report on the experimentation and evaluations of the 3D-LIVE Tele-Immersive Environment*. Further information on the EXPERImonitor can also be found in section 5.9 of the 3D-LIVE deliverable D1.2: ‘Report on the Study and Creation of the Holistic User Experience Model’ and also a more detailed architectural discussion can be found in the EXPERIMEDIA deliverable ‘*D2.1.6 Second Blueprint Architecture*’.

4 Traceability matrix

In the following tables details are provided about the evolution of the 3D-LIVE platform along the different experimental phases.

In Table 17, the addressing of the different project requirements is depicted in terms of actions taken and prototype integration.

1.	**Project Requirement:** Create a geo-technical model of the experimentations sites respecting textural, mechanical and atmospheric properties of the real environments.
	Action: A 1 meter accurate topology of each site has been created and imported into the rendering engines. Buildings and important surrounding elements have been created and textured in order create the most consistent experience for the users. Wide area models have been generated to reinforce the immersion of players in the actual 3D scene (mountains, far fields ...etc.). The Physics properties of the terrains have been studied and set up to match the activity of the users depending on the virtual surface of interactions. (Ballistic behaviour for golf, friction of the ground, etc.)Weather effects were also included in the context of the ERS queries. (Section 2.8)
	Evaluation: The behaviour has been experimented and validated by users who did interact in a natural way with the system, without complaining about visual and physical inconsistencies during Live #3.
	Addressed in Prototype: Ski Live #1, Ski Live #3(updated), Golf Live #1, Jogging Live #1

2.	**Project Requirement:** Apply visual weather effects live in the 3D environment.
	Action: An Environment Reconstruction Service (section 2.5) was created to aggregate local weather data with two types of sensing technologies: Open Weather Map(using closest weather stations) and Sensordrone (using local sensors inputs). The service was then queried by applications and a visual effects model has been set up to generate weather changes in the game for: Clouds, light intensity, rain, snow, temperature, humidity and wind. (Section 2.8)
	Evaluation: Apart from the Golf scenario, the visual effects were matching real weather conditions. In the golfing scenario the closest OWM station was 100km away.
	Addressed in Prototype: Ski Live #1, Golf Live #1, Jogging Live #1
3.	**Project Requirement:** Allow Real time transmission of game data.
	Action: The game engines chosen for the development of the 3D-LIVE platform come with a network architecture very suitable for real time game data transmission. All the game data was then transmitted over the different clients using the standard game engine protocols (game state synchronization) via UDP communications. (Sections 2.3, 2.8 and 2.9)
	Evaluation: Game state synchronization worked fine. No inconsistency was noticed during experiments. Though, no technical evaluation on latency was performed.
	Addressed in Prototype: Ski Live #1, Golf Live #1, Jogging Live #1

4.	**Project Requirement:** Allow game rendering on different platforms (Mobile devices, Smart goggles, PCs, HMDs, CAVEs).
	Action: The Unity3D game engine has the ability to deploy on multiple platforms including mobile devices and immersive systems such as Head Mounted Displays or Cave Automatic Distributed Environments and Smart Goggles. Some game adaptions based on platform specifics were required in order to run the same content on different machines allowing tele-immersive mixed reality experience. An adaptive GUI system has been implemented in order to react to user activities in the virtual environment depending on the platform (head tracking inputs, GPS inputs, resolution and screen types, image distortion). (Section 2.8)
	Evaluation: All the platforms targeted were evaluated. The scenario had an implementation for each of the platforms running smoothly.
	Addressed in Prototype: Ski Live #3, Golf Live #2, Jogging Live #2

5.	**Project Requirement:** Integrate a set of sensory-motor interfaces to enrich in real time user experience.
	Action: All the sensory interfaces were selected to allow the transmission of wireless data at the highest possible rate. To allow natural user interactions and force feedbacks, the solution chosen was to select real equipment for users (golf clubs, ski simulator, treadmill indoors - golf clubs, ski equipment outdoors). (Section 2.1)
	Evaluation: The interactions were quite natural and provided acceptable feedbacks in real time according to the experiments.
	Addressed in Prototype: Ski Live #1, Golf Live #2, Jogging Live #1
6.	**Project Requirement:** The avatars of the outdoor players must perform appropriate animations.
	Action: Use avatar with predefined animations for smooth idle, walking, running animation transitions. (See section 2.8.1)
	Evaluation: The avatars of the outdoor players performed smooth animations during the live experiments.
	Addressed in prototype: Jogging Live #1
7.	**Project Requirement:** Comparative Statistics must be displayed after a game.
	Action: Collect activity recognition data and display performance for each runner after the experience. (See section 3.2)
	Evaluation: The users' running performance is displayed after the experience. Users compared these results after game while chatting.
	Addressed in prototype: Jogging Live #2 and updated in Jogging Live #3.

8.	**Project Requirement:** Enhance the quality of the skeleton in occluded areas.
	Action: Developed Skeleton Merging algorithm from multiple kinects (See section 3.1.2).
	Evaluation: Skeleton tracking is improved in occluded cases. Furthermore, the performance of the module was improved due to addressing R1-5 (D3.4, section 6.1)
	Addressed in Prototype: Ski Live #1
9.	**Project Requirement:** Evaluate the performance of the users' activity.
	Action: Developed Activity Recognition Algorithm for both Jogging and Skiing (See section 3.2).
	Evaluation: This requirement's development evolved through different prototypes according to user feedback. Eventually, again according to user feedback, the final prototypes integrated a robust and working solution for evaluating user's activity and estimating user's speed as necessary.
	Addressed in Prototype: Ski Live #1, Jogging Live #1
10.	**Project Requirement:** Reconstruct Humans in Real-Time.
	Action: Developed 3D-Reconstruction algorithm and integrated it in 3D-LIVE Capturer (See section 3.1.4).
	Evaluation: Positive user feedback acknowledges the value and quality of the module as well as its successful integration in 3D-LIVE applications.
	Addressed in Prototype: Jogging Live #1

11.	**Project Requirement:** Allow 3D-Reconstruction to be transmitted over the network.
	Action: - Developed 3D-Reconstruction Compression Algorithms and integrated it in 3D-LIVE Capturer. - Added Compressed 3D-Reconstruction network streaming support in 3D-LIVE Capturer - Developed 3D-Reconstruction Network receiver / decompressor (See section 3.3)
	Evaluation: Transmission/reception of reconstructions was successfully integrated in 3D-LIVE applications.
	Addressed in Prototype: Jogging Live #1
12.	**Project Requirement:** Provide evaluation metrics for 3D-Reconstruction pipeline
	Action: Added initial EXPERImonitor metrics about 3D-Reconstruction pipeline
	Evaluation: Initial evaluation metrics for 3D-Reconstruction pipeline successfully reported to EXPERImonitor.(See D4.1 Section 5.1.10)
	Addressed in Prototype: Jogging Live #1
13.	**Project Requirement:** Evaluate the quality of 3D-Reconstruction.
	Action: Developed method for quality evaluation of 3D-Reconstructions and integrated it in 3D-LIVE Capturer. (See section 3.1.4) The evaluation is reported via EXPERImonitor.
	Evaluation: The algorithm was successfully integrated in 3D-LIVE Capturer and the reconstruction evaluation metric is successfully reported to EXPERImonitor in real-time.
	Addressed in Prototype: Jogging Live #2

14.	**Project Requirement:** New EXPERImonitor requirements set from T3.6 inclusion.
	Action: Added new 3D-Reconstruction EXPERImonitor metrics in 3D-LIVE Capturer (See D4.1)
	Evaluation: The new metrics were successfully reported to EXPERImonitor .
	Addressed in Prototype: Jogging Live #3
15.	**Project Requirement:** With T3.6 a new EXPERImonitor metric “Reconstruction Receiving Framerate” was added.
	Action: Updated 3D-Reconstruction Network receiver / decompressor to report new EXPERImonitor metric.
	Evaluation: The new metric was successfully reported to EXPERImonitor .
	Addressed in Prototype: Jogging Live #3
16.	**Project Requirement:** Evaluate the visual quality of compressed 3D-Reconstructions to address the needs of newly added task T3.6.
	Action: Added evaluation of the quality of the compressed 3D-Reconstruction in 3D-LIVE Capturer (See section 3.1.4).
	Evaluation: The algorithm was successfully integrated in 3D-LIVE Capturer and the reconstruction evaluation metric is successfully reported to EXPERImonitor in real-time.
	Addressed in Prototype: Jogging Live #3

17.	**Project Requirement:** Network monitoring & Content Adaptation (according to newly added task T3.6).
	Action: Added support for Adaptive Mesh Compression in 3D-LIVE Capturer (See section 3.3).
	Evaluation: This feature although implemented for Ski Live #3, it was never put into practice during the live experiments.
	Addressed in Prototype: Ski Live #3

Table 17: 3D-LIVE Platform Evolution: Project Requirements

Finally, similar to Table 17, Table 18 below depicts the most important technical feedback across the evaluation of the different prototypes. (i.e. observations of 3D-LIVE's technical people during LIVE experiments). As before, the table depicts the new requirements that were formulated in response to this feedback, the actions taken to meet the new requirements the evaluation of the actions and the prototypes that integrated the changes required to meet the new requirements.

1.	**Technical Feedback (Ski Live #2):** 3D-LIVE capturer crashes after long continuous use.
	New Requirement: Develop a more robust capturing platform to avoid system crashes and enhance the user experience.
	Action: Fixed a memory leak in 3D-LIVE Capturer's code.
	Evaluation: The 3D-LIVE Capturer application did not suffer from the same error again.
	Addressed in Prototype: Jogging Live #1

2.	**Technical Feedback (Ski Live #1):** Users reported that sometimes their Ski score would be reported as "NaN".
	New Requirement: Increase the robustness of the activity recognition and score calculation algorithm for the ski case.
	Action: Identified and fixed the bug in ski activity recognition module. (Section 3.2)
	Evaluation: This error did not occur again.
	Addressed in Prototype: Ski Live #2
3.	**Technical Feedback (Jogging Live #1):** The requirement for all clients to disconnect from the EXPERImonitor service before each experiment is inconvenient.
	New Requirement: New EXPERImonitor experiments should be able to be performed without clients disconnecting the EXPERImonitor service.
	Action: Updated 3D-LIVE Capturer & Game Clients to EXPERImonitor API v2.0.
	Evaluation: The new API integration successfully allowed for new experiments to be conducted without disconnecting from EXPERImonitor .
	Addressed in Prototype: Jogging Live #2

4.	**Technical Feedback (Jogging Live #2):** 3D-Reconstruction visual evaluation algorithm involves manual calibration steps and contains some minor synchronization issues between Kinect and Camera Capturing.
	New Requirement: Automate the calibration process and enhance the synchronisation between Kinect & Camera.
	Action: Updated 3D-Reconstruction's Visual Quality Evaluation algorithm accordingly, fixed the capturing synchronization issues, and integrated it in 3D-LIVE Capturer. (See section 3.1.4)
	Evaluation: The calibration process was successfully automated (no more manual calibration steps involved) and sync issues were eliminated.
	Addressed in Prototype: Jogging Live #3
5.	**Technical Feedback (Jogging Live #2):** Need to have backup solutions for EXPERImonitor metric logging.
	New Requirement: Add EXPERImonitor logging capabilities to 3D-LIVE Capturer.
	Action: Update 3D-LIVE Capturer accordingly.
	Evaluation: Logging capabilities were successfully integrated in 3D-LIVE Capturer.
	Addressed in Prototype: Jogging Live #3

6.	**Technical Feedback:** In the ski scenario 3D-reconstruction was inconsistent with the subject's movements in terms of mirroring.
	New Requirement: 3D-Reconstruction needs to map the actual user orientation.
	Action: Updated Unity3D reconstruction rendering script to mirror 3D-Reconstruction accordingly.
	Evaluation: 3D-Reconstruction's rendering was made consistent with user orientation.
	Addressed in prototype: Ski Live #3
7.	**Technical Feedback (Ski Live #1):** Voice Delays in Golf and Ski applications are decreasing the framerate of game clients.
	New Requirement: Improve the Voice communications using Unity3D to cancel delays and improve framerate in Game Client.
	Action: Initially worked on multithreading of voice compression/decompression with Unity3D. Finally, Mumble was used for all use cases as it was found to be a more stable and reliable solution. (See sections 2.6, 2.8, 2.9 and 2.10)
	Evaluation: Delays were not noticeable anymore during the games. The game clients' framerates were not affected anymore. (~60FPS for indoor game client, ~30 FPS for outdoor game client).
	Addressed in Prototype: Ski Live #3

8.	**Technical Feedback (Ski Live #1):** Reconnection problems after crash / network failure.
	New Requirement: Guarantee the full recovering of the game in the case of network failure / disconnection from game server.
	Action: Implement game state saving and recovery, both server side and client side.
	Evaluation: Under poor network conditions some networks some disconnections did occur. Automatic reconnections from game clients to the game server worked flawlessly, keeping the status of the game consistent.
	Addressed in Prototype: Golf Live #2
9.	**Technical Feedback (Ski Live #1):** The skier controller script does not respect physics calculations on virtual slope.
	New Requirement: Make the virtual skier controller more accurate depending on topology of the slope.
	Action: Developed a new controller script that included ski physics while taking into account activity recognition results (sections 2.8.2 and 3.2).
	Evaluation: The time and speeds to go down any slope were closer to reality, respecting the topology of the terrain. Activity Recognition results (section 3.2) allowed users to better control their turns and speeds on the slope.
	Addressed in Prototype: Ski Live #3

10.	**Technical Feedback (Golf Live #2):** Upon indoor user trials, the balls are thrown too far away in the golf course.
	New Requirement: Rework physics to feed ball ballistic equations with adjusted values.
	Action: The coefficients in ballistic equations were adjusted to result consistent strokes via empirical tests on the golf course. (See section 2.8.2)
	Evaluation: None of the users reported inconsistent actions with respect to what they are actually able to do in real life.
	Addressed in Prototype: Golf Live #3
11.	**Technical Feedback (Ski, Golf Live #1):** Connectivity issues of outdoor game client when network coverage is poor.
	New Requirement: Ensure proper connectivity when network coverage is poor.
	Action: An attempt to make more a more robust connection process in Unity 3D was made.
	Evaluation: In case of bad network coverage, the outdoor game client disconnects from the server anyways. An automatic reconnection is in place, but it cannot connect if the network conditions are really poor.
	Addressed in Prototype: Not addressed. In case of bad network conditions, a "local game mode" might be necessary.

12.	**Technical Feedback (Ski Live #1):** The ski controller script for the outdoor user does not work correctly in high speeds.
	New Requirement: Develop a new algorithm to allow tracking the outdoor skier at high speeds.
	Action: A new algorithm was developed by merging gaming techniques to GPS and optionally IMU sensors, allowing tracking and predictions of user's path depending on topology of the slope and the activity of the skier. (See sections 2.8.2 and 2.9.2)
	Evaluation: The stripped version (without IMU) was used during experiments. It allowed a very acceptable tracking of users depending on topology of the slope. Adding the IMU was only tested and validated in the lab.
	Addressed in Prototype: Ski Live #3
13.	**Technical Feedback (Ski Live #1):** Reception of RabbitMQ data on mobile devices requires java specific implementation.
	New Requirement: Implement a Java RabbitMQ library to retrieve ERS and Skeleton data for mobile devices.
	Action: Specific RabbitMQ implementation was made for mobile devices to retrieve skeleton and ERS data on Android.
	Evaluation: Everything run smoothly**.**
	Addressed in Prototype: Ski Live #2

14.	**Technical Feedback (Golf Live #1):** Bluetooth bandwidth insufficient when using a stereo wireless headset and IMU sensors.
	New Requirement: Find a way to simultaneously use IMU Bluetooth sensors and allow voice communications with users.
	Action: A wired audio headset was deployed instead of a Bluetooth one, to save bandwidth for the IMU sensors.
	Evaluation: Everything run smoothly. The users did not complain about the wired headset.
	Addressed in Prototype: Golf Live #2
15.	**Technical Feedback (Golf Live #1):** The motion sensors EXEL S1 are drifting much after a few golf strokes.
	New Requirement: Preserve IMU sensors from drifting after strikes.
	Action: Sensors EXEL S1 were replaced by sensors EXEL S3. (See section 2.1.1)
	Evaluation: Not evaluated in Golf.
	Addressed in Prototype: Ski Live #3

16.	**Technical Feedback (Golf Live #1):** Motion sensors EXEL S1 error while using putters or ferromagnetic clubs.
	New Requirement: Control the error while using IMU sensors attached to golf clubs.
	Action: For the indoors users there was no need to do anything since the error occurred only on one unused axis. For the outdoor users only Carbon shafts were used in order to get a consistent orientation of the club on the field.
	Evaluation: The error has not been reproduced in this new configuration.
	Addressed in Prototype: Golf Live #3
17.	**Technical Feedback (Jogging Live #2):** Jogging scenario UI displays speed information incorrectly.
	New requirement: Display speed information correctly.
	Action: Calculate correct speed from GPS positions for outdoor jogger and display speed for indoor joggers calculated by 3D-LIVE Capturer. (See section 3.2.1)
	Evaluation: UI correctly handles speed calculations. Outdoor speed still somewhat erratic due to variance in GPS accuracy and location updates.
	Addressed in prototype: Jogging Live #3.

18.	**Technical Feedback (Jogging Live #1):** Incorrect in-game response to the users' physical efforts in Jogging Live #1.
	New requirement: Balance the users' physical efforts with their impact in the virtual world.
	Action: Check and calibrate real movement to match the avatar in the virtual world. Performed calibration procedure using realXtend and virtual Oulu city to match the speed and avatar movement to the real speed of the runner.
	Evaluation: Users still felt movements of the avatar are not up to par what they did in real life. This problem is further addressed in LIVE3. Slight adjustment of speed vector seems to have fixed the initial perception of sluggish avatars.
	Addressed in prototype: Jogging Live #2 and Jogging Live #3
19.	**Technical Feedback (Jogging Co-Creation):** Users do not want to wait for any configuration to happen before the run. Just equip and start running.
	New requirement: Zero configuration on start-up.
	Action: Launching the realXtend server and clients for jogging scenario automatically initializes various protocols and devices.
	Evaluation: After the loading phase, the users are immediately able to start using the application.
	Addressed in prototype: Jogging Live #1.

Table 18: 3D-LIVE Platform Evolution: Technical Feedback

5 Discussion

In this deliverable all the 3D-LIVE core technologies have extensively been described. Details about the interaction between the modules and some architectural clues have also been given, explaining the rationale behind the decisions and the suitability in the 3D-LIVE project (Section 2). In addition, low-level and in-depth details about the highly innovative technologies that are integrated in the 3D-LIVE modules have been provided. Both existing and innovative technologies were orchestrated and combined to form the first well-designed, stable and reliable mixed-reality platform that can be used as a reference for future research and development, either by the scientific community or by the cutting-edge industry.

6 Appendix A

Fuzzy inference model for Jogging:

VARIABLE: HipsAngle	Range: 0.0 – 90.0
TERM: **LOW**	Type: Ramp(20,0)
TERM: **HIGH**	Type: Trapezoid(0,20,90,90)
VARIABLE: **ArmBones**	Range: 0.0 – 180.0
TERM: **LOW**	Type: Ramp(90.0, 0.0)
TERM: **MED**	Type: Triangle(0.0, 90.0, 180.0)
TERM: **HIGH**	Type: Ramp(90.0, 180.0f)
VARIABLE: ArmSide	Range: 0.0 – 180.0
TERM: **LOW**	Type: Trapezoid(0.0, 0.0, 35.0, 80.0)
TERM: **HIGH**	Type: Trapezoid(35.0, 80.0, 180.0, 180.0)
VARIABLE: KneeAngle	Range: 0.0 – 90.0
TERM: **LOW**	Ramp(75.0, 0.0)
TERM: **HIGH**	Trapezoid(0.0, 25.0, 180.0, 180.0)
VARIABLE: LeanForw	Range: 0.0 – 180.0
TERM: **LOW**	Type: Ramp(25.0, 0.0)
TERM: **HIGH**	Type: Trapezoid(0.0, 25.0, 180.0, 180.0)

VARIABLE: JogEvaluation	Range: 0.0 – 1.0
TERM: **LOW**	Type: Ramp(1.0, 0.0)
TERM: **HIGH**	Type: Ramp(0.0, 1.0)

Table 19: Fuzzy inference model for Jogging

Fuzzy inference model for Skiing:

VARIABLE: KneeAngle	Range: 0.0 – 180.0
TERM: **LOW**	Type: Gauss (110.0, 10)
TERM: **MEDIUM**	Type: Gauss (140, 10)
TERM: **HIGH**	Type: Gauss (170, 10)
VARIABLE: Bending	Range: 0.0 – 180.0
TERM: LOW	Type: Gauss (90.0, 15.0)
TERM: **MEDIUM**	Type: Gauss (130.0, 15)
TERM: **HIGH**	Type: Gauss (170, 15)
VARIABLE: ArmSide	Range: 0.0 – 180.0
TERM: **LOW**	Type: Gauss (10.0, 10.0)
TERM: **MEDIUM**	Type: Gauss (40.0, 10.0)
TERM: **HIGH**	Type: Gauss (70,10.0)
VARIABLE: TorsoUpright	Range: 0.0 – 30.0
TERM: **LOW**	Type: Gauss (2.0, 5.0)
TERM: **HIGH**	Type: Gauss (15,5.0)
VARIABLE: LegsLean	Range: 0.0 – 20.0
TERM: **LOW**	Type: Gauss (2.0,5.0)
TERM: **HIGH**	Type: Gauss (6.0, 5.0)
VARIABLE: KneeDist	Range: 0.0 – 1.0
TERM: **LOW**	Type: Gauss (0.2, 0.6)

TERM: **HIGH**	Type: Gauss (0.5, 0.6)
VARIABLE: EvalSki	Range: 0.0 – 1.0
TERM: **LOW**	Type: Gauss (0.1, 0.1)
TERM: **MEDIUM**	Type: Gauss (0.5, 0.1)
TERM: **HIGH**	Type: Gauss (0.9, 0.1)

Table 20: Fuzzy inference model for Skiing

The Ramp, Triangle and Trapezoid functions are depicted in the figures below. The Gauss function represents a Gaussian distribution with Gauss(m,s) having mean m and standard deviation s.

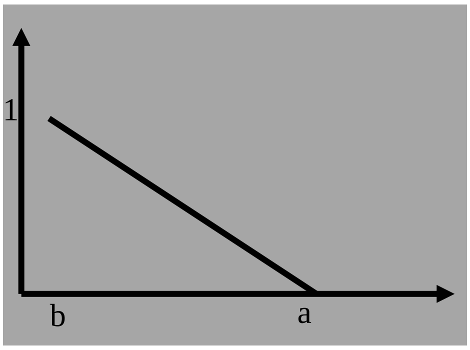


Figure 74: Ramp(a,b) function with a > b

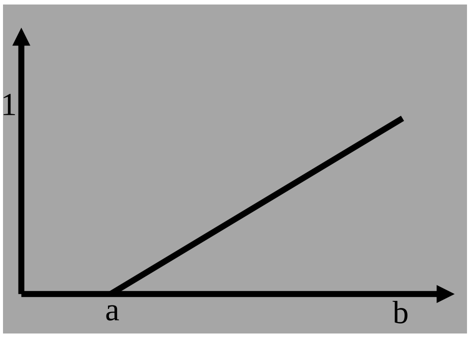


Figure 75: Ramp(a,b) function with a < b:

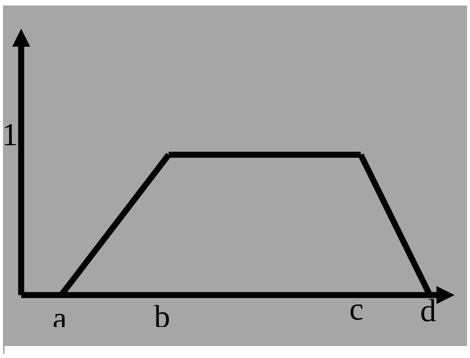


Figure 73: Trapezoid(a,b,c,d) function

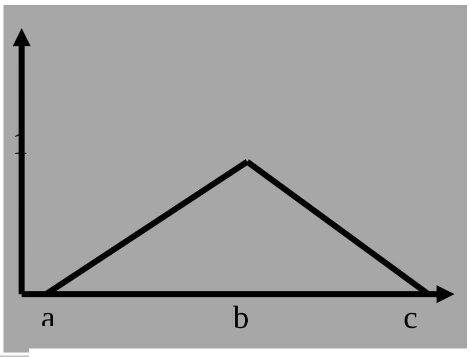
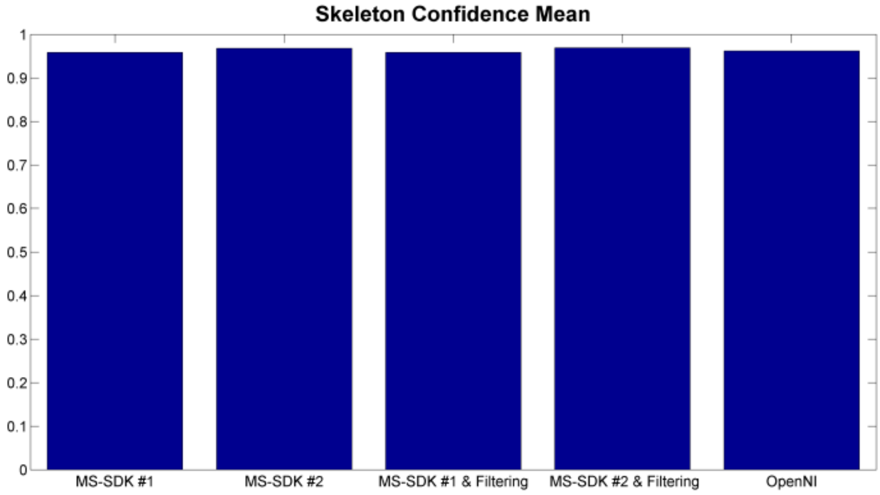
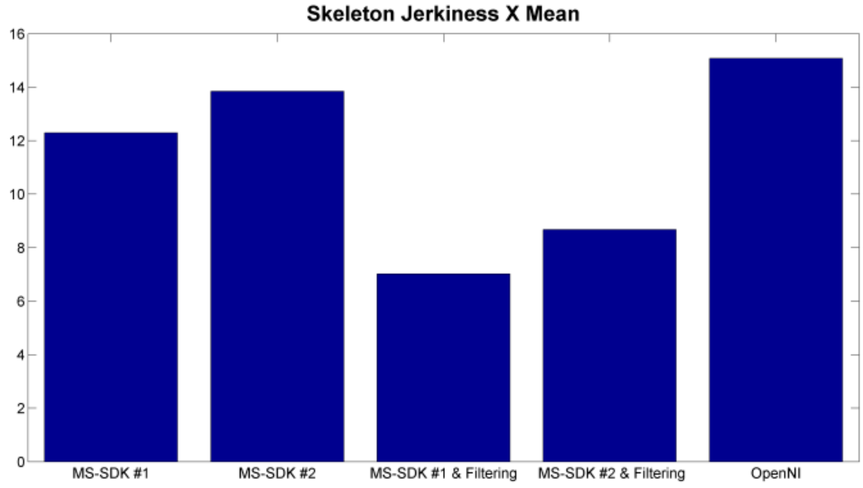


Figure 72: Triangle(a,b,c) function

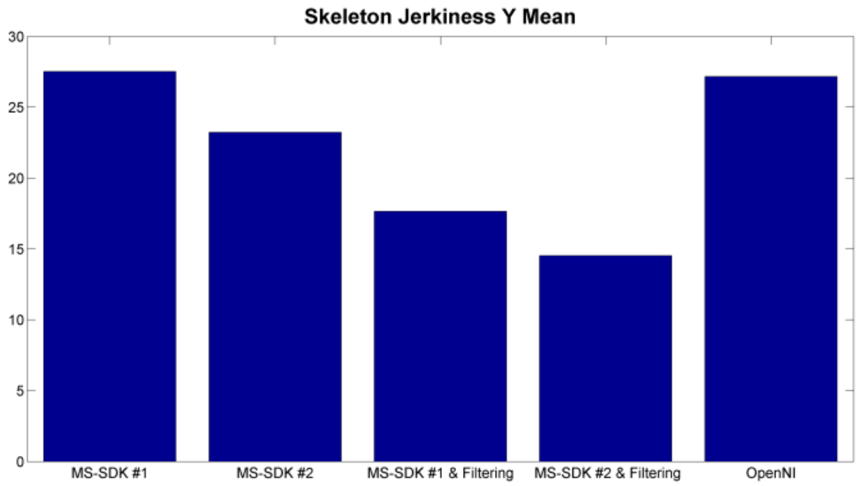
7 Appendix B

Following section 2.7, in this appendix we present the evaluation results of the two SDKs (MS SDK vs OpenNI), through EXPERImonitor metric analysis on skeletal data. The definitions of the relevant EXPERImonitor metrics are provided in the relevant Table 5. The evaluation was performed in 3 different datasets. Two datasets (namely MS-SDK #1 and MS-SDK #2) were used for Microsoft SDK evaluation and another one was used for evaluating OpenNI. On the Microsoft SDK datasets we have also evaluated the skeleton filters reported in section 3.1.1. The results of the EXPERImonitor metric analysis is shown as bar graphs in Figure 76.

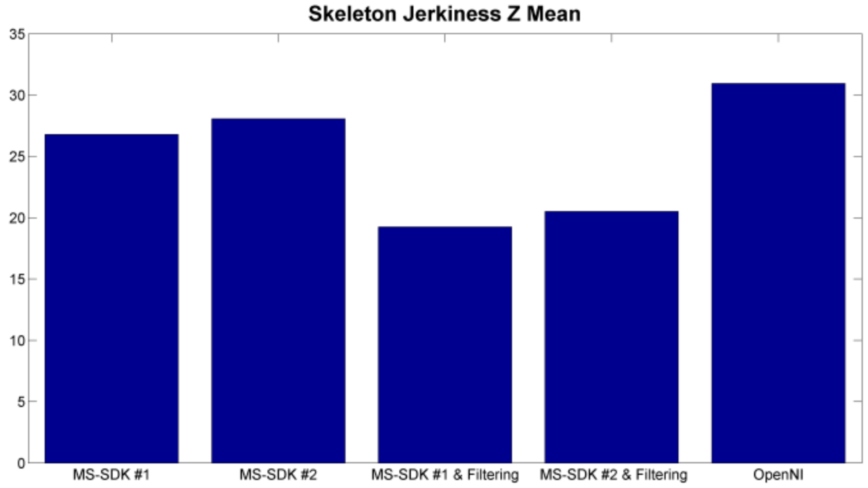




Skeleton Jerkiness X Mean
0
2
4
6
8
10
12
14
16
MS-SDK #1
MS-SDK #2
MS-SDK #1 & Filtering
MS-SDK #2 & Filtering
OpenNI



Skeleton Jerkiness Y Mean
0
5
10
15
20
25
30
MS-SDK #1
MS-SDK #2
MS-SDK #1 & Filtering
MS-SDK #2 & Filtering
OpenNI



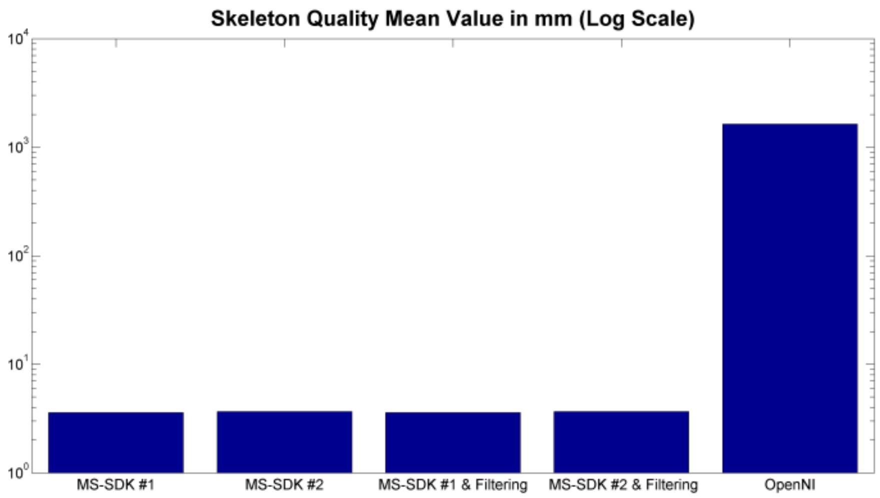


Figure 76: EXPERImonitor metric analysis on skeletal data between Micrososft SDK & OpenNI.

As shown in Figure 76 above, for the skeleton confidence metric, both APIs provide similar performance of negligible difference. However, for the rest of the metrics (Skeleton Jerkiness and Skeleton Quality) the MS-SDK, especially when enhanced with the skeleton filters, provides lower (thus better) metric values resulting into improved performance. Specifically, for the Skeleton Quality metric the improvement gained from the use of MS-SDK is significant and orders of magnitude better. In conclusion, this metric analysis supports that the replacement of OpenNI with MS-SDK provides skeletal information of higher accuracy, especially when combined with the CERTH proposed skeleton filters, thus the requirement of the users' for better avatar animation has been fulfilled.

8 References

[Mamdani, 1975] E. H. Mamdani, and S. Assilian. "An experiment in linguistic synthesis with a fuzzy logic controller." *International Journal of Man-Machine studies* 7, no. 1 (1975): 1-13.

[Chipperfield, 1995] A. J. Chipperfield and P. J,. Fleming. "The MATLAB genetic algorithm toolbox." In *Applied Control Techniques Using MATLAB, IEE Colloquium on*, pp. 10-1. IET, 1995.

[Asteriadis, 2013] S. Asteriadis, A. Chatzitofis, D. Zarpalas, D. Alexiadis, P. Daras, "Estimating human motion from multiple Kinect Sensors", Mirage 2013, 6th International Conference on Computer Vision / Computer Graphics Collaboration Techniques and Applications, June 6–7, 2013, Berlin, Germany

[Vogiatzis, 2007] G. Vogiatzis, C. Hernandez, P. Torr, and R. Cipolla, “Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency,” IEEE Trans. Pattern Anal Mach. Intell., vol. 29(12), pp. 2241 –2246, 2007.

[Furukawa, 2009] Y. Furukawa and J. Ponce, “Carved visual hulls for image-based modeling,” Int. Journal of Computer Vision, vol. 81, p. 5367, 2009.

[Franco, 2009] J.-S. Franco and E. Boyer, “Efficient polyhedral modeling from silhouettes,” IEEE Trans. Patt. Anal Mach. Intell., vol. 31, pp. 414–427, 2009

[Curless, 1996] B. Curless and M. Levoy, “A volumetric method for building complex models from range images,” in Proc. SIGGRAPH, 1996

[Vasudevan, 2011] R. Vasudevan, G. Kurillo, E. Lobaton, T. Bernardin, O. Kreylos, R. Ba-jcsy, and K. Nahrstedt, “High quality visualization for geographically distributed 3D teleimmersive applications,” IEEE Transactions on Mul- timedia, vol. 13(3), June 2011.

[Maimone, 2011] A. Maimone and H. Fuchs, “Encumbrance-free telepresence system with real-time 3D capture and display using commodity depth cameras,” in 10th IEEE ISMAR 2011

[Alexiadis, 2013] D. S. Alexiadis, D. Zarpalas, and P. Daras, “Real-time, full 3-D

reconstruction of moving foreground objects from multiple consumer depth cameras,” IEEE Trans. on

Multimedia, vol. 15, pp. 339–358, 2013

[Alexiadis, 2014] D. S. Alexiadis, D. Zarpalas, P. Daras, "Fast and smooth 3D reconstruction using multiple RGB-Depth sensors", IEEE International Conference on Visual Communications and Image Processing, VCIP 2014, Dec 7-10, Valletta, Malta

[Herrera, 2012] D. Herrera, J. Kannala, and J. Heikkila, "Joint depth and color camera

calibration with distortion correction," IEEE Trans. Pattern Anal Mach. Intell., vol. 34, 2012

[Kazhdan, 2005]M. Kazhdan, "Reconstruction of solid models from oriented point sets,"

In Proc. 3rd Eurographics symposium on Geometry processing, 2005

[Doumanoglou 2014 (A)], 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, Apr 2014

[Doumanoglou 2014 (B)], , A. Doumanoglou D. Alexiadis, S. Asteriadis, D. Zarpalas, P. Daras, On human Time-Varying Mesh compression exploiting activity-related characteristics, 4-9 May 2014, Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on.

[Alexiadis 2014], D. Alexiadis, A. Doumanoglou, D. Zarpalas, P. Daras, "A case study for tele-immersion communication applications: from 3D capturing to rendering", IEEE International Conference on Visual Communications and Image Processing, VCIP 2014, Dec 7-10, Valletta, Malta

[K. Mamou 2009], K. Mamou, T. B. Zaharia, and F. J. Prêteux, "TFAN: A low complexity

3D mesh compression algorithm," Journal of Visualization and Computer

Animation, vol. 20, no. 2-3, pp. 343–354, 2009.