2.2 Interfaces

2.2.1 Interfaces at the Source Area

2.2.1.1 Ambulance Emergency Network

In the Ambulance Emergency Network all the acquired data are collected at the Proxy Server. A Local Area Network (LAN) is established at the ambulance. This network can be exclusively wired or it can also have wireless parts.
In particular the cameras deployed to record ambient videos are connected through coaxial cables to the IMP box. The IMP box can then send digital videos to the Proxy Server using the LAN through an Ethernet cable or through a Wi-Fi access point. For the medical data, instead, the ultrasound machine is connected with a VGA cable to the TVONE box; then a coaxial cable connects this box to the VITEC box which is connected in its turn to the Proxy Server with an Ethernet cable.
The Proxy Server is then relied to a router with a dongle LTE and can then access a commercial 4G network to transmit to the hospital.
A VPN is established between the ambulance and the hospital across the LTE network and it is used for both data transmissions, realized using RTP protocol, and for cross-layer signalling transmissions, realized through the DDE system (using the interfaces described in section 2.2.3.2).

2.2.1.2 Body Area Network

In the Body Area Network the deployed sensors transmit the acquired data to the smartphone, which acts as the aggregation point of the network, through Bluetooth. The customized smartphone has several interfaces and can transmit the data to the hospital through its Wi-Fi interface or the 3G interface, according to the available networks.
In addition the smartphone has a DDE client implemented and exchange cross-layer signalling information through the DDE system and the NIS. More details are provided 2.2.3.

2.2.2 Interfaces at the Hospital side

2.2.2.1 Coordination Centre

The full demonstrator is shown in Figure 14. The building blocks and media flows at the Coordination Centre are highlighted with the blue ellipse on the right side of the figure.

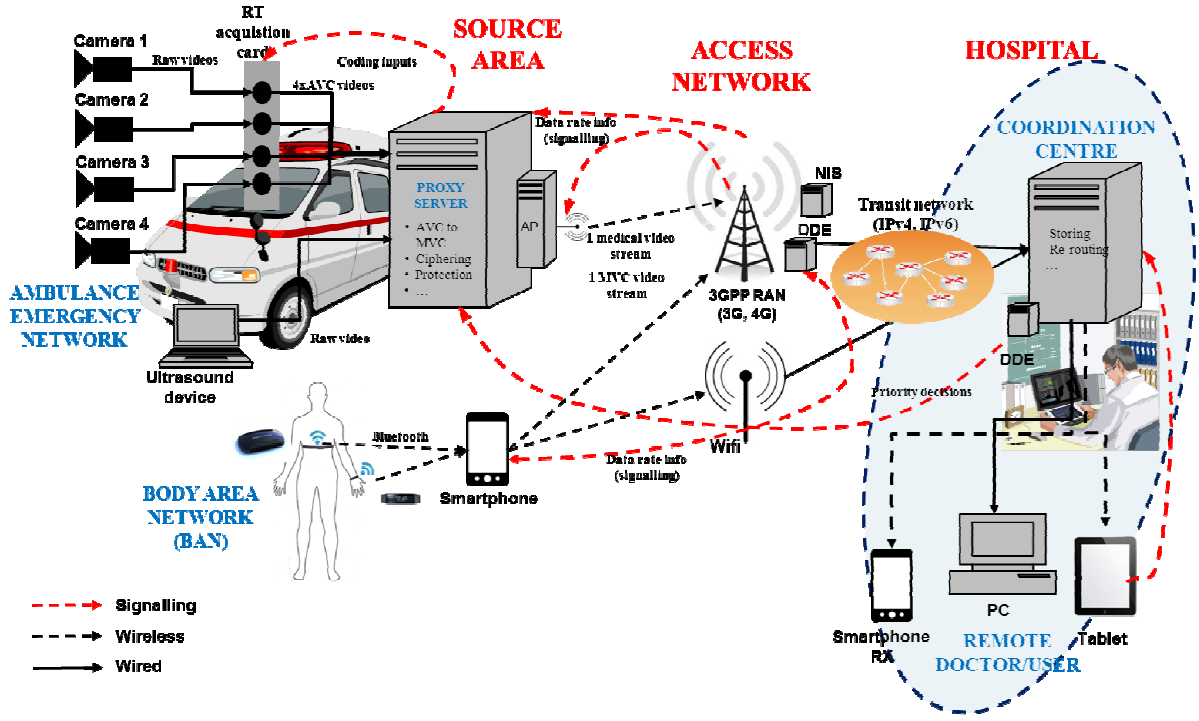

Figure 14: Building blocks and interfaces at coordination centre

The building blocks and the respective interfaces at the coordination centre are further detailed in Figure 15.

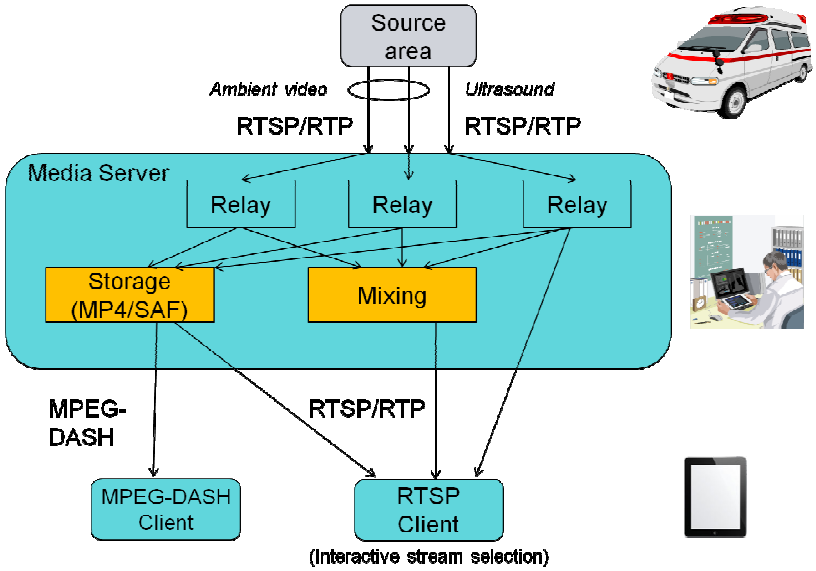

Figure 15: Building blocks and interfaces at coordination centre

Input media streams are received from the Source Srea (i.e., the Ambulance Emergency Network) as RTSP/RTP streams. The input media streams can be either ambient video feeds or ultrasound medical video streams. The video streams (both, ambient and ultrasound) are coded using the H.264/AVC format. In order to allow multiple usages of the video streams and avoid multiple transmissions of the same stream, RTSP/RTP relays are used at the input side of the Media Server.

All live input streams are mixed to achieve a stitched output stream. The stitched output video is presented as an RTSP/RTP stream to the clients. The video is compressed in the H.264/AVC or the HEVC format. The RTSP client receives the compound RTSP/RTP stream. One of the video streams can be interactively selected at the client. In this case, the selected video stream is received and displayed in addition to the stitched stream.

Further, all input video streams are stored in the MP4/SAF format. The single streams can be retrieved from the storage either as RTSP/RTP or MPEG-DASH stream. The video content is compressed in the H.264/AVC format. Note that interactive stream selection is not possible for stored content.

As stated above, two codecs are employed in the demonstrator, H.264/AVC and HEVC.

The configuration interface for H.264/AVC encoding is (simplified):

```
##############################################################################
# Encoder Control
##############################################################################
FrameWidth        = 672   # Image width in Pels, must be multiple of 16
FrameHeight       = 512   # Image height in Pels, must be multiple of 16

ReferenceFrames   = 1     # Number of previous frames used for inter motion search (1-5)
EntropyCoding     = 0     # CAVLC/CABAC (0=CAVLC, 1=CABAC)
RateControlModel  = 1     # Rate control (0=disable, 1=enable)
TargetBitRate     = 1280000 # Target bit rate
InitQP            = 28    # Initial quantitative parameter
SourceFrameRate   = 10    # Source frame rate
TargetFrameRate   = 10    # Target frame rate
TargetNALUSize    = 4000  # Target size of NAL units
```

The configuration interface for HEVC encoding is (simplified):

```
##############################################################################
# Encoder Control
##############################################################################
FrameSize             = 720p
PacketType            = 1 # Raw/AnnexB/RTP (0=Raw, 1=AnnexB, 2=RTP)
RTPPacketAggregation  = 0 # RTP Packet Aggregation (0=Disable, 1=SingleTimeAggr, 2=MultiTimeAggr)
```

```
MaxRTPPacketSize      = 1420      # Max size of a whole RTP packet, only valid when RTP is enabled
TargetNALUSize          = 80000      # Target size of NAL units

############################################################################
# Advanced Encoder Control
############################################################################
Log2MaxCUSize         = 6 # Log2 of maximum coding unit size in pixel
MaxPartDepth          = 2 # Maximum coding unit depth ( >=0, Log2MaxCUSize-MaxPartDepth >= 3 )
TULog2MaxSize         = 5 # Log2 of maximum transform size for quadtree-based TU coding (2...5)
TULog2MinSize         = 2 # Log2 of minimum transform size (must be < Log2MaxCUSize - MaxPartDepth)
InitQP                = 32 # Initial quantitative parameter (0 .. 51)
DeltaQPChroma         = 0 # value added to InitQP for Chroma quantization

############################################################################################
# Encoder Rate Control
############################################################################################
TargetFrameRate         = 25        # Target frame rate
RateControl             = 1         # Rate control (0=disable, 1=enable)
BitRateMax              = 500000    # upper bit rate boundary, [Kbps]
BitRateQPmin            = 10        # Minimum QP (may override lower boundary)
```

2.2.2.2 Real-time quality meter

The real time quality meter is connected with the media server at the Coordination Centre and receives as inputs 2D/3D video (multimedia traffic), information on application type and categorical scale (configuration) and lower layers feedback through cross-layer signaling. The outputs are the user opinion scores that can be locally stored or used to produce events for the DDE in order to adapt the transmission.

2.2.2.3 Interfaces for 3D medical data storage and encoding

For 3D medical data storage and encoding a web-based user interface and a database interface for 3D medical data files have been implemented at the Coordination Centre (Figure 16).

Figure 16: Web interface of the 3D medical data storage and encoding

2.2.3 Cross-layer signalling interfaces

The demonstrator includes three components for delivering and retrieving cross-layer information, namely DDE for the event dissemination, the network information service (NIS) for the retrieval of information about networks and base stations/access points, commonly referred to Point-of-Attachments (PoAs), and a special, socket-based API designed for controlling network layer mobility execution, based on application layer decisions.

2.2.3.1 NIS interface

NIS runs a TCP server for information queries sent over the IPv6 protocol. The inquirer sends its current location in the message and NIS discovers the available networks and PoAs nearby the mobile device.

The query message format is: "*MS_LOCATION(latitude,longitude)*".

NIS returns information encoded in JSON and the message structure is given in Table 2. The networks and PoAs inside the "DWithin" attribute relate to PoAs that do not cover mobile device's current location but are in the vicinity. Information inside the "Within" attribute lists the networks and PoAs that cover mobile device's current location.

Table 2: NIS information message format

```
{
"DWithin":{
   "[cell_id]":{
                  "cost":"[value]:[currency]:[time unit]",
                  "channel":"[channel number]",
                  "poa_location":"WGS84([lat] [lon])",
                  "cell_id":"[cell id]",
                  "distance_to_bs":"[distance in meters]",
                  "mac_address":"[MAC address]",
                  "network_name":"[network name]",
                  "frequency_band":"[band]",
                  "ip_address":"[IP address]",
                  "ssid":"[SSID in case of WLAN]",
                  "poa_area_sqm":"[PoA coverage in m2]",
                  "access_technology":"[access technology name]"
   }
},
"Within":{
   "[cell_id]":{
                  "cost":"[value]:[currency]:[time unit]",
                  "channel":"[channel number]",
                  "poa_location":"WGS84([lat] [lon])",
                  "cell_id":"[cell id]",
                  "distance_to_bs":"[distance in meters]",
                  "mac_address":"[MAC address]",
                  "network_name":"[network name]",
                  "frequency_band":"[band]",
                  "ip_address":"[IP address]",
                  "ssid":"[SSID in case of WLAN]",
                  "poa_area_sqm":"[PoA coverage in m2]",
                  "access_technology":"[access technology name]"
  }
}
}
```

2.2.3.2 DDE interface

DDE is based on a publish-subscribe type of event delivery. The event exchange happens over the TCP transport protocol. The messages are encoded in External Data Representation (XDR) [7]. Message structures for different messages are listed in Table 3.

Table 3: DDE message structures

Event: [message type][producer ID][event ID][Event type][Time-to-live][payload length][payload][digit. signature] **Registration:** [message type][producer ID][producer ID][event IF][event name] **Subscription:** [message type][consumer ID][cons. name][cons. addr.][event ID][type filter][producer filter][digit. sign.]

2.2.3.3 Cross-layer mobility management interface

The decision performed by the AHP algorithm triggers flow mobility events in the network layer. The cross-layer triggering itself is achieved by the TCP socket-based API of MIP6D-NG. Using this API the decision module is able to control MIP6D-NG modules. From the mobility management point of view, Flow Register and Flow Update API commands are to be introduced: these API messages make the decision scheme able to register application flows using their five-tuple IDs, bind them to interfaces (i.e., to Wi-Fi or 3G/4G), and also to adaptively modify these bonds in case of need, meaning the movement of flows between available network interfaces. The API messages for fine-grained (i.e., flow level) cross-layer mobility management are introduced by Figure 17 and Figure 18.

Module Name (20 byte)	**Command** (2 byte)	**Seq. No.** (1 byte)	**Source Address** (16 byte)	**Source prefix** (1 byte)	**Source port** (2 byte)
Destination Address (16 byte)	**Dest. prefix** (1 byte)	**Dest. port** (2 byte)	**Protocol** (4 byte)	**BID** (2 byte)	**Key** (16 byte)

Figure 17: Flow Register API command

Module Name (20 byte)	**Command** (2 byte)	**Seq. No.** (1 byte)	**BID** (2 byte)	**Flow ID** (4 byte)

Figure 18: Flow Update API command

The Flow Register command starts with a module name: the MIP6D-NG module responsible for flow mobility to be called by the API is defined here. Important fields of this API command are the destination and source addresses, prefixes and ports and the higher layer protocol (TCP, UDP, etc.), which together define the flow's five-tuple ID. The Binding Unique Identifier (BID) is for selecting the interface (Wi-Fi, 3G, 4G, etc.) for the flow binding procedure. In this way we can bind an application flow to a selected network interface.
The Flow Update command simply updates a defined application flow by defining a unique, inner flow identification number (Flow ID) and the BID.

2.3 Functionalities

In this section the most significant and innovative functionalities of the CONCERTO multimedia platform are described. These functionalities have been studied and validated analytically and/or through dedicated simulations in the scope of the CONCERTO technical work packages (WP2 to WP5) before to be implemented in the demonstrator.

2.3.1 Network information service

NIS provides the intelligent mobile device information about networks and PoAs in range and nearby the current location of the device. This is illustrated in Figure 19. NIS is located in order that the mobile device can reach it over the network connection it is using. In practice, the NIS server can reside, for example, in mobile network operator's core network used by the mobile device. In this case, NIS provides information about the networks and PoAs the mobile operator operates and the networks the operator has roaming agreements with.

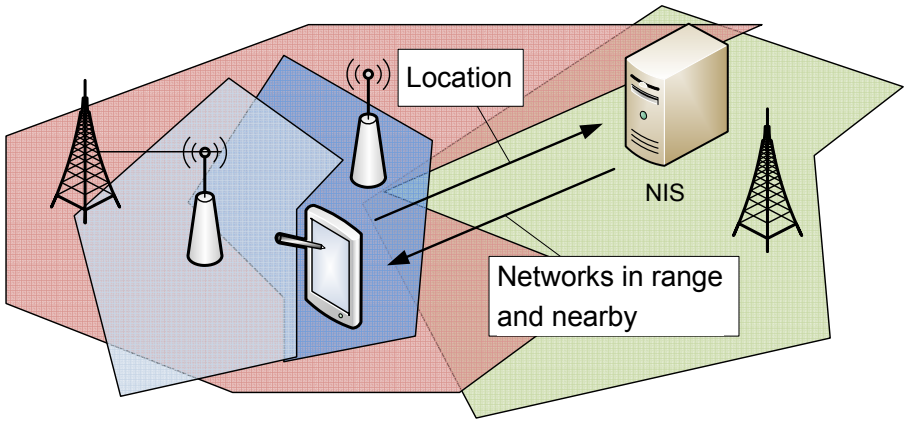

Figure 19: NIS in the demonstrator

In the demonstrator, the information about four networks and PoAs will be pre-stored in the NIS server. In addition, because it is not possible to determine the coverage areas of the PoAs with signal strength measurements, the coverage area polygons are randomly generated according to the PoA range and PoA location given. The information stored in the NIS server is composed of information given in Table 4. NIS is validated in the integrated demonstration scenario and is used by the intelligent mobile device of the Body Area Network.

Table 4: NIS parameterisation

```
Networks:
    Network1:
        network_name: concerto_wlan
        PoA1:
            access_technology: IEEE802_11n
            SSID: concerto1
            channel: 10
            frequency_band: 2_4GHz
            cost: 0:EUR:DAY
            poa_location: 43.077878,12.354725
            mac_address: 00:11:22:33:44:55
            ip_address:
            cell_id: 00:11:22:33:44:55
            range: 30
        PoA2:
            access_technology: IEEE802_11g
            SSID: concerto2
            channel: 6
            frequency_band: 2_4GHz
            cost: 0:EUR:DAY
            poa_location: 43.077856,12.354819
            mac_address: 00:12:23:34:45:56
```

```
            ip_address:
            cell_id: 00:12:23:34:45:56
            range: 40
    Network2:
        network_name: commercial_cellular_network
        PoA1:
            access_technology: UMTS_HSPA
            channel: 10
            frequency_band: 1_8GHz
            cost: 0.2:EUR:MB
            poa_location: 43.079078,12.354622
            mac_address: 00:21:32:43:54:75
            ip_address:
            cell_id: HSPA_1234
            range: 1600
        PoA2:
            access_technology: LTE
            channel: 10
            frequency_band: 1_9GHz
            cost: 0.3:EUR:MB
            poa_location: 43.078678,12.356375
            mac_address: 00:21:32:43:54:75
            ip_address:
            cell_id: LTE_1234
            range: 1000
```

2.3.2 Distributed Decision Engine

DDE EventCache is deployed in the coordination centre and in the WLAN access network. Moreover DDE clients are implemented in the ambulance, at the smartphone of the BAN and at the doctor's mobile device, and in the hospital. DDE provides flexible event-based information dissemination between different network nodes and layers regardless of their location in the network. In the demonstrator, the events listed in Table 5 are used.
DDE is validated in the integrated demonstration with the CC and intelligent terminal.

Table 5: DDE events and their respective DDE EventCache in the demonstrator

DDE Event Name	Event ID	Producer	DDE position	Consumer	Optimization
Final user preference list	30	Receiving Terminal	CC	CC	Camera selection
				Sender Terminal (multimedia source)	Rate adaptation algorithm (Master Application Controller)
				Aggregation unit on ambulance	Prioritization of video flows
Target source rate list	41	CC	CC	LTE eNodeB	LTE scheduling and RRA
				Sender Terminal (multimedia source)	Rate adaptation algorithm (Master Application Controller)
				Aggregation unit on ambulance	Multi-video rate adaptation and aggregation
WLAN AP	10	WLAN probe	WLAN	Sender	Rate adaptation algorithm

parameter list			network	Terminal (multimedia source)	(Master Application Controller) Mobility management of mobile device
				Aggregation unit on ambulance	Multi-video rate adaptation and aggregation

2.3.3 WLAN probe

The WLAN probe is deployed in the WLAN network of the demonstrator. The probe collects statistics from the WLAN access points (two in the demonstrator) and provides this information to the mobile device, capable of making intelligent handovers, over DDE. The WLAN probe is validated in the context of intelligent mobility in the integrated demonstration.

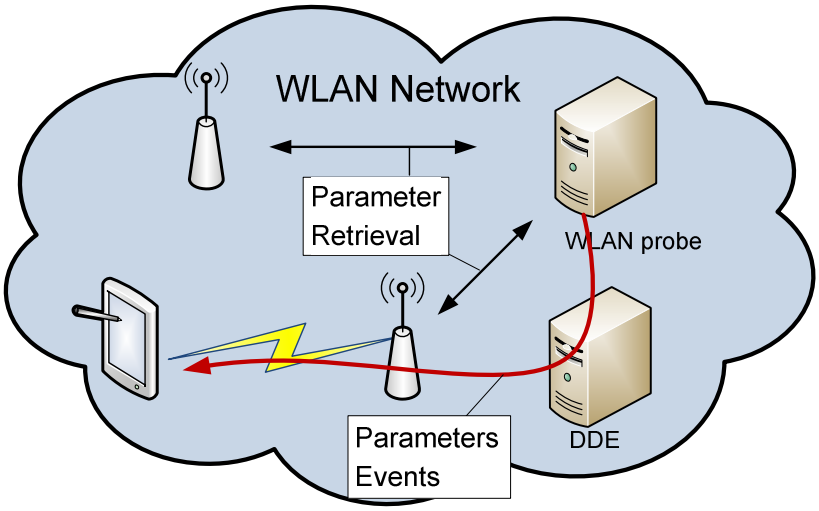

Figure 20: WLAN probe in the WLAN network

The WLAN probe collects the following parameters from the access points:

- IfInOctets: Transmitted bytes in the uplink
- IfInDiscards: Number of discarded packets (e.g. due to queue overflow) in the uplink
- ifInErrors: Number of erroneously received packets in the uplink
- IfOutOctets: Transmitted bytes in the downlink
- IfOutDiscards: Number of discarded packets (e.g. due to queue overflow) in the downlink
- IfOutErrors: Number of erroneously received packets in the downlink
- ifInUcastPkts: Number of unicast packets in the uplink
- ifInNUcastPkts: Number of non-unicast packets in the uplink
- ifOutUcastPkts: Number of unicast packets in the downlink
- ifOutNUcastPkts: Number of non-unicast packets in the downlink

The payload of a DDE event generated by the WLAN probe is encoded as follows:

"[wireless_interface_speed]:[number_of_active_clients_in_access_point]:[uplink_bytes_transmitted]:[uplink_num_discarded_pkts]:[uplink_num_error_pkts]:[uplink_packet_error_rate]:[downlink_bytes_transmitted]:[downlink_num_discard_pkts]:[downlink_num_error_pkts]:[downlink_packet_error_rate]"

The parameters are aggregated values from all active users. Each parameter is calculated from the period between sequential events. WLAN probe sends an event every second.

2.3.4 Real-time quality evaluation

Thanks to the developed real-time quality meter, the CONCERTO platform is capable to measure the 2D/3D video quality perceived by users in real-time.
The quality meter is developed as an Android application with a graphical interface, enabling to evaluate the quality in a categorical scale (1 to 5) using a slider. Figure 21 shows a snapshot of the developed Android App.

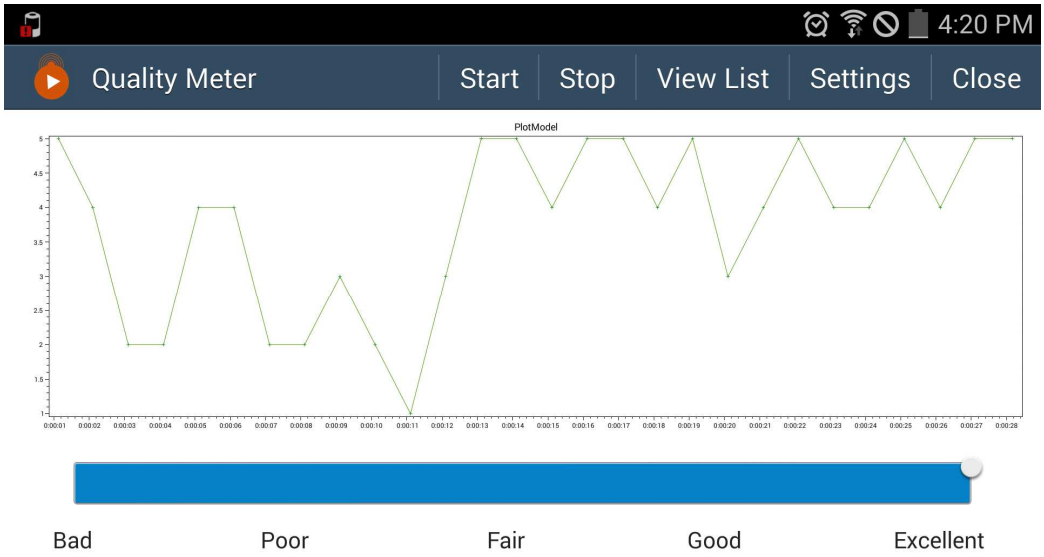

Figure 21 Graphical user interface of the real-time quality meter

The Opinion Scores (OS) from the users can be stored locally or transmitted to a remote server. In addition, status messages can be generated (e.g., if the quality of the video drops below MOS level 2, a FLAG can be generated) to provide feedback to the DDE, which could take appropriate actions to improve the perceived quality of streamed video.
The results stored locally in a database can be further analysed (e.g., for the comparison of candidate objective quality measures).
The video application can be one of the following:

- LIVE 2D/3D video streaming (streaming from IP cameras)
- LIVE Internet streaming (RTP/UDP/RTSP streaming)
- VoD streaming over Internet
- Playback from a locally stored file

The use of the developed real-time quality meter application is very important for the validation of the CONCERTO solution, since it enables subjective (and objective) quality evaluation over time for the proposed and the benchmark solutions. The application has been initially validated independently from the rest of the platform. Further tests have been realised during EuCNC conference in June 2014 in Bologna, where a significant part of the CONCERTO multimedia platform was deployed. Finally, it has been used to collect opinions from medical doctors during the CONCERTO demonstrator validation session organized in September 2014 at the hospital of Perugia.

2.3.5 Video encoding and decoding

For video encoding and decoding, H.264/AVC Baseline profile and HEVC Main profile are employed. The implementations work compliant with the respective specification. For the HEVC encoder, algorithms for low-complexity encoding, hierarchical temporal prediction and rate control are developed and implemented within the demonstrator development, see Annex 1, Annex 2, and Annex 3, respectively.

2.3.6 Video adaptation and selection

The acquisition and transmission of the videos from a remote point (i.e., the ambulance) to the coordination centre of the hospital is not realized statically, but can be adapted according to the channel conditions and user preferences. The control of the compression rate and the selection of the flows to send are realized by the application controller. This component bases its decision on the inputs received from the network about the conditions of the radio channel and from the hospital on doctors' preferences. This information is transmitted through the DDE. Full description of the

application controller behaviour and on the different adaptation strategies that can be applied are provided in Deliverables D4.2 [11] and D4.3 [12].
In addition to video adaptation and selection performed at the remote point (e.g. ambulance), adaptation can be performed on the link between the coordination centre and the mobile user (e.g., the doctor) when dynamic adaptive streaming over HTTP is used. This is implemented into the demonstrator in addition to RTP-based streaming as described earlier in this deliverable. HTTP-based streaming is useful for the later usage such as verifying, reporting and archiving. For the demonstrator, it is implemented for near-real-time viewing using a generic MPEG-DASH client. Adaptability is achieved by generating different versions of the same content with different parameters (bitrate, resolution, ...) and a manifest file (MPD) which describes the version available. The end-user client software (e.g. a tablet used by the consulting expert/doctor at the hospital) then selects dynamically the version of the content according to the link capacity between the coordination centre's database and the client device.

2.3.7 3D medical data storage and encoding

To deal with 3D medical data the following functionalities have been implemented: viewer for 2D Medical data (1 slice), montage-viewer for set of 2D medical data, free point of view volume renderer for 3D medical data, transfer function changeability for volume renderer, efficient 3D medical data storage with JPEG2000 and H264 video coding, 3D RAW data encoder and decoder.

2.3.8 Cryptography

In CONCERTO use cases, medical and context video sequences, containing sensitive data, need to be transmitted. To satisfy the requirements of the identified healthcare use cases, secure transmissions has then to be guaranteed in particular for medical video streams, but also for context video streams. The developed algorithm presented in D3.3 (Annex A) [16], proposes a selective encryption scheme that allows secure transmission of video streams - such as H.264/AVC video streams - while keeping complete compatibility with the corresponding video standard. Hence, this algorithm allows any standard-compliant decoder to decode the cyphered stream, albeit incorrectly from the visual point of view. In the CONCERTO demonstrator the algorithm has been implemented to guarantee cyphered communications from the Ambulance Emergency Network to the Hospital.

2.3.9 Video protection

To ensure quality at the receiver side, we have proposed Application-Layer Forward Error Correction (AL-FEC) to protect streams against erasure through the transmission (LTE, Wi-Fi...) thanks to the Proxy Server, which can adapt the overall bitrate by adding dynamically redundancy, and thanks to the Coordination Centre which can recover loss packets by using redundancy.
In the demonstrator, we have implemented at the Proxy Server deployed inside the ambulance, the Application Controller functionality (see Section 2.3.6) which adds redundancy at the application layer, by computing redundancy packets with PL-FEC module (presented in D4.2 [12] and in D4.3 [13]).
At the hospital side, the Coordination Centre will interpret redundancy packets to recover packets lost during the transmission with the PL-FEC codes and re-route corrected video streams to the different devices.

2.3.10 Adaptive flow mobility management

Figure 22 presents the operation of our cross-layer optimized adaptive, client-based flow mobility management scheme incorporating the decision algorithm. The most important input parameters are the actual measurement data, the static information obtained during the network measurements in the currently used networks, and the user preferences.
The first step of the algorithm is checking the available wireless access networks. The default interface is the 3G access, therefore the system registers data flows to the 3G interface using cross-layer communication between the application and the network layers. After this step and if there is at least one available Wi-Fi network, the algorithm starts the phase of passive measurements of Wi-Fi networks. If there are no available WLANs, the algorithm holds the flows on the 3G interface and waits for the appearance of new Wi-Fi access points. Otherwise, it starts the cross-layer measurements and DDE-based network information collection, in which it gathers the signal strength from link-layer, and packet loss, RTT and jitter from network layer, WLAN access point information and other relevant data from the NIS. If the decision engine does not find the parameters of the current network suitable for the application flow's QoS profile, the scheme starts to evaluate the next available network. If the measured QoS values are appropriate, the MN connects to this Wi-Fi network and moves the corresponding flows to the Wi-Fi interface based on the flows QoS profiles. After that, the application waits for a specific time to avoid the ping-pong effect. (Note, that in situations where the MN moves around the border of wireless accesses, a series of unnecessary handovers may occur during a very short time, such creating the so-called ping-pong effect.) Cross-layer communication mechanism allows us to trigger and execute flow updates in a different layer of the stack which further increases the efficiency of our system. In each case when MIP6D-NG executes a flow registration or update, it sends a Flow Binding Update (FBU) message to Home Agent, who sends back a Flow

Binding Acknowledgement (FBA) message. The third and last part of the Java layer application is the Sensor Data Aggregator which produces streams of high definition videos and different body sensor data.

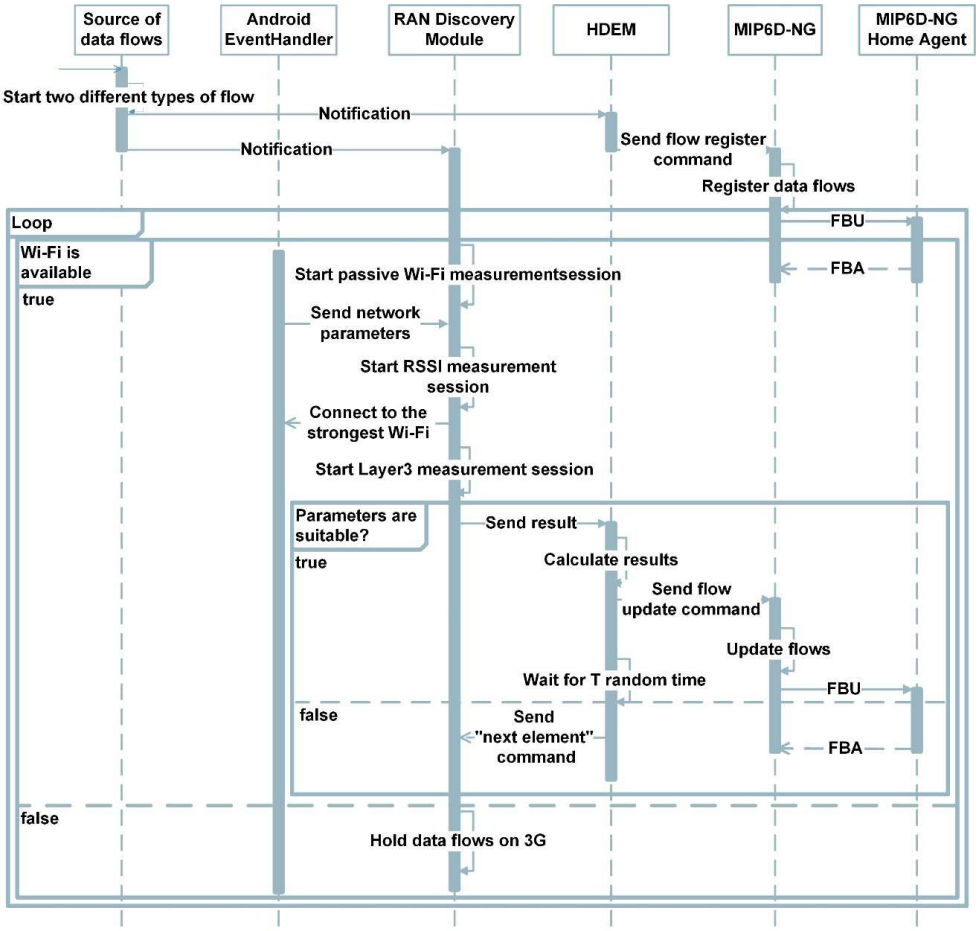

Figure 22: Cross-layer optimized flow mobility management scheme

3 Proof-of-concept demonstration scenarios

3.1 Emergency demonstration scenario

3.1.1 Use cases included in the scenario

Requirements derived from two use cases are mapped in this demonstration scenario, namely the “ambulance and emergency area” and the “emergency area with multiple casualties” use cases.

This demonstration scenario considers the transmission of image and video data from an emergency area in which an ambulance is deployed to a remote hospital using a 4G commercial network. Different types of image/video data have to be transmitted from the ambulance to the hospital in order to allow an accurate diagnosis to help the personnel on the field and to speed up the acceptance process once back at the hospital. Hence, multiple data flows need to be transmitted from the ambulance to the hospital; this includes both environment videos (from cameras deployed inside and outside the ambulance) and medical videos (i.e., ultrasound videos).

More details about the two use cases mapped in this scenario can be found in Deliverable D2.1.

3.1.2 Demonstration storyline

The storyline of the demo in this scenario is illustrated in Figure 23.

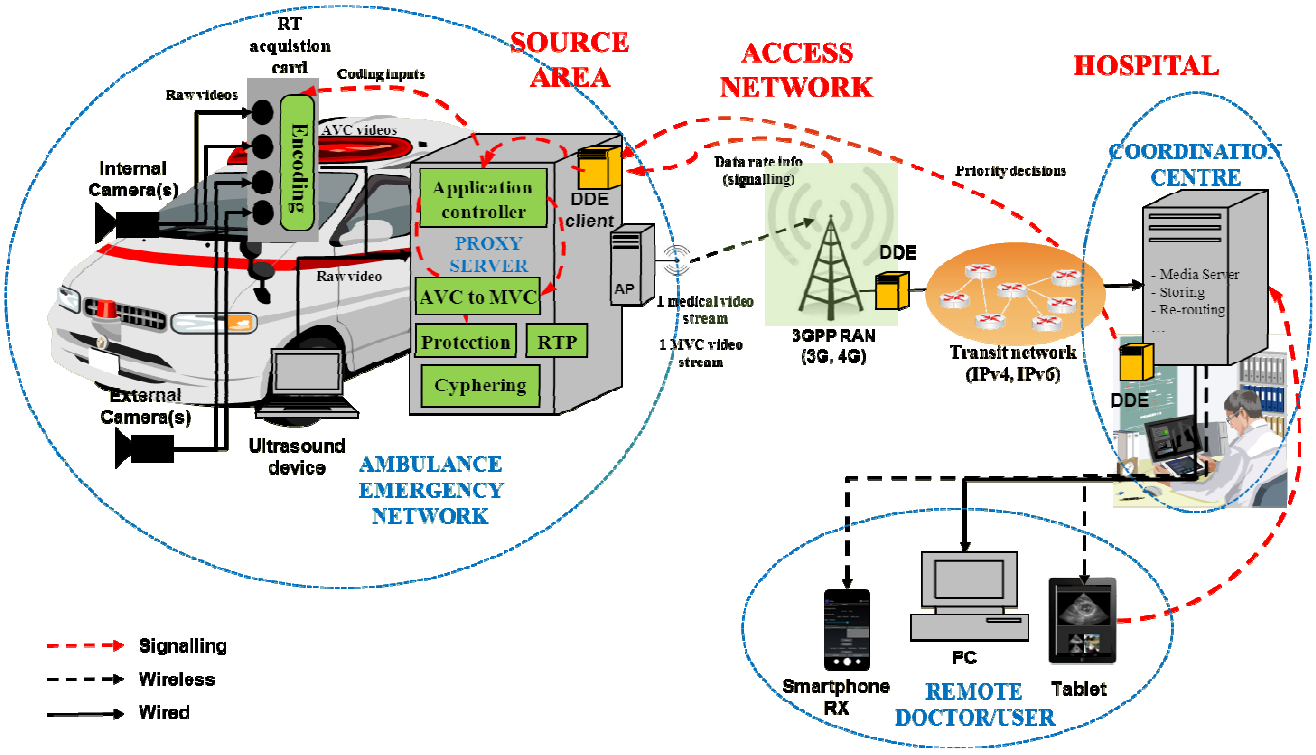

Figure 23: Demo setup of emergency scenario

An ambulance is deployed in an emergency area were an incident occurred. The ambulance is equipped with cameras and a portable ultrasound device to acquire data in the emergency area and send them to the hospital where the intervention is monitored. In case of need, the video flow received from the emergency area at the Coordination Centre can be sent in real time on the mobile device of a specialist that is walking inside the hospital.

In the emergency area, ambient video feeds are recorded and encoded by an H.264/AVC hardware IMP box. The videos are acquired by cameras both inside and outside the ambulance. From the DDE information provided by the network, the Proxy Server will control the IMP box to adapt the compression rate of the different encoded streams with the total

available bandwidth on the network. Moreover, the Proxy Server will transcode the videos in a single MVC flow adapting the quality according to feedbacks received by the hospital through DDE.

Similarly, ultrasound image data acquired through the portable ultrasound machine is processed. Due to the important needs of quality of medical videos, the Ultrasound images are compressed at a fix bitrate. The Proxy Server adds for both ambient and medical videos Forward Error Correction (FEC) codes and transmits the flows over an LTE network.

On the hospital side (i.e., in the coordination centre), all video streams are received, each at a dedicated RTSP/RTP interface, which serves as a relay. On the one hand, the received streams are mixed in order to create a single video stream containing all information. This stream is viewed on a dedicated client by a medical doctor. A single stream of the compound view can be interactively selected. The selected view is streamed from the RTSP/RTP relay in the coordination centre and displayed in full resolution in addition to the stitched stream.

In addition to the live viewing, all streams are recorded in the MP4/SAF format. Again, all streams are received from the RTSP/RTP relay. The recorded streams can be replayed on a dedicated RTSP/RTP player or as an MPEG-DASH stream.

3.1.3 Validation methodology

To validate the CONCERTO solution in this demonstration scenario and to show its benefits, we will run a first session of transmission with disabling all CONCERTO optimisation and adaptation, to establish a reference that permits us to estimate the gain obtained. Then we run the platform activating CONCERTO optimization techniques. The statements on the performance are based on the evaluation of the quality of received video streams in the two cases. The evaluation is done in two different ways: firstly, the quality of received videos is evaluated using objective metrics like PSNR, MSE and VQM in order to measure the gains introduced by CONCERTO solution. Secondly, a subjective evaluation is done by doctors on the quality of the received medical videos following the methodology described in Deliverable D3.3. Doctor's feedback is the most significant validation, since it allows to state if the quality is acceptable to perform a diagnosis and hence if the CONCERTO system can be used in a real-life emergency scenario. All the videos streams at the emergency area (emission side) and at the hospital side (receiver side) are recorded to perform later the computation of objective metrics, and validate the results of our solutions. This also allows to re-run the platform using pre-recorded videos for transmission in order to compare the results on the same sequences with and without CONCERTO optimization and adaptation.
Moreover, knowing the fact that the 4G link cannot be equivalent through the different sessions, we also estimate the instantaneous delay, jitter and Packet-Loss Rate (PLR) for each session. This allows on one side to compare fairly the results with and without CONCERTO solution activated and, on the other side, to validate the possibility to use the CONCERTO system in real time through a commercial network.

3.1.4 Field-trials realized and planned

We have successfully demonstrated our solution for real-time acquisition of ambient video and ultra sound with validation from medical doctors in September 2014, at the Faculty of Medicine of University of Perugia, Italy. The cypher solution is implemented on Android devices (Samsung Note) was demonstrated in Bologna at the EuCNC 2014 conference (June 2014, Bologna, Italy) as a part of the full CONCERTO demonstration. A preliminary version of the demonstration setup was tested and shown already in March 2014 at the Future Internet Assembly (FIA 2014) in Athens, Greece.

A complete demonstration of the emergency demonstration scenario, including a 4G transmission of real-time medical and ambient image/video content from an ambulance to the hospital, is planned for February 2015, again in the Hospital of Perugia, i.e., the Faculty of Medicine of University of Perugia, Italy.

3.2 Ubiquitous tele-consultation demonstration scenario

Recently, multi-access mobile devices interconnected with different wearable vital sensors via Body Area Network (BAN, also referred to as a Body Sensor Network or BSN) played an increasingly important role in healthcare and provide significant solutions for home healthcare, remote patient monitoring and real-time tele-consultation [8]. The spreading of heterogeneous and overlapping wireless access technologies make possible for mobile healthcare applications to use the available network resources efficiently and call into being the ubiquitous Internet connection for these applications. These facts motivated us to design and implement a context-aware IPv6 flow mobility management scheme for multi-sensor-based mobile patient monitoring and tele-consultation. We designed and implemented a cross-layer optimization platform and a flow-aware, client-based mobility management mechanism for Android-based systems interconnected with different medical body sensors and Smartphone CCD video cameras. This demonstration scenario is focused on context-aware flow mobility mechanisms for multi-sensor mHealth services and highlights the effectiveness of CONCERTO solution in the context of ubiquitous tele-consultation applications with real-time patient monitoring support.

3.2.1 Use cases included in the scenario

The main use case is No. 4 (Ubiquitous tele-consultation) where preliminary consultation with nursing-home staff, patients and other doctors may be carried out using hand-held devices, which provide access to multiple heterogeneous wireless networks, regardless whether he/she is in an ambulance, office, airport, traffic jam, or other place. However, optimization solutions applied in this demonstration are also applicable in other CONCERTO use cases like 1, 2, and 7.

3.2.2 Demonstration story-line

The demonstration story-line of this scenario focuses on a ubiquitous and real-time patient monitoring situation where two different data sources are used for continuous transmission of patient's vital information from the Body Area Network to the Coordination Centre located in the hospital. The two sources are heart rate information from Zehpyr HxM, and HD live stream from the Smartphone's camera:

1. Zephyr HxM uses Bluetooth to transmit heart rate, speed, distance and RR interval (R wave to R wave interval, which is the inverse parameter of heart rate) towards the Smartphone device. In the demonstration this device is applied as a continuous heart rate monitor, and the system forwards the measured heart rate values to the hospital's Coordination Centre. When using this sensor we have to pair the device with the host by relying on the Android SDK's BluetoothDevice class. Also Zephyr provides an SDK, which allows us to easily collect and process body vital data on the host Smartphone. The SDK provides the ZephyrProtocol and HRSpeedDistPacketInfo classes; using these we are able to communicate and receive the necessary information (heart rate data and timestamp) with the Zephyr HxM sensor node.

2. To provide high-quality video streams to the hospital we apply the internal CCD camera of the Smartphone. The Libstreaming external android library is used to stream the camera and microphone data of the Android device with RTP protocol over UDP. The library supports H.264, H.263, AAC and AMR encoders (we have chosen H.264). On the client side (i.e., in the hospital) a Wowza Media Server plays the streamed camera video originated from the Smartphone. Also an HTTP server (Apache 2) was deployed at the hospital side to manage the playout of the stream. The library allows the phone to play the role of an RTSP server. In that case the smartphone is waiting for a RTSP client to request a stream. It is possible to use Libstreaming without RTSP, thus the session using only SDP over arbitrary transport protocol. Libstreaming requires Android 4.1 or above.

The demonstration starts with a state where both of these vital data sources are transmitted using a wide coverage 3G/4G access network. Then – due to the movement of the patient – alternative access networks appear and according to the varying wireless environment, intervention will be needed for the optimal usage of the available wireless resources on the application flow level.
Figure 24 shows the cross-layer information exchange and user data path of the two flows before and after a certain mobility event in the demonstrator. According to the initial case, the mobile device uses the commercial 3G/4G cellular network. Due to mobility-related or other reasons, e.g. too expensive usage cost and/or insufficient QoS, the mobile device queries for alternative connection points from NIS. NIS resides in the current cellular network. Based on the location of the mobile device, NIS determines that there is a WLAN network in range and access points AP1 and AP2 cover mobile device's current location. Because NIS knows this network, it can be operated by the mobile network operator of the current cellular network or the operator has made roaming agreement with the WLAN network. The mobile device connects to the AP1 and moves the bandwidth-sensitive HD video stream to the Wi-Fi interface while

leaving the heart rate sensor information on the 3G/4G network. The WLAN network provides also a service for monitoring the quality of the access points through DDE. Thus, the mobile device subscribes to parameters report event by sending a DDE message to the DDE EventCache in the WLAN network. After that, the mobile device receives parameters events as presented in Section 2.3.3. Based on the events, the mobile device notices that the AP1 is getting congested and the mobile device switches to the AP2, handles the mobility of the video flow still bound to the Wi-Fi access, and leaves the heart rate data transmission on the 3G/4G network.
Flow-aware multi-access communication helps to continuously monitor the patient's vital data while also considering application / user profiles and the continuously changing wireless networking parameters, and performing a highly adaptive mobile transmission for better QoS/QoE.

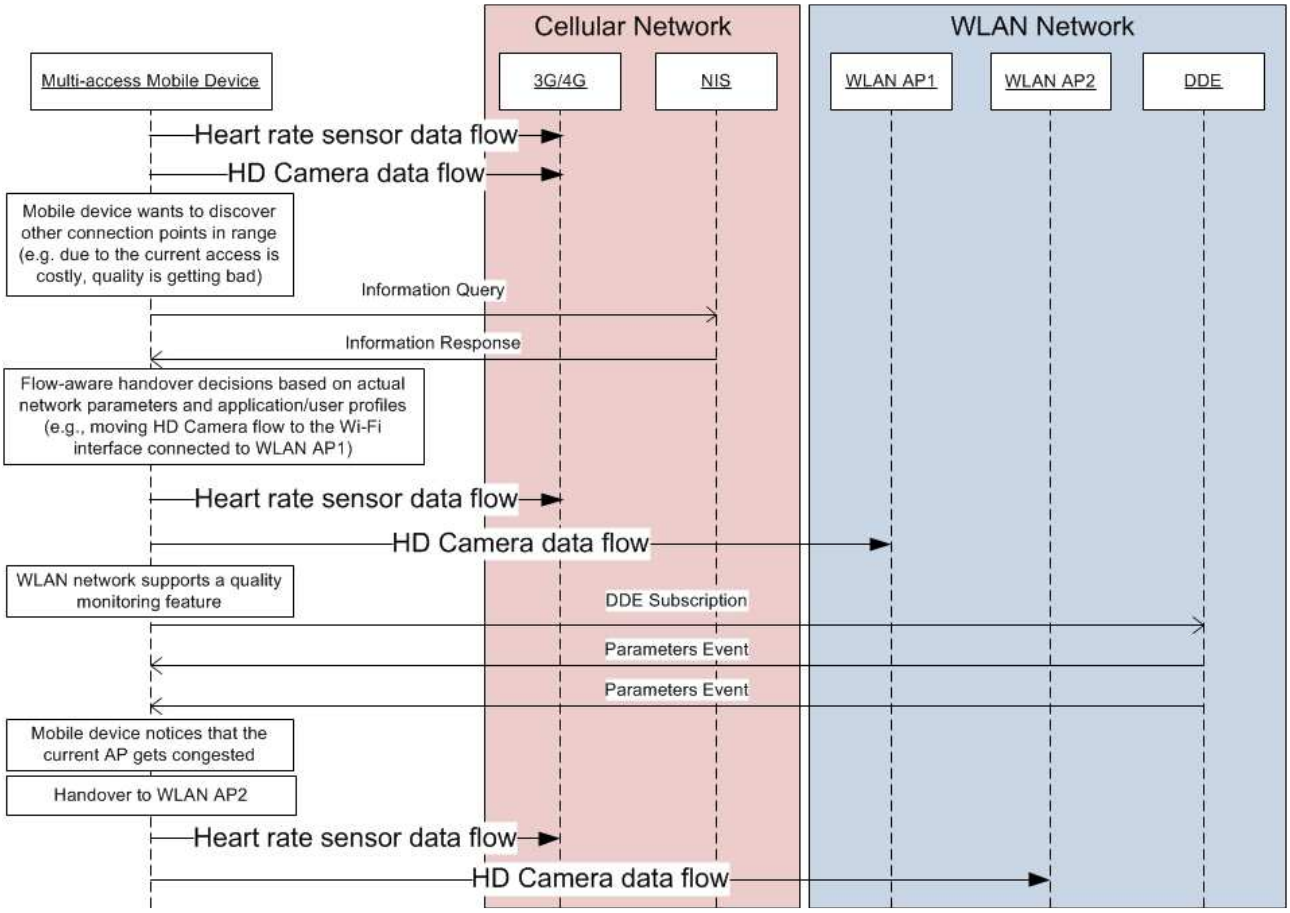

Figure 24: Cross-layer information services for intelligent flow-aware mobility management

3.2.3 Validation methodology

In order to present the efficiency of the CONCERTO mHealth framework in this use case and to evaluate the proposed underlying context-aware, flow mobility architecture, we designed and implemented different measurement scenarios. We have put our focus on the performance characteristics of the flow vertical handover process of the proposed scheme. In the first scenario we measure the quality of the transmitted video from the mobile node towards the correspondent node using MSU Video Quality Measurement Tool. We send a YUV video file with 4:2:0 sampling rate and with QCIF (176x144) resolution. For the measurements both of the sender and receiver side were developed in Java, using Datagram Socket implementation. We add a unique sequence number to each transmitted package to guarantee the in-order delivery of video frames on the receiver side. For the demonstration a live camera stream is originated from the smartphone, and visual distortions of the video material will be obvious when wireless networking parameters will be degrading.

The received and transmitted videos are then compared based on three different video metrics: APSNR, VQM and MSE. PSNR metric is used often in practice, called "peak to peak signal to noise ratio" To take simple average of all the per frame PSNR values, we can use APSNR.

VQM measures the perceptual effects of video impairment including blurring, global noise, block and colour distortion, and combines them into a single metric. MSE is the mean squared error in relation to the maximum possible value of the luminance [21]. If the frame is flawless the value of APSNR is 100.0, the VQM and the MSE are 0.0.

In the second measurement scenario we define two different flows: a UDP flow for video transmission and another UDP flow for heart rate sending. The heart rate (in beats per minute (BPM)) and the APSNR metric of the video are then measured in two different cases. In the first case, the flows of the video stream and of the heart rate are assigned to the 3G interface, because 3G is the only available network. In the second case the two flows are assigned to different interfaces (3G and Wi-Fi.
The results obtained in the described measurement scenarios will be reported in deliverable D6.5.

3.2.4 Field-trials realized and planned

We have successfully demonstrated our solution for remote patient monitoring and ubiquitous real-time tele-consultation on 11/09/2014, at the Faculty of Medicine of University of Perugia, Italy. The infrastructure basics of our Android-based IPv6 flow mobility architecture for multi-sensor based mHealth services were demonstrated in Bologna at the EuCNC 2014 conference (23-26/06/2014, Bologna, Italy) and during the FIA 2014 exhibition (Future Internet Assembly, Athens, Greece, 18-20/03/2014), as a part of the full CONCERTO demonstration.

3.3 3D medical data storage and encoding

3.3.1 Use cases included in the scenario

The effective medical data encoding system could be applied in all scenarios, where large data medical sets are stored or transmitted. (Use cases: 1, 2, 3, 4, 6, 7). In Use case 5 - Surgical assistance – near-real-time communication is essential, so solutions based on high-complexity algorithms cannot be applied.

3.3.2 Demonstration story-line

The 3D medical data sets are stored in the database of our test server. One 3D data set consists of slices through an object, where each 2D slice is normally presented as a grey scale image. We can view the 2D images one-by-one, or get a montage of the 2D slices (as a preview), in 9, 16, or 64 pictures in array (see Figure 25).

Figure 25: Preview from 16 and 9 slices

In order to view the original 3D data, we apply a texture-based volume rendering technique, where we perform the sampling and compositing steps by rendering a set of slices inside the volume (see Figure 26).

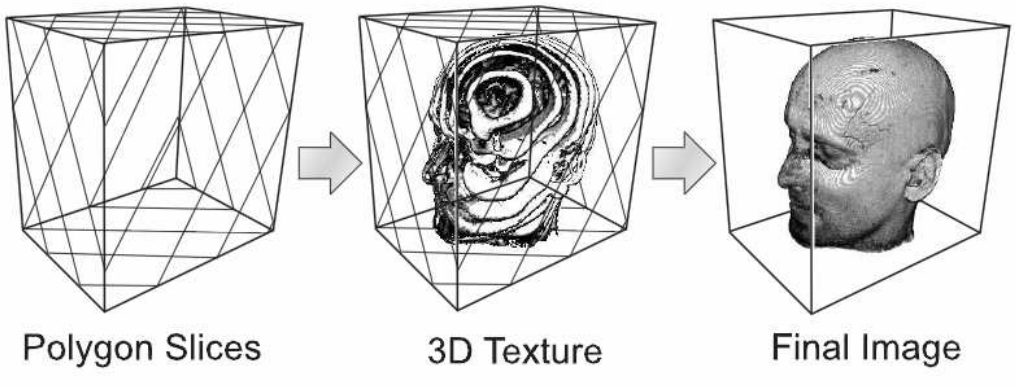

Figure 26: Texture-based volume rendering

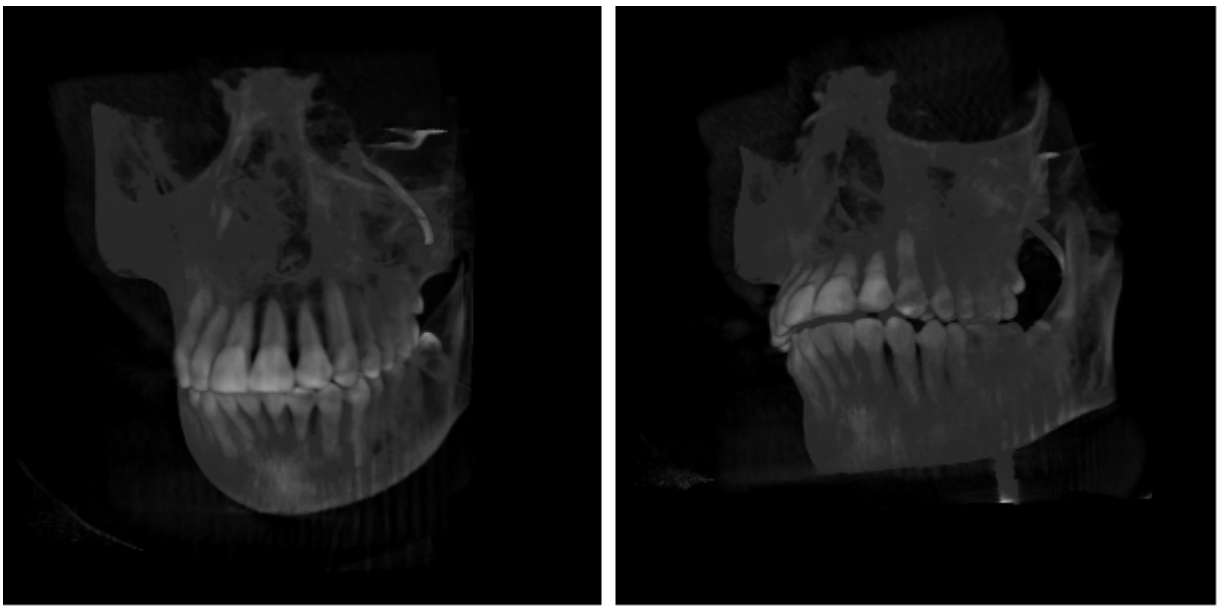

Figure 27: Rendered 3D images from different point-of-view.

Using our volume rendering engine, we can render a free point of view 3D image from the original data set. We use an *optical model* to map data values to *optical properties* (like color and opacity). Optical parameters are calculated by applying *transfer functions* to the data. For example, with changing the transfer function, we can render the tooth with more intensity than the surrounding bones (Figure 28).

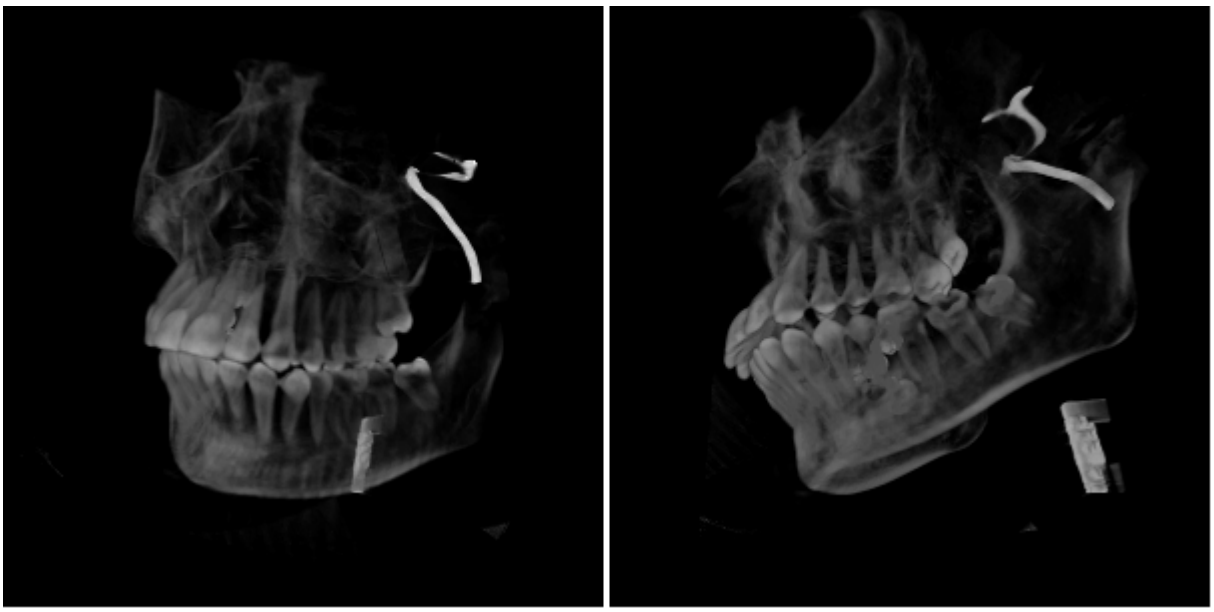

Figure 28: Rendering tooth with higher intensity

The work flow is summarized in Figure 30: At first, the user selects the original 3D data to be processed. An optional montage of 2D slices is available, where the user sets the number of slices to be viewed. A free point-of-view volume rendered 3D image is also applicable, with some optional settings for the rendering (like parameters of the alpha blending). After choosing the applied directory-based methods (lzma, gz2 etc), and setting the parameters for JPEG 2000 and H.264/AVC-based encoding, the medical data file is encoded in several different ways, and statistics are generated about the different compression methods.

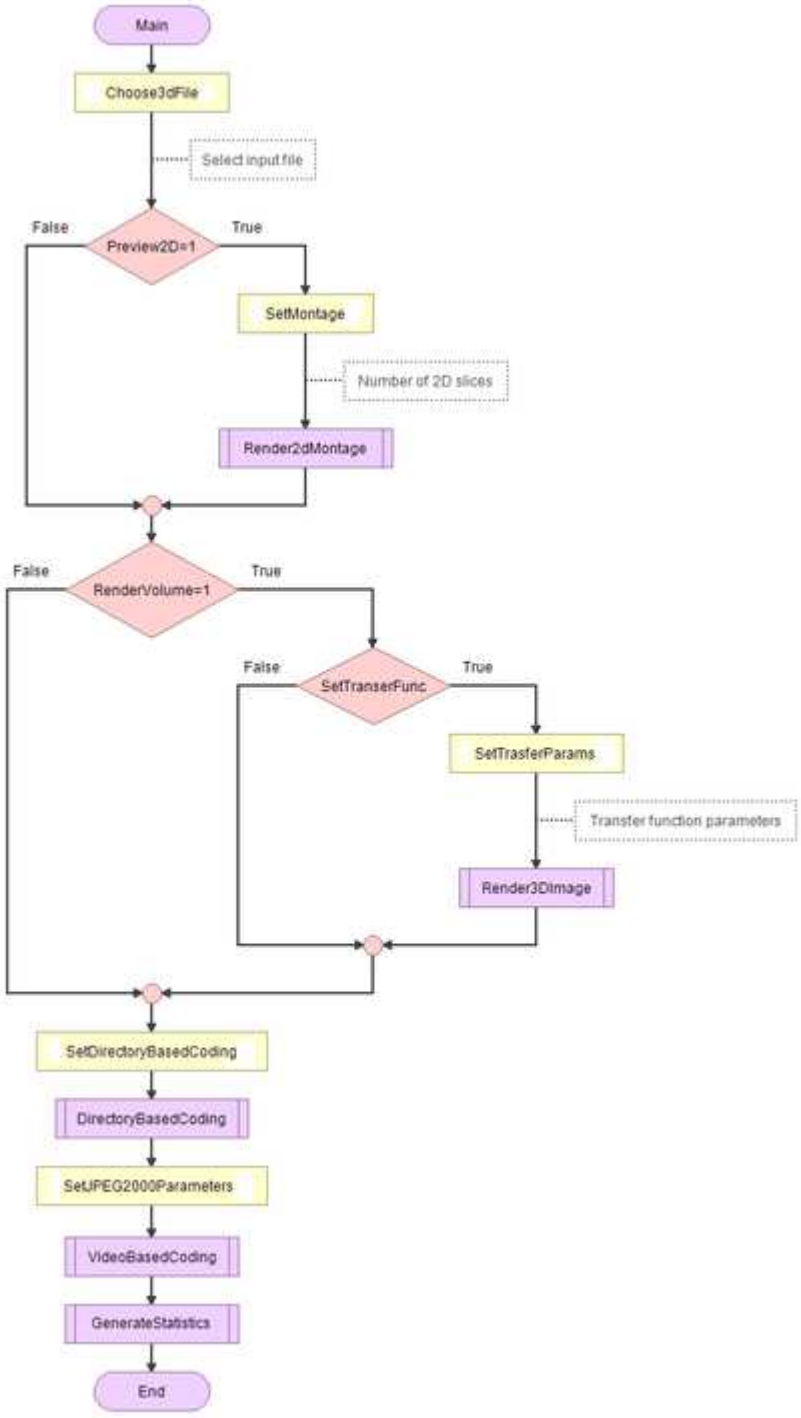

Figure 29: Flowchart of the 3D medical data storage and encoding demo

3.3.3 Validation methodology

For the efficient storage of 3D medical data, we run the demonstration scenario appling and comparing several compress methods. Then, our test system generates statistics from the efficiency of compression. The used compression methods are: lzma, gzip, bz2, lzma with –e parameter, and lossless operation of JPEG 2000.
The elaborated statistics allow comparing the different compressing methods and evaluating their performance.
Discussions on the obtained results will be presented in deliverable D6.5.

4 Conclusions

The objective of this deliverable was to present the final implementation of the CONCERTO multimedia platform that has been developed in the scope of the project in order to demonstrate on a real-time implementation the CONCERTO solution for different scenarios. The components and the functionalities integrated in the demonstrator are issued of the technical work carried on in the other work packages of the project; their full description is available in the correspondent technical deliverables.

In addition to the overall platform, the three demonstration scenarios selected to validate the CONCERTO solutions are equally presented along with their validation strategy.

This deliverable also provides a description of the validation methodology and of the realized and planned field trials, but it does not include the related results that will be presented in the next and last deliverable of the work package: D6.5 “Report on final validation”, expected in February 2015. In Deliverable D6.5, the results of the different field trials performed with the CONCERTO multimedia platform (described in this deliverable) will be reported as well as medical doctors’ evaluations and conclusions on all the actions engaged to validate the CONCERTO solution in the selected demonstration scenarios.

5 Annex

5.1 Annex 1: Low-complexity encoding of HEVC video streams

Introduction

In previous studies concerning fast coding mode decisions for encoding HEVC bitstreams in the scope of this work (i.e., Deliverable D3.3), the size of the individual Coding Units (CUs) was assumed to be fixed. For example, the picture to encode was decomposed into Coding Tree Units (CTUs) of size 32x32 or 16x16, and no further splitting of the CTUs into smaller CUs was allowed. The process of mode decision was limited to the selection of an efficient intra or inter prediction mode. As a part of this, also the partitioning of a CU into multiple Prediction Units (PUs) was conducted for some modes, i.e., splitting into 2 PUs for inter picture prediction (INTER 2NxN and INTER Nx2N partitionings), as well as splitting into 4 PUs for intra frame prediction (INTRA NxN partitioning). An overview of the mode decision scheme evaluated so far is depicted in Figure 30.

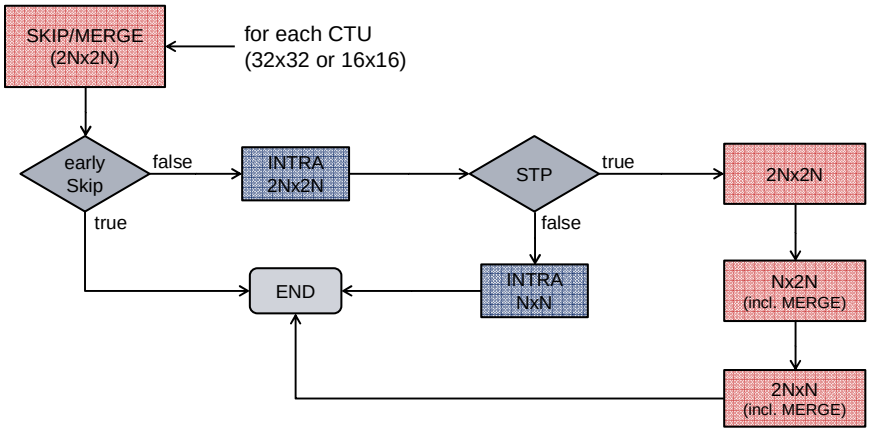

Figure 30: Former mode decision flow chart for a predefined CU size

Besides the overall structure of the mode decision process, Figure 30 also reveals two conditions within the decision flow, which are incorporated to speed up the process. First, there is the “early skip” (ES) condition, where it is checked whether the rate-distortion (RD) costs using the skip mode are smaller than those using the merge mode for coding the current CU. If this is the case, no other mode tests are performed for this CU and skip is immediately selected as the best mode. Second, there is the “spatial or temporal prediction” (STP) condition, formerly also denoted as “merge” condition. This condition evaluates if the merge mode is more efficient than the INTRA 2Nx2N mode for the current CU, i.e., if the RD costs for merge are smaller. If the condition is true, the three inter modes shown on the right-hand side of Figure 30 are tested (INTER 2Nx2N, Nx2N, and 2NxN). Otherwise, if INTRA 2Nx2N is more efficient, the INTRA NxN mode is tested additionally.

In this deliverable, an extension of the just described mode decision flow is presented where the limitation of the CU size is loosened to a certain extend. In general, HEVC allows CU sizes from eight by eight to 64 by 64 pixels. However, depending on the video content to encode, it may not be necessary to place the root of the CU quadtree at the maximum CU size and to evaluate quadtree partitionings down to the smallest size. If all possible sizes are considered, a lot of computing time is invested to find the best mode, although one of the intermediate CU sizes may deliver a reasonably well performance in the majority of cases. Consequently, a trade-off between the investigated range of CU sizes, i.e., the invested encoding time, and the resulting coding efficiency has to be found. Furthermore, the question should be addressed, whether it is beneficial to enable splitting CUs into multiple PUs while limiting the allowed depth of the CU quadtree, or to do it the other way around, i.e., to disable PU splitting while testing smaller CUs.

The next section describes the extended mode decision scheme, which begins with the fast determination of the CU quadtree. Afterwards, experimental results are presented using this extended scheme, before a conclusion is given.

Fast Mode Decision Including Quadtree Partitioning

In order to quickly determine an efficient quadtree decomposition of a CTU, the mode decision process is subdivided into two parts. In the first stage the CU quadtree is created, where only the RD costs of the skip or merge mode (2Nx2N PU partitioning) are used as criterion to decide for further splitting or no further splitting. In a second step, all CUs of

this previously determined quadtree are revisited and the prediction mode of each CU is reassessed by testing other modes, too. In the following, the merge and skip mode of HEVC will be explained based on [2], as it is used as the key for the creation of the CU quadtree. Afterwards, the investigated mode decision flow will be presented and some characteristics of this scheme will be discussed.

Due to fact that a picture is divided into a fixed CTU raster and the concept of decomposing CTUs into smaller CUs using a quadtree structure in HEVC, the problem arises that picture areas may be segmented into different CUs although the motion of the complete area is homogeneous. In order to avoid coding redundant motion information in different PUs, the so-called merge mode has been introduced into HEVC. The merge mode allows inheriting motion information for the current PU from a set of spatially or temporally causal PUs already coded. As a first step, the motion information from these neighbouring CUs is collected and a list of merge candidates is created. The collection of possible merge candidates continues until the length of the list reaches a certain number *NumMergeCands*, which is signalled in the slice header and is set to five in the HEVC reference software HM-14.0 [3] by default, for example. When creating the list, first five spatial candidates are considered to be added to the list. The positions of the spatial candidates can be seen in Figure 31(a). If available, up to four of the five candidates are added to the list in the following order: A_1, B_1, B_0, A_0, and B_2. Furthermore, a redundancy check is performed in order to avoid adding redundant motion information to the list.

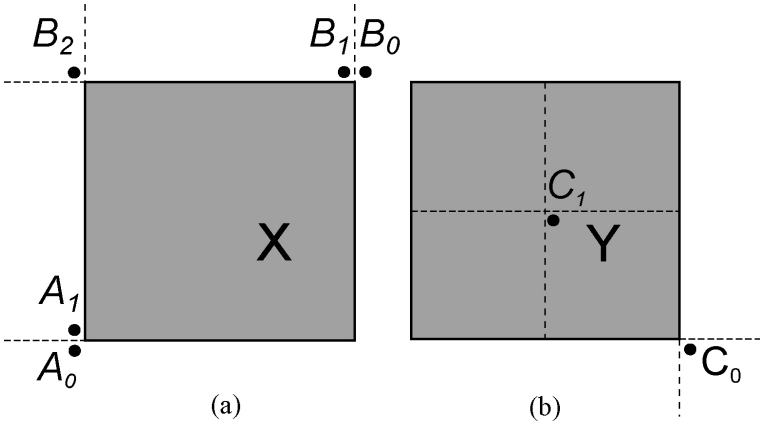

Figure 31: (a) Positions of the five spatial merge candidates of the current block X. (b) Positions of the temporal merge candidates relative to the collocated block Y of X [1].

Afterwards, one of the two temporal candidates C_0 and C_1 as shown in Figure 31(b) is added to the list, where C_0 is used if it is available and inside the current CTU row, and C_1 is used otherwise [1]. If the total number of spatial and temporal merge candidates in the list is less than *NumMergeCands*, additional "virtual" candidates are created by combining motion data already in the list and finally by using zero vectors with different reference picture indices [1]. Finally, only the index of the respective list entry is encoded to signal the best merge candidate. In addition, when considering RD costs, it may be advantageous to use the motion compensated prediction obtained by employing the merged motion data directly as reconstructed picture and not to transmit additional residual data. If this is the case, the so-called skip mode is activated by setting the skip flag and coding of the residual data is skipped. More detailed information on the merge mode of HEVC can be found in [1].

As already mentioned, the RD costs of the skip/merge mode are used to create the CU quadtree. To be more precise, the minimal skip/merge costs for coding a CU in a certain level of the tree are recursively compared to the minimal skip/merge costs of the four sub-CUs, using 2Nx2N PU partitioning in all cases. The CU of the current quadtree depth is split into four sub-CUs if the total minimal skip/merge RD costs of the sub-CUs are smaller than the minimal skip/merge RD costs of the current CU. In order to compute the minimal skip/merge RD costs of a PU, the motion information of all merge candidates within the created list is tested in order to predict the current PU and the respective RD costs are computed. Additionally, the RD cost evaluation is done including the coding of residual data as well as without coding of residual data (skip mode). The mode belonging to the minimum of all these RD costs is considered as the most efficient one.

After the quadtree decomposition of a CTU based on minimal skip/merge costs is found, each individual CU of the quadtree is revisited in decoding order, i.e., in depth-first order using a z-scan. A mode refinement process is executed for each CU in order to find the optimal coding mode considering a larger set of possible modes, not only skip/merge. It should be emphasized that the skip/merge candidate selection and the cost computation for a certain 2Nx2N PU in general have to be recomputed in the refinement step, although both steps have already been performed in the first stage

when creating the quadtree. The reason for this is that the initially selected merge candidate may have become not optimal any more or even invalid due to a change of the selected mode for this referenced PU in the refinement step. For example another merge candidate may now deliver a better prediction for the current PU due to modified motion information of this candidate. Moreover, a candidate may have become invalid if intra picture prediction has been selected for this referenced PU in the refinement step. Thus, there is no motion information to inherit any more. The described behaviour applies for the spatial merge candidates of a PU. Although the motion data of the temporal candidates is constant, it is very unlikely that the motion information of all spatial candidates has not been modified during the mode refinement process so far. There is only one exception, namely the PU in the top left corner of the CTU. For this PU the mode of the spatial merge candidates cannot have been changed between the creation of the quadtree and the mode refinement of this particular CU. However, in order to keep the description simple and the implementation consistent with the description, the skip/merge costs are always computed again at the beginning of the mode refinement stage. An exception may be introduced for this case when optimizing an implementation of this mode decision scheme.

In principle the mode refinement stage consists of traversing the mode tests as depicted in Figure 30 for each CU of the determined quadtree. However, due to the incorporation of different CTU sizes and depths of the coding tree, the question arises whether it is beneficial to allow splitting into smaller PUs (e.g., INTRA NxN, INTER NxN, or INTER Nx2N) while restricting the maximal quadtree depth, or to allow one additional depth level while disabling PU splitting. Figure 4 depicts the flow of mode tests for each CU in the mode refinement stage, where those mode tests where a splitting into smaller PUs is applied are drawn with dashed boxes, illustrating that they may be switched on or off. The best trade-off has to be found by experimental evaluations and will be addressed in the next section.

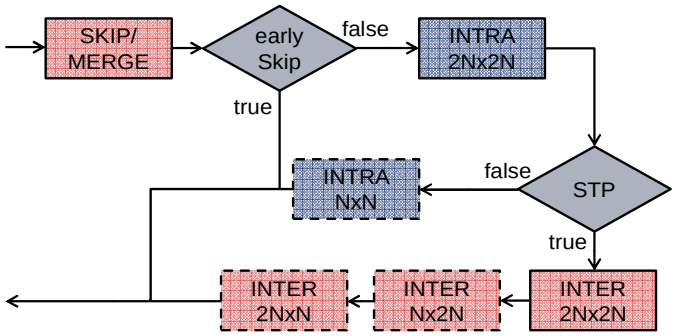

Figure 32: Decision flow for each CU in mode refinement stage.

In order to speed up the mode decision process, several fast methods are employed. As it is shown in Figure 32, the early skip as well as the STP condition are used to save processing time. These conditions have already been incorporated in previous studies as shown in Figure 30. Besides these two conditions to speed up the mode decision in the refinement step, another condition called “early CU” can be applied when creating the quadtree in the first stage. The early CU condition works in a similar manner as the early skip check. The early CU condition is evaluated on each depth level of the emerging quadtree after the skip/merge test has been performed. If the RD costs using the skip mode are smaller than those using the merge mode, no further splitting of the respective CU is considered. Otherwise, further splitting is allowed. The effects of these three conditions in terms of saved processing time compared to the potentially decreased quality and increased bitrate will be evaluated in the following section.

Experimental Results

The presented two-stage decision flow scheme has been implemented into the HEVC reference software HM-14.0 [3]. For the presentation of the experimental evaluation this implementation is denoted as “FastQT” hereinafter. As already mentioned, those mode tests leading to a splitting of the CU into several PUs (drawn with dashed boxes in Figure 32) may be switched on or off within the software. Furthermore, also the three conditions for speeding-up the mode decision process may be enabled and disabled for testing purposes. Besides this modified software, also simulations using the original HEVC test model HM-14.0 are run as an anchor. In general, different CTU sizes and depths of the coding tree are considered. Abbreviations are used for a compact representation of these different configurations and software implementations. These abbreviations are summarized in Table 6. The digit “1” is appended to an abbreviation if the respective feature is activated for a simulation. Otherwise the suffix “0” is added.

Two sequence classes of the HEVC Common Test Conditions [4] are used for the simulations, namely *ClassC* consisting of four sequences with resolutions of 832x480 pixels and *ClassE* consisting of three sequences with resolutions of 1280x720 pixels.

Abbreviation	Description
HM	Original HM-14.0 software
FastQT	Modified HM-14.0 software using the introduced two-stage mode decision scheme
64, 32, 16, 8	Considered CU sizes; highest number gives CTU size (e.g. 64x64), smallest number gives smallest CU size (e.g. 8x8)
PUS	PU splitting: INTRA NxN, INTER 2NxN, and INTER Nx2N modes are tested if enabled, otherwise not.
ECU	Early CU condition
ES	Early skip condition
STP	Spatial or temporal prediction condition
fast	All fast methods enabled, i.e., ECU, ES, and MC enabled (unless otherwise specified)

Table 6: Summary of the abbreviations used in this section.

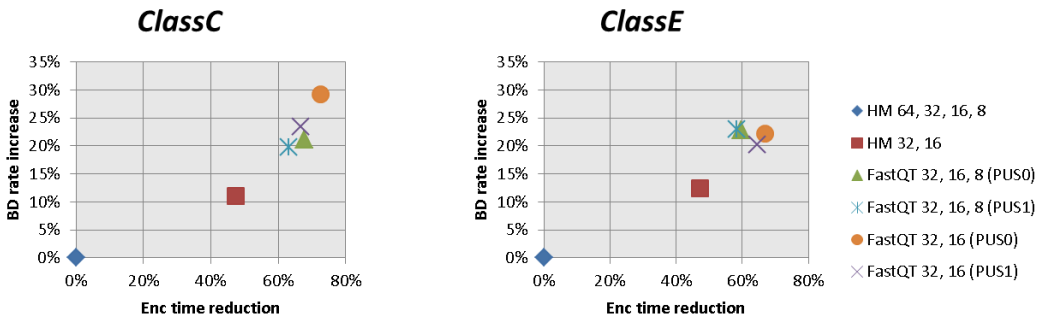

Figure 33: Evaluation of coding performance: CU quadtree depth versus PU splitting (PUS) without employing any fast mode decision conditions.

As a first step the question should be addressed, whether it is advantageous to use very small CUs while disabling PU splitting (PUS0), or better prefer disabling the lowest CU depth but enabling PU splitting (PUS1). For the discussion of this question let us consider coding results without any fast methods activated. Intuitively, the use of 8x8 CUs is dependent on the video content to encode. If a picture contains more details, smaller blocks like 8x8 CUs are useful to achieve efficient compression. Naturally, the amount of details per area of a picture decreases when the resolution increases. This can be observed in Figure 33 when the results for disabled 8x8 CUs are compared to the results where 8x8 CUs are enabled. The figure shows averaged results of the encoding time reduction versus the increase of the BD rate. Thus, the more efficient (concerning time as well as BD rate) an encoder is, the further to the bottom right the respective result appears in the diagram. Anchor is the original HM-14.0 implementation using the *Low Delay P Main* configuration according to [4]. It can be clearly seen that the two FastQT tests without 8x8 CUs perform worse than the two other tests where 8x8 CUs are enabled for the *ClassC* video sequences (relatively low resolution). In contrast to that, the two FastQT configurations without 8x8 CUs perform best of all FastQT tests considering the rate as well as the encoding time for the *ClassE* sequences (higher resolution). Considering the differences between enabled and disabled PU splitting it may be concluded that the splitting into sub-PUs may be used to adjust the trade-off between encoding time and bitrate. However, the enabled PU splitting is not able to fully compensate for disabled 8x8 CUs (compare “FastQT 32, 16, 8 (PUS0)” to “FastQT 32, 16 (PUS1)”). When it comes to decide for a certain mode selection scheme, it may be wise to first decide for or against the use of 8x8 CUs depending on the target resolution of the video encoder. Afterwards, the use of sub-PUs may be adjusted depending on the desired trade-off between complexity and coding efficiency.

As can be seen in the left part of Figure 33, the processing blocks using no 8x8 CUs and no PU splitting are too large for the content of the *ClassC* sequences. The BD rate increases significantly. Due to this observation and the fact that

the influence of the enabled or disabled PU splitting should also be evaluated when fast mode decision methods are used, 8x8 CUs are enabled for the remaining analysis in this section of the deliverable.

In a second step the individual fast methods of the two-stage mode decision scheme are analysed in the following. Figure 34 presents results comparing mode decision enabling the early CU (ECU) condition and without this condition. The anchor is the same as in Figure 33. For the FastQT implementation, active as well as inactive PU splitting (PUS) is tested. Considering the HM anchor, basically no additional BD rate has to be accepted if ECU is enabled. The very high amount of encoding time reduction when using ECU with the HM software for the *ClassE* sequences may be explained by the content of these videos, which are teleconferencing scenarios containing a very large area of static background. Thus, using the FastQT implementation in this scenario is only advantageous concerning the encoding time if ECU is enabled. Nevertheless, activating ECU using FastQT achieves a considerable complexity reduction with only a rather small amount of additional rate compared to FastQT without ECU for the *ClassC* as well as the *ClassE* sequences. Besides that, the effect of enabled or disabled PUS is only marginal.

The second fast condition to analyse is the early skip (ES) condition. Respective simulation results are depicted in Figure 35. Compared to the ECU condition there is a clear difference observable. Although the activation of the ES condition leads to a speed-up of the coding process of about 7 to 9% for both sequence classes, this comes along with an increase of the BD rate of approximately 5%.

An even more considerable increase of the BD rate compared to the reduction of the encoding complexity can be observed for the evaluation of the STP condition in Figure 36. Consequently, the STP condition may not be recommended for the evaluated coding configurations. However, the STP condition (formerly called “merge condition”) seemed to have worked reasonably well according to previous investigations with a slightly modified mode decision scheme. The exact reason for the bad performance using the mode decision scheme presented here still has to be clarified in detail.

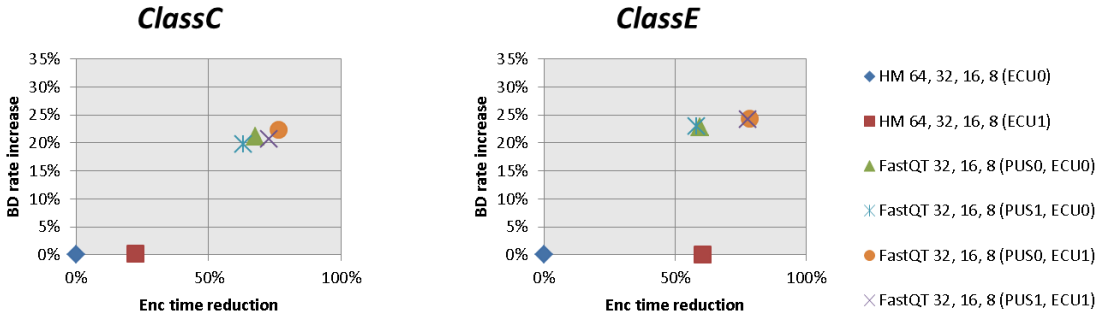

Figure 34: Evaluation of coding performance using the ECU condition (with and without PU splitting).

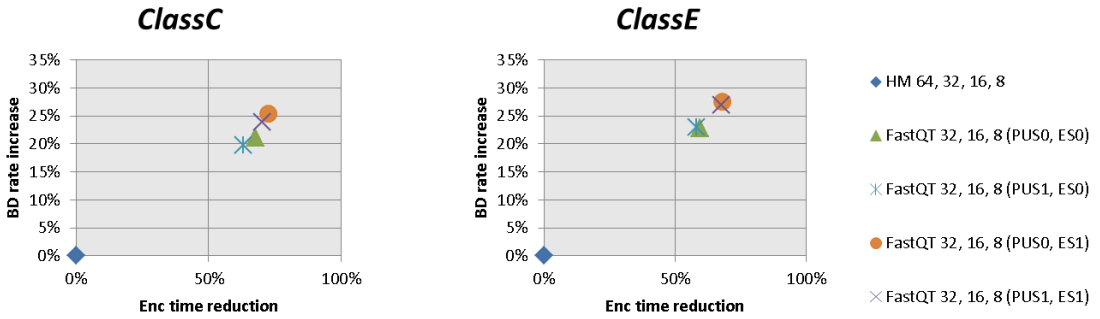

Figure 35: Evaluation of coding performance using the ES condition (with and without PU splitting).

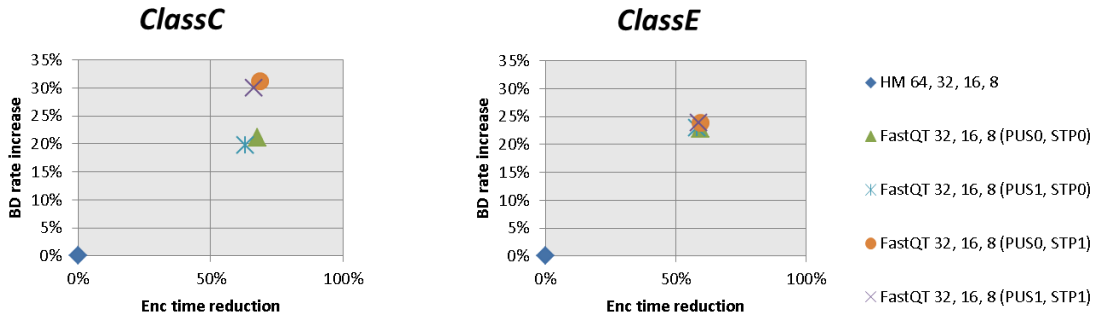

Figure 36: Evaluation of coding performance using the STP condition (with and without PU splitting).

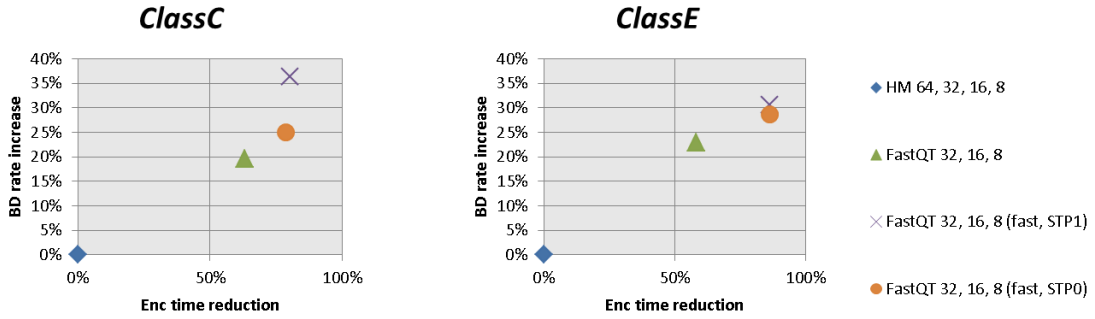

Figure 37: Evaluation of coding performance: all fast methods (ECU, ES, STP) versus all fast methods without STP versus no fast methods (PUS1 for all cases).

Finally, the combination of all three fast conditions remains to be evaluated. For the sake of completeness also the STP condition is considered here, although the results shown in Figure 36 already suggest disabling this feature. The respective coding results are depicted in Figure 37, where PUS is set to one for all simulations. It should be noted that the scaling of the vertical axis has been adapted to fit the extended range of the rate increase. Combining all fast conditions leads to a reduction of the encoding time of approximately 80% for *ClassC* (with about 36% BD rate increase) and approximately 86% for *ClassE* (with about 31% BD rate increase). Due to the fact that the STP condition did not perform very well when it is applied individually, as shown in Figure 36, additional results are shown where the fast methods except STP are employed, i.e., ECU and ES only. Disabling STP leads to a significant rate gain, especially for the *ClassC* sequences, analogously as it could be observed in Figure 36. As already explained, this behaviour is actually contradictive to previous results using a slightly modified mode decision scheme and the reason for the bad performance remains to be analysed.

Conclusion

A fast mode decision scheme for HEVC, denoted as FastQT, has been introduced and evaluated, where the coding tree based on the RD costs of the skip/merge mode is created in a first stage, followed by a mode refinement for each individual CU of the quadtree. Simulation results using this mode decision scheme show that incorporating those prediction modes which split a CU into multiple PUs (PUS) can be used to adjust the trade-off between complexity and coding efficiency of the encoder. However, the possibility to split into sub-PUs cannot compensate for disregarded small CU sizes. Thus, the size of the smallest CUs, i.e., the depth of the coding quadtree, should be adjusted according to the resolution of the video content to encode. Besides that, three conditions intended to speed up the encoding process have been evaluated, namely “early CU” (ECU), “early skip” (ES), and “spatial or temporal prediction” (STP). ECU as well as ES are methods adapted from the original HM software, whereas STP is a newly introduced method. ECU is able to achieve a significant encoding time reduction with a marginal increase of the BD rate. Also ES is able to speed-up the encoding process. However, here the amount of additional BD rate is not negligible. The STP is not able to confirm the promising results of previous investigations. This is probably caused by one of the modifications made to the mode decision scheme. Further analysis is necessary to reveal the exact reason for this behaviour.

5.2 Annex2: Hierarchical prediction mode for HEVC encoding

Introduction

Traditionally video frames are coded in "group of pictures" (GOP) mode. That means the first picture of a group is coded in INTRA (I) mode, having no dependencies on previously coded frames. Subsequent pictures in the group are dependent on the previously coded picture (INTER, P), sometimes on one or more other than the previously coded pictures from the same group. A video will be coded as IPPPP IPPPP ... sequence. This is intended to enable any decoder to start decoding the video stream at the start of a GOP rather than having to decode the full video starting at its first coded frame.

To facilitate this "random access" feature, typically a new GOP is started in the range of twice per second of coded video up to every few seconds. That translates into GOP lengths between ten and a few hundred frames, assuming the typical frame rate of twenty-four to thirty frames per second.

However, there are a few drawbacks. Decoding can only ever start at the first frame of a GOP, so in order to access the last frame of a GOP, all of the GOP has to be decoded. Additionally, a lost frame early in the group of pictures will render the rest of the group unable to decode correctly.

To improve upon this situation, hierarchical INTER coding was invented. A simple way to describe this feature would be to say the group of pictures has been subdivided into several sub-groups, called temporal layers. The basic layer (layer 0) corresponds to the INTRA frames used in traditional GOP coding. The next layer, layer 1, depends only on the previously coded layer 0 frame, while layer 2 frames depend on layer 0 and layer 1 frames. A video sequence coded with four layers would look as illustrated in Figure 38. The number of layers used is limited by the memory of the decoder, which has to match the number of reference frames required. In the example, the number of reference frames is equal to the number of layers minus one.

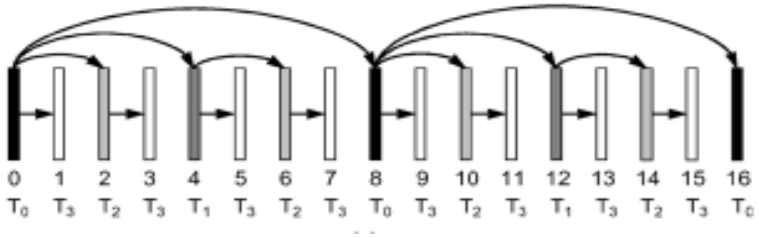

Figure 38: Hierarchical prediction structure for 4 layers.

The benefits of this coding mode are manifold, especially in an environment, where random access is required or where the video is transmitted over networks with packet loss, often resulting in frames that are not decode-able. If a frame loss should occur in one of the deeply nested layers (layer 3 in the above example), only little degradation of the video quality is experienced by the user. A frame loss in a higher layer will result in a more massive loss in video quality, as all images depending on the lost image are not decodeable anymore.

With traditional group of picture coding, different quantisation parameters (QP) are usually chosen for the INTRA and INTER images, to balance visual quality versus data rate. A typical setting would be to decrease the INTER quality by three steps respective to the INTRA quality.

The same principle also applies to the hierarchical prediction structures. Each layer is coded with a different QP, which is given as difference to a reference QP, usually the layer 0 QP. A proposal to calculating the QP differences is currently available in [5].

$$(1) \qquad QP(T>0) = QP(T0) + 3 + T$$

Please note, this proposal is meant for a slightly different use case, i.e., coding of B frames and not taking effects on the rate variance into account, but is used here as a starting point.

QP difference testing

Using four temporal layers as shown in Figure 38, layers 2 and 3 have two times and four times more frames compared to layers 0 and 1, however their quality should be lower because they are less important for motion prediction when

decoding the sequence. Layer 3 frames are not used at all for prediction, thus it is possible to reduce the rate for layer 3 without affecting the overall visual quality of the video proportionally. Of course, using abysmally low quality for layer 3 will be noticed by the viewing person and is not desirable.

As stated, the quality difference between the best frames in a sequence (e.g. layer 0) and the worst (e.g. layer 3) should not exceed a certain difference. However it is also desirable to have the rate distribution between layers and the frame size differences between successive frames well limited. Large variations in either affect the rate control performance adversely and will lead to issues when transmitting live video over the network. Typical network behaviour makes it more likely to have packet loss during bursts of data being transmitted and less likely during continuous transmission at a steady rate.

As a result, the QP differences between layers have to be carefully evaluated.

Initial statistical evaluation of data measured with the settings for quantisation parameter according to (1) showed a variation in frame size exceeding twentyfold on successive frames. This is deemed unacceptable for rate control application as this will lead to massive fluctuation between frames. Also, with the layer 0 frames being of excessive size they will be especially vulnerable to packet loss, rendering the full group of pictures unusable for decoding. A number of tests were run to try and find better settings, see Table 7 for the tested configurations. All configurations were tested with various initial QP settings and on multiple sequences with either two hundred or one thousand frames. Please note that a higher value for QP results in worse quality. The range of QP is from zero (best) to fifty-one (worst).

Mode	L0	L1	L2	L3
0000	0	0	0	0
1111	0	+1	+2	+3
2111	0	+2	+3	+4
3111	0	+3	+4	+5
4111	0	+4	+5	+6
2211	0	+2	+4	+5
M1M1M1M1	0	-1	-2	-3
M2M1M1M1	0	-2	-3	-4
M1011	0	-1	0	+1

Table 7: QP deltas for 4 layers as tested, with “+1” meaning “one step lower quality than reference QP”

The test results in Figure 39 and Table 8 show uniform behaviour between all sequences and all settings for initial QP settings. An initial test run with about thirty sequences over a variety of frame sizes was evaluated. It was found that the encoder behaves consistently over all sequences. For the purpose of this work, two sequences were selected, one recorded during a video conference and the other one being the BigBuckBunny sequence [6]. The latter sequence contains a wide variety of content which was found during the initial run to be the most demanding sequence for the encoder, especially with the rate-control development in mind. All following tests are run with the two sequences and four settings for the initial QP (22, 27, 32, 37). This test set was chosen to keep the effort of evaluating the results reasonable. Tests over more sequences are run infrequently to verify the consistency of the results.

Viewing the resulting sequences confirms the measurements. The evaluation will thus focus on specific samples selected from the test results. In order to prepare the video stream for lossy connections, several configurations with a reduced quality layer 0 are tested, although the overall rate-distortion performance is expected to be worse.

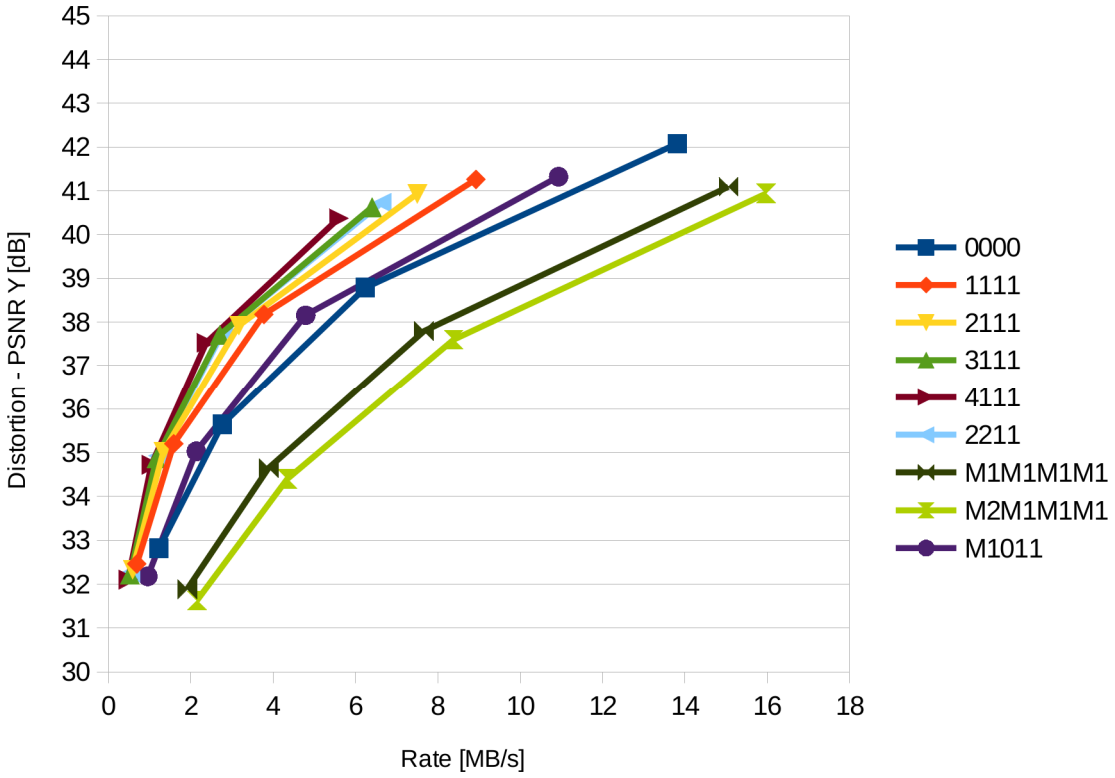

Figure 39: Measurement results for QP deltas (BigBuckBunny sequence, initial QP range: 22-37)

Mode	BD rate	Color	
0000	Reference	White	*
1111	-31.3%	Lilac	
2111	-38.6%	Cyan	*
3111	-44.2%	Red	
4111	-48.2%	Yellow	
2211	-43.3%	Blue	*
M1M1M1M1	65.5%	Black	
M2M1M1M1	90.8%	Green	
M1011	-8.5%	Blue	*

Table 8: Measurement results for hierarchical coding with QP delta configuration (Bjøntegaard delta rate values; negative values showing gain) (BigBuckBunny sequence)

The above test gives a number of candidates to follow up on. The ones considered interesting for further study are marked with an asterisk ("*") in Table 8.

As expected, all modes tested result in well-defined rate-distortion curves. Using the dark blue curve (0000 mode) as reference, it is obvious that the 1111 through 4111 and the 2211 modes perform better, while the M1M1M1M1 and M2M1M1M1 modes perform much worse, even more when compared to the 1111 and 2111 modes, respectively. The M1011 mode even outperforms the 0000 mode with a gain of 7.4%. It was expected and confirmed by the test that the 3111 and 2211 modes perform very similar,given their very similar QP deltas. Visually the 2211 mode seems to be marginally better, but the visual difference between any of the modes is only noticeable when using seven temporal layers. Seven temporal layers represent an equivalent GOP length of 64 frames and a difference between the best and the worst layer of up to 9 QP values (4111 mode). Visual evaluation of seven temporal layers, even for the 1111 mode, shows a noticeable negative impact on visual quality and was not pursued any further. For practical reasons all evaluation will be done for four visual layers.

In the following section, the four modes 0000, 2111, 2211 and M1011 will be evaluated further.

Mode 0000

Table 9 shows the distribution of frames per layer vs. the distribution of kBytes/Layer (Rate). Figure 40 (a) shows the frame sizes after compression for the first 1000 frames of the Big Buck Bunny sequence. Figure 40 (b) is exactly the same as Figure 40 (a) but annotated to show the individual layers.

Layer	No. Frames	kBytes / Layer	Avg. kBytes / Frame
L0	25	2953	118
L1	25	2295	92
L2	50	3518	70
L3	100	5537	53

Table 9: Layer overview for mode 0000 (200 frames)

Figure 40 shows that the sequence has a fade-in (Frames 0 to 100), followed by a relatively steady section. There are small scene complexity changes near frames 300 and 800. Two large scene complexity changes occur near frames 400 and 600. Between the large scene complexity changes is a relatively simple scene as indicated by the low data rate, while before and after are more complex scenes.

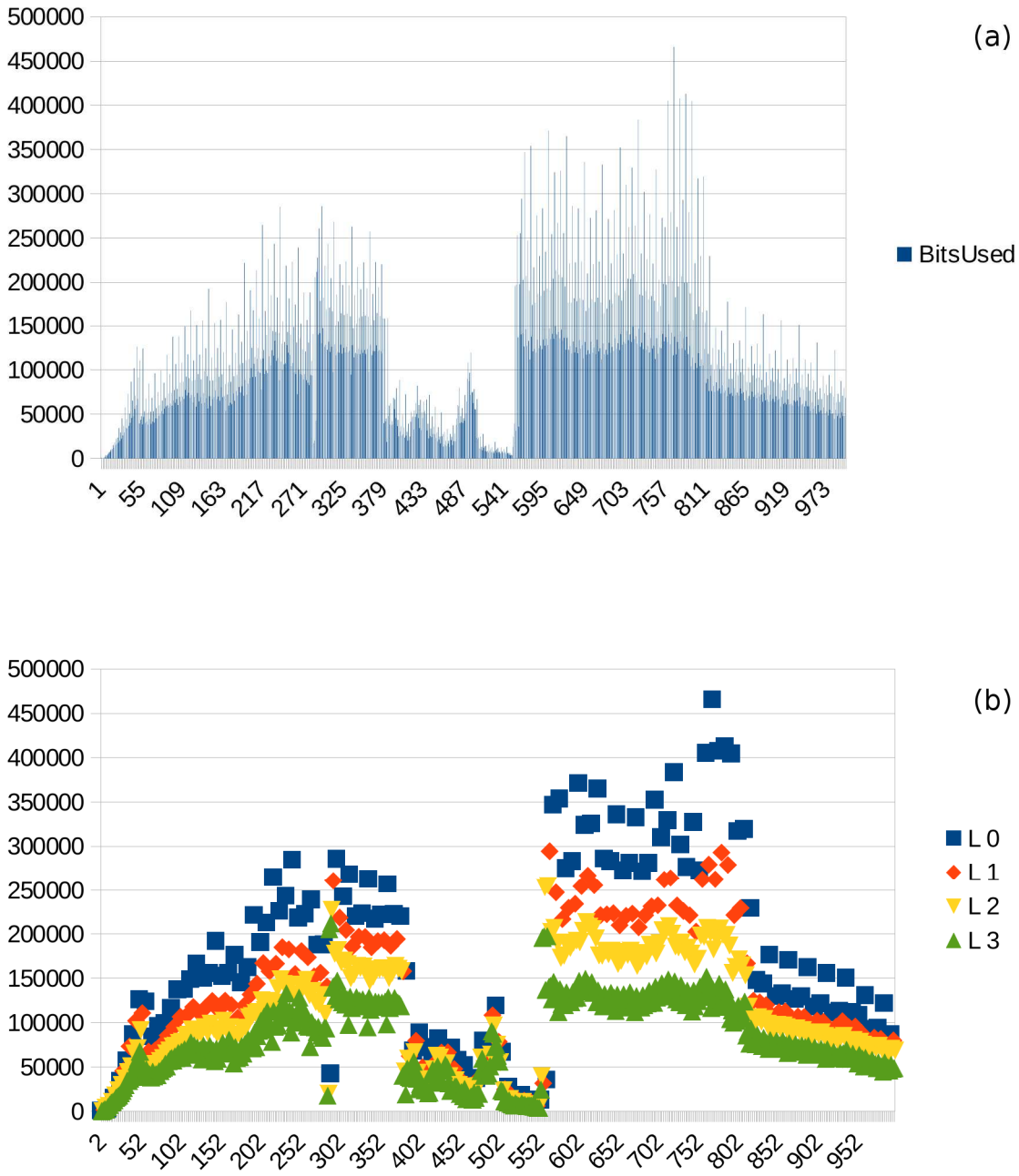

Figure 40: Results for QP delta in mode 0000

In this test, all layers are compressed with the same QP, the differences between layers is due to two separate effects. One effect is an increased number of Intra coded blocks in Layer 0. This is due to the fact that layer 0 frames can only be predicted from the last layer 0 frame which is relatively far away (8 frames previous). Along similar lines, layer 1 frames can only be predicted from previous layer 0 (4 frames previous) or previous layer 1 (8 frames previous). This does result in a lower bitrate compared to layer 0 as the Inter mode prediction is better, though not ideal. Layers 2 and 3 benefit from frames available for prediction that are increasingly closer, thus reducing the data rate noticeably and systematically.

The results are usable for rate control implementation, however not ideal for the aforementioned network transmission.

Mode 2111

Table 10 shows the distribution of frames per layer vs. the distribution of kBytes/Layer (Rate). Figure 41 (a) shows the frame sizes after compression for the first 1000 frames of the Big Buck Bunny sequence. Figure 41 (b) is exactly the same as Figure 41 (a) but annotated to show the individual layers.

Layer	No. Frames	kBytes / Layer	Avg. kBytes / Frame
L0	25	2953	118
L1	25	1536	61
L2	50	1693	34
L3	100	1800	18

Table 10: Layer overview for mode 2111 (200 frames)

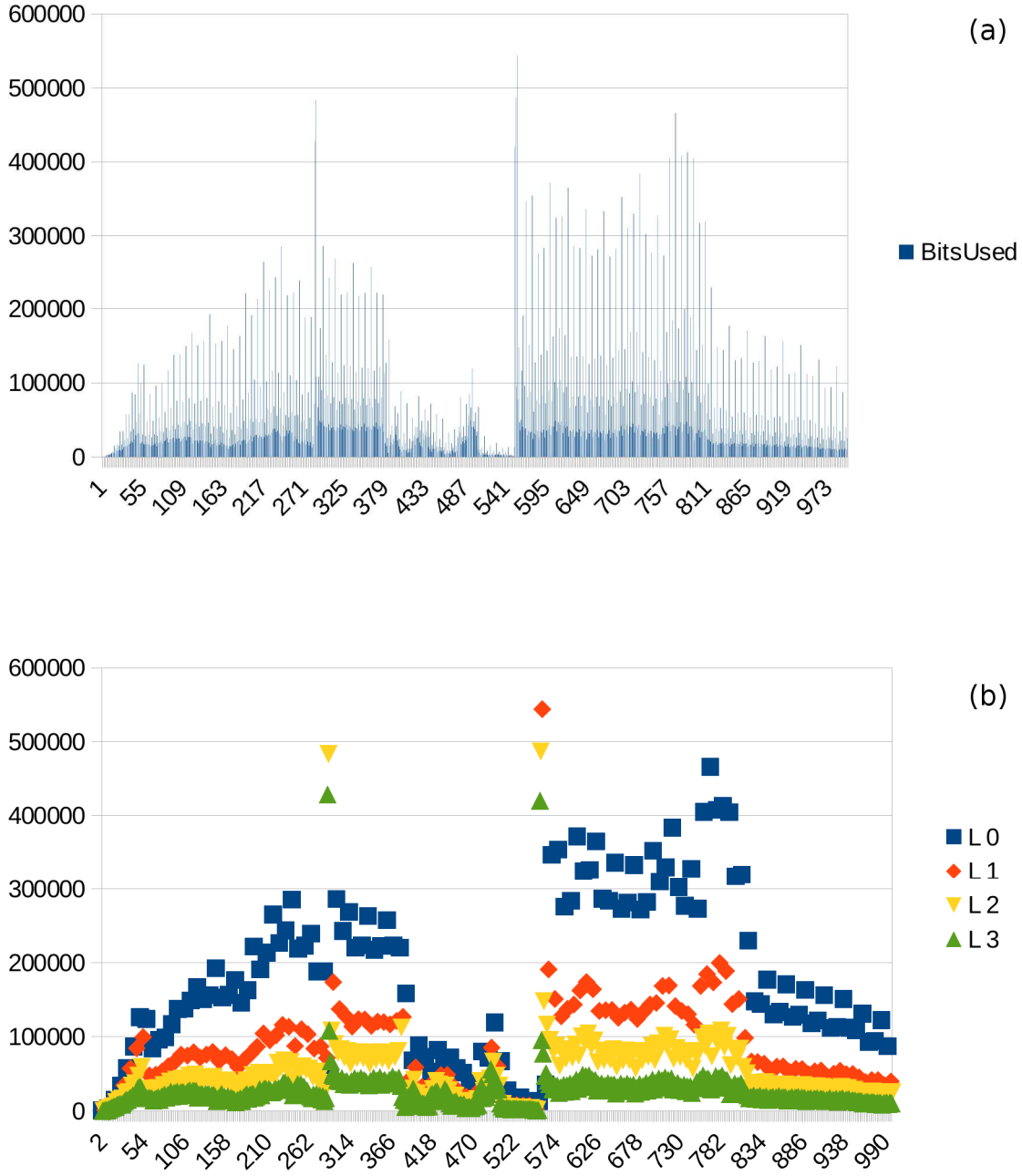

Figure 41: Results for QP delta in mode 2111

Beginning with this mode, the big scene complexity changes near frames 400 and 600 result in overly large frames. In this case it is obvious that the scene complexity change happens compressing a layer 3 frame. As layer 3 is not used for prediction, the same content has to be coded again for the next frames in layers 2, 1 and 0, unfortunately. A strategy to handle such scene complexity changes has been developed and is discussed in chapter 4.3.5.4.

Otherwise the performance of this mode is somewhat worse than mode 0000 as far as rate control and network transmission are concerned.

Mode 2111

Table 11 shows the distribution of frames per layer vs. the distribution of kBytes/Layer (Rate). Figure 42 (a) shows the frame sizes after compression for the first 1000 frames of the Big Buck Bunny sequence. Figure 42 (b) is exactly the same as Figure 42 (a) but annotated to show the individual layers.

Layer	No. Frames	kBytes / Layer	Avg. kBytes / Frame
L0	25	2953	118
L1	25	1536	61
L2	50	1216	24
L3	100	1353	14

Table 11: Layer overview for mode 2211 (200 frames)

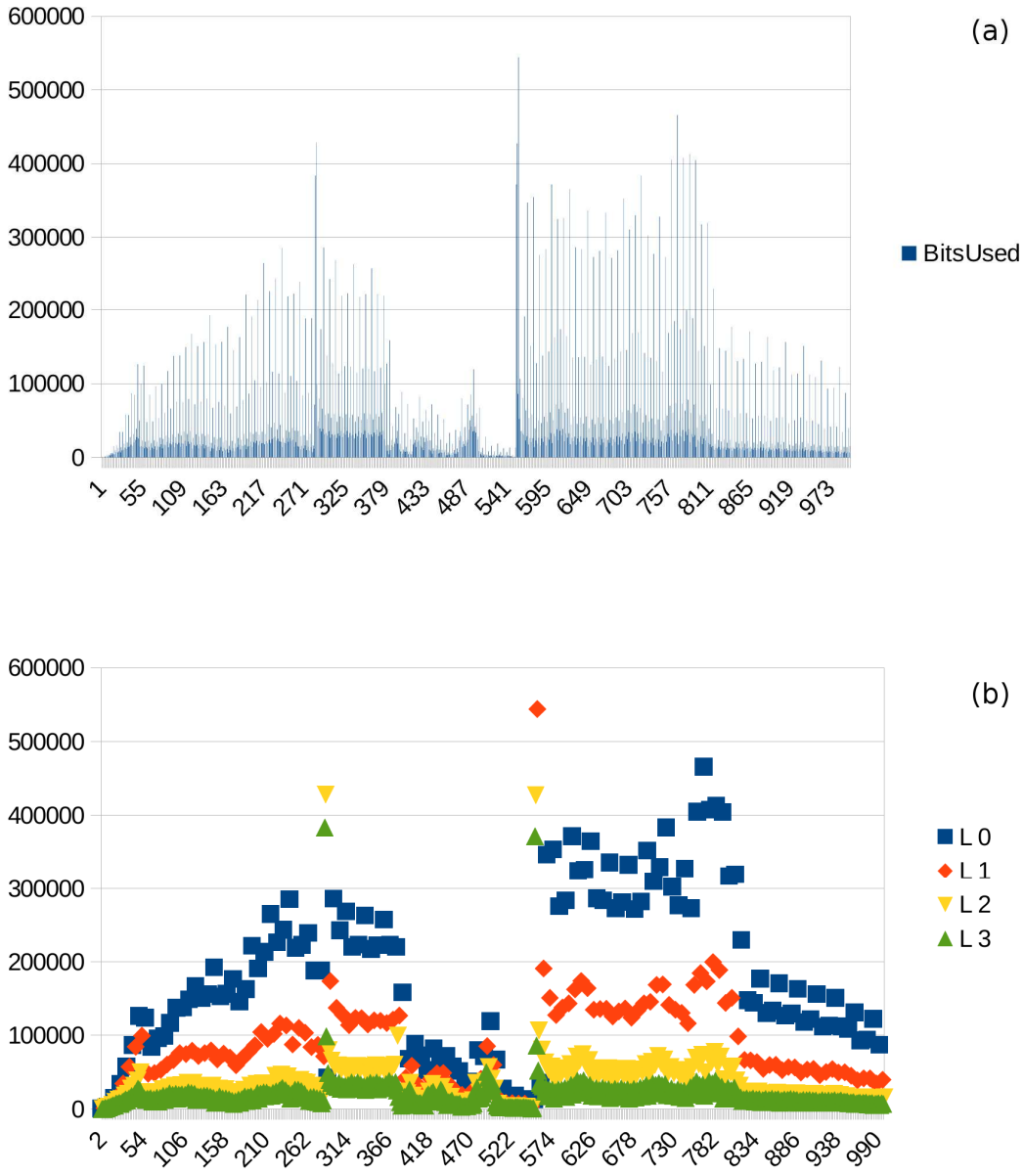

Figure 42: Results for QP delta in mode 2211

As seen in mode 2111, the big scene complexity changes near frames 400 and 600 result in overly large frames. In this case it is obvious that the scene complexity change happens compressing a layer 3 frame. As layer 3 is not used for prediction, the same content has to be coded again for the next frames in layers 2, 1 and 0, unfortunately. A strategy to handle such scene complexity changes has been developed and is discussed later.

Otherwise the performance of this mode is decidedly worse than modes 0000 and 2111 as far as rate control and network transmission are concerned.

Mode M1011

Table 12 shows the distribution of frames per layer vs. the distribution of kBytes/Layer (Rate). Figure 43 (a) shows the frame sizes after compression for the first 1000 frames of the Big Buck Bunny sequence. Figure 43 (b) is exactly the same as Figure 43 (a) but annotated to show the individual layers.

Layer	No. Frames	kBytes / Layer	Avg. kBytes / Frame
L0	25	2582	103
L1	25	2473	99
L2	50	2889	58
L3	100	3363	34

Table 12: Layer overview for mode M1011 (200 frames)

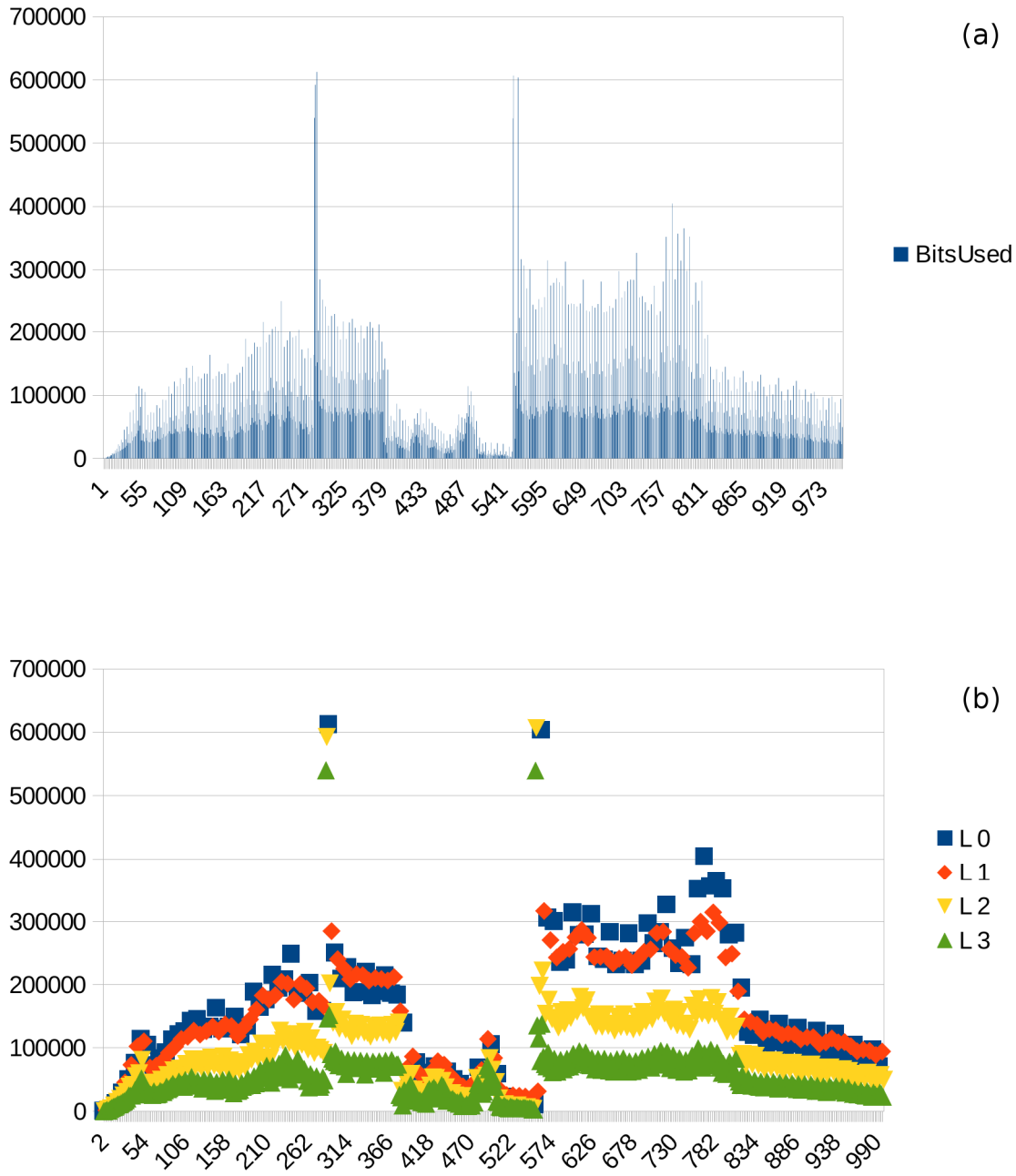

Figure 43: Results for QP delta in mode M1011

As seen in modes 2111 and 2211, the big scene complexity changes near frames 400 and 600 result in overly large frames. In this case it is obvious that the scene complexity change happens compressing a layer 3 frame. As layer 3 is not used for prediction, the same content has to be coded again for the next frames in layers 2, 1 and 0, unfortunately. A strategy to handle such scene complexity changes has been developed and is discussed in “Scene complexity change detection”.

However, testing over several sequences and with different settings for the Base-QP showed that the layers each make up roughly a quarter of the bit rate (compare Table 12). Layer 0 and layer 1 frames are virtually the same size, while layer 2 frames are roughly half the size and layer 3 roughly half again. As seen in Figure 43 (b), the differences between successive frames are very well controlled and there are no overly large frames apart from the scene complexity changes. This is a very good result regarding the rate control algorithm and regarding network transmission.

Conclusion

The requirements for the rate control, predictable frame sizes, and for network transmission, similar frame sizes throughout the layers, are considered while evaluating.

While mode 0000 is a most significant improvement over not using temporal layers, the variation of frame sizes in layers 0 and 1 is still quite obvious and it will be difficult to fulfil the requirements with this mode.

In comparison, the rate-distortion performance of the 2211 mode is among the best in the overall test set, but this mode has a rather large variation of frame sizes in layers 0 and 1. Also, the average layer 0 frame is quite a bit larger than the frame sizes for layers 2 and 3, rendering mode 2211 unfavourable with regard to the network transmission requirement.

Overall the mode 2111 performs slightly worse than mode 2211. The frame sizes are slightly better controlled, the Rate-Distortion performance is slightly worse, but not significantly so.

In contrast, mode M1011 meets the requirements very well. There is little variation of frame sizes in layer 0, which is small compared to the other modes, while the layers 1, 2 and 3 have next to no variation. Also, both layer 0 and layer 1 frames are similar in size, meeting the network transmission requirement not to have overly large frames. In fact each layer takes up roughly a quarter of the overall data rate, so over a group of pictures there is very little variation in data rate.

The behaviour of mode M1011 was verified with several sequences and several different base-QP settings. It is very consistent, apart from the fact that all other sequences show a less pronounced structure, with no scene complexity changes or fade-in. Also, the relations between layer or frame sizes are very similar for all tested sequences and base-QP settings.

Although there may be some room for optimization, for the development of the rate control algorithm mode M1011 is chosen. At the same time, the Big Buck Bunny sequence is considered most challenging for the rate control tasks as it contains a number of different content properties (e.g., steady content, panning, fades, scene complexity changes), resulting in an difficult task for the rate control algorithm.

5.3 Annex 3: Rate control for HEVC video compression

Before starting on the rate control, a large number of sequences were analyzed with respect to frame sizes during encoding. The result was a noise-like graph with little or no discernible dependency between successive frames. Nothing statistically relevant was found to base a rate control on.

The test was repeated using hierarchical temporal prediction, with a much more satisfying result (see “0000” results above). However, it was obvious that the QP deltas between layers needed optimization. The “M1011” mode was chosen successively and used for rate control development. Using these settings, the frame sizes output by the encoder are predictable in the short term.

The rate control has the following interaction points with the encoder:

- Setting configuration parameters.
- Before starting to encode a frame, the rate control is queried for the frame's initial QP and the maximum compressed frame size limit.
- At the end of a frame, the number of bits used is given to the rate control.

Use of the encoder in real-time communications on mobile devices requires careful optimization of computational load and memory access throughout the encoder. A complexity analysis of every frame before the encoding of the frame was considered not an option due to excessive memory access.

Frame-based rate control

A first, frame-based algorithm for rate control was implemented as a starting point for the rate control implementation. There was no expectation that the frame-based algorithm would be useable without further modification. Surprisingly, with the finely tuned hierarchical prediction, the frame-based algorithm proved to be quite useable, especially within a scene that is typical for the primary use case (e.g., streaming of ultrasound and ambient video).

A number of restrictions, simplifications, and assumptions are made for the first implementation:

- The encoder is operated in the hierarchical prediction mode “M1011.”

- No rate control within a frame, but only at the start of a frame. Thus, the QP will be fixed per frame.
- A fixed frame rate (via configuration parameter) is assumed.
- A violation of the maximum rate is tolerated for short periods of time.
- A maximum quality (minimum QP) is set via configuration parameter.
- The initial QP is set manually to a value that resembles a good starting point (startup problem).
- Overhead for video stream multiplexing and network packaging is not considered.

The frame-based algorithm uses data from previously encoded frames to estimate the current output rate. Based on the estimation the QP is changed to decrease or increase the output rate. The rate control algorithm employs a window of acceptable rate within which no changes to the QP are made, as well as different settings for increasing vs. decreasing the QP in order to reduce oscillation.

Obviously, at the start of a video sequence, there is little or no data from previously encoded frames. As long as the assumption about the initial QP is met, the algorithm can gather data for the period of several frames until the effect of regulation is required.

The crucial part of the algorithm is the estimation of the current rate. To do so, the algorithm uses a weighted sliding window average calculating the current average frame size. For each frame, the following formula is calculated:

$$(2) \qquad A_n = A_{n-1} * (1 - c) + (S_n * c),$$

with A_n being the weighted average after frame *n*, A_{n-1} the weighted average after the previous frame, S_n the reported size of frame *n*, and *c* a weighting parameter calculated from frame timing information. The weight *c* is called the window size. After optimizing the weight, this results in a fairly stable and robust estimation of the current frame size.

The actual output rate is estimated by multiplication of A_n with the nominal frame rate (see assumptions).

Since the rate is measured in bit per second, not in frames per second, the frame timing information is important. When replaying previously recorded video, as in the tests executed during the rate control algorithm development, a fixed frame rate can safely be assumed. In case of real-time communications, however, the input frames may or may not be recorded at a fixed rate. A very typical example is a video taken in low-light conditions. Many modern cameras will use a longer shutter time in low-light conditions, resulting in a lower number of frames per second. In case of a non-steady frame rate, *c* should be adapted to reflect the frame rate.

Please note that the current rate is calculated by the rate control over a very short window size of 8 frames (1/3 second) to allow a fast reaction to changes. The window size is chosen to coincide with the GOP size resulting from four temporal layers being used.

The current output rate is compared successively to the maximum output rate and to the minimum output rate set as configuration parameters.

If the current output rate is within the bounds of minimum and maximum rate, nothing is done.

If the current output rate is higher than the maximum rate, the QP is increased by a fixed amount set via the “strength” configuration parameter.

If the current output rate is lower than the minimum rate, the QP is decreased by a fixed amount one half of the “strength” configuration parameter.

Adjustments to the QP are made in fractional increments, for which the rate control internally uses a floating-point representation of the QP.

With the encoder finely tuned as described above, the frame-based rate control actually works quite well in most cases. The resulting rate is well controlled, even when using the “BigBuckBunny” sequence, which has a number of scene complexity changes challenging the rate control algorithm. Evaluation of sequences which are less demanding on the rate control result in a reasonably well controlled output rate, comparable to the results between frame 650 and 1000. The algorithm is able to handle small to medium changes in scene complexity reasonably well, while medium to large changes in scene complexity (at the start of the sequence, near frames 300 and 600) cause some rate overruns but are reasonably quickly compensated for.

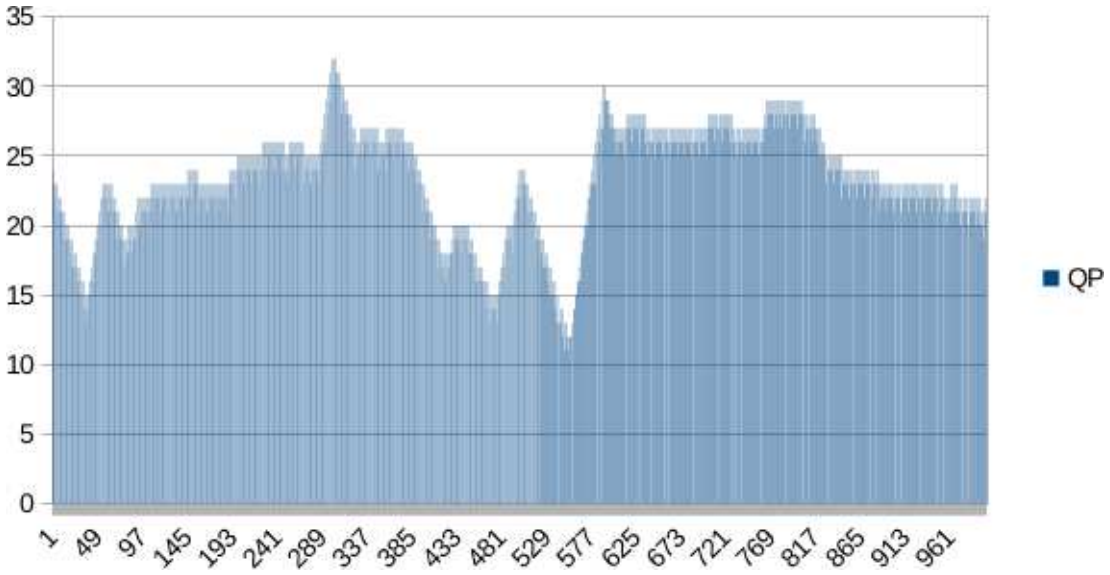

Figure 44: Results of frame-based rate control – QP, not compensated for QP changes due to the layer settings (BigBuckBunny sequence, first 1000 frames, initial QP = 22)

At the start of the sequence, the frame sizes (i.e., the data rates for the single frames) are very low (since the sequence contains a fade-in from black), in fact lower than the minimum rate set via configuration parameter. This increases the severity of the startup problem as the data rate observed by the rate control algorithm is much lower than the lower boundary while the statistical data available is not yet sufficiently stable. Obviously, the rate control tries to compensate a few frames into the sequence, as indicated by the change in QP in Figure 44. As the scene becomes more complex and the rate control is still improving the quality, there is a single oscillation with a peak near frame 50. Decreasing the QP by half the value (half the speed) used for increasing it proved to be vital to control the oscillation, as proved the use of a window defined by minimum and maximum data rate. There is a scene complexity change near frame 300 resulting in a short peak rate. Earlier testing indicates this frame change impacts three subsequent frames on different temporal layers, which needs to be addressed via a different mechanism, called scene complexity change detection. Please refer to the scene complexity change discussion below. However, the rate control algorithm reacts to the change starting with the very next frame and limits the rate overrun to approximately 25 frames. Thus, the requirement for overruns not exceeding a second is met, even under worse than normal circumstances. The subsequent scene requires a low bit rate, resulting in a fast reaction of the rate control algorithm. Some level of oscillation is encountered near frame 500. Another scene complexity change near frame 575 again impacts three subsequent frames on different temporal layers. Recovery from the second scene complexity change proves to be difficult for the rate control, especially with the low-rate scene before and the higher-rate scene after the scene complexity change. The rest of the sequence, which has some slow variation in scene complexity (rate requirements) is handled very well by the rate control.

Obviously, different values for the strength parameter results in different speeds of the rate control reaction. While a “strong” value will give speedy recovery after a scene complexity change, the rate control will oscillate during periods of slow changes with additional rate overruns due to the oscillation. On the other hand, a “weak” value will result in overly long recovery times but less or no oscillation.

The achievement of the frame-based rate control can be summarized as follows:

- The startup problem is very obvious in this sequence, i.e. the initial rate is outside the rate control window and the statistics of the rate control are not yet stable.
- Scene complexity changes (impacting multiple frames) are handled, recovery could be faster.
- There is a tendency to oscillate if the change in frame size is larger than the reaction of the rate control.
- The rate control is sufficiently stable for slow changes in the scene complexity.
- In order to improve the rate control, several changes will be made to the algorithm in the next stage.

Improvement of initial rate control algorithm (“iteration 2”)

As a second iteration for the development of the algorithm, a lot of modifications of the rate control algorithm were evaluated. Some of the changes did not yield improvements and were discarded immediately. For example, changing or

removing the “minimum bit rate” from the rate control calculations did not lead to positive results. The “minimum bit rate” was thus restored and fixed to 80% of the “maximum bit rate” which generally results in optimum behavior.

In the above examples, the “big” scene complexity changes occurs in a frame encoded on temporal layer 3. Due to the prediction structure, the changed scene has to be encoded on temporal layers 3, 1 and 0 consecutively. This does result in three very large frames inside less than eight frames, which impacts the rate and rate control algorithm negatively. While this kind of scene complexity change is not considered to occur frequently in CONCERTO use cases, a vehicle like an ambulance moving suddenly from full sunlight into a tunnel will result in massive changes in the video recorded inside the vehicle and can be considered a scene complexity change. The penalties encountered by a large scene complexity change are very disruptive to the video transmission and to the viewing of the video. A basic scene complexity change detection and handling is discussed in this section. As a first step, a mechanism to estimate the frame size of the next encoded frame is built into the rate control. The estimated frame size is multiplied by a constant factor in order to provide an upper limit for the next frame, triggering the scene complexity change detection.

To calculate the frame size limit, the relation between average layer and frame sizes from Table 10 is used to initialize the frame sizes for each layer at the start of encoding a sequence. This data has been verified on the large test set with many sequences and several initial QP settings to be valid within rounding error. The algorithm described in (2) is used to keep track of current frame sizes once per temporal layer. During testing, the average layer and frame sizes for the different QP delta modes were tested for a large number of sequences. For the mode M1011, which has been chosen successively, the variation in relative layer and frame sizes is low. This allows using fixed values to initialize the estimation for the temporal layers.

To further improve the responsiveness to rate overruns, two strategies proved useful. The rate overrun is calculated as a factor relative to the maximum bit rate setting, according to (3). The increase in QP, as described for the frame-based rate control algorithm, is multiplied by the factor, resulting in a large QP increase in case of a large bit rate overrun and a small increase in case of a small bit rate overrun. This will lead to a very quick response to rate overruns. After tuning the strength parameter, the response to a bit rate overrun is an estimated two times faster than with the first implementation.

$$\text{factor} = \text{current rate} / \text{maximum bit rate} \quad (3)$$

To allow a more fine-grained response, in case of a severe bit rate overrun (the values for the decision still needing to be optimized), in addition to increasing the overall QP, the innermost temporal layer is applied an extra +1 penalty on the QP, momentarily reducing the rate. When viewing the resulting video sequence, this is not noticeable with the human eye; however it helps in dumping excessive bit rate.

With both improvements the bit rate is controlled well with very little oscillation. With the improvements in the rate control algorithm the QP before the scene complexity change is even lower than with the first implementation, due to better regulation, thus leading to a larger peak. There is a single oscillation after the peak. While this impacts the visual quality of the video negatively, it also allows for recovery in case of packet loss due to the excessively large frames at the scene complexity change.

Strength parameter	PSNR [dB]	Average Rate [kbit/s]	Number of rate overruns	Average rate overrun	Maximum rate overrun
1/8	**36.51 dB**	**427 (85%)**	**97 (9.7%)**	**+250%**	**+969%**
2/8	**36.60 dB**	**435 (87%)**	**72 (7.2%)**	**+98%**	**+533%**
3/8	**36.63 dB**	**437 (87%)**	**35 (3.5%)**	**+23%**	**+124%**
4/8	**36.86 dB**	**440 (88%)**	**17 (1.7%)**	**+22%**	**+54%**
5/8	**36.83 dB**	**440 (88%)**	**20 (2.0%)**	**+24%**	**+105%**
6/8	**36.81 dB**	**441 (88%)**	**22 (2.2%)**	**+30%**	**+110%**
7/8	**36.96 dB**	**443 (89%)**	**19 (1.9%)**	**+27%**	**+110%**
8/8	**36.83 dB**	**443 (89%)**	**19 (1.9%)**	**+28%**	**+99%**

Table 13: Rate overrun and distortion variation over changes in strength parameter for Ambulance sequence (GOPR0046).

A test run with different settings for the strength parameter was done, and the rate overrun estimated. In Table 13, the average rate was compared to the target rate of 500 kbit/s. As expected the rate control regulates towards the center of

the window between minimum (80%) and maximum (100%) bit rate. Increasing the minimum rate results in a higher average rate but also more rate overruns.

A rate overrun is counted any time the current rate, measured after encoding every frame, using an 8 frame window, exceeds the maximum rate plus the 10% allowance as stated in the requirements. With a low strength parameter, the reaction to changes in encoded image size is obviously too slow. Setting the strength parameter to 4/8 or higher results in a stable amount and size of rate overruns. Analysis of the correspondence between frame sizes and overruns shows that the overruns occur at encoding a single large frame. Some of these large frames can be considered a scene complexity change and will be treated by a separate mechanism, see the discussion of scene complexity changes below.

The results from the last test run show two effects: at a low strength parameter, obviously the rate control reaction to a change is too slow, both in limiting rate overruns as well as recovering from a low bit rate. With a high strength parameter however, the rate control tends to oscillate and over-compensate high or low bit rate. The strength parameter is used to balance both effects.

As stated above, recovery from low bit rate is at half the strength compared to the reaction to rate overruns. This proved essential in order to limit oscillation in the rate control implementation and has been verified during the iteration 2 work. Without the slower recovery from low bit rate, the maximum rate overruns tend to be much larger, similar to the results from a higher strength parameter, the maximum rate overruns increase in severity, but not in number.
Another test run with different settings for the maximum rate was done to evaluate whether the rate control is impacted by the range of QP being used for different output rates. The results were similar throughout.

Finally some effort was spent to simplify the rate control calculations, which has no discernible effects on the behavior of the rate control algorithm, other than speeding up the internal calculations.

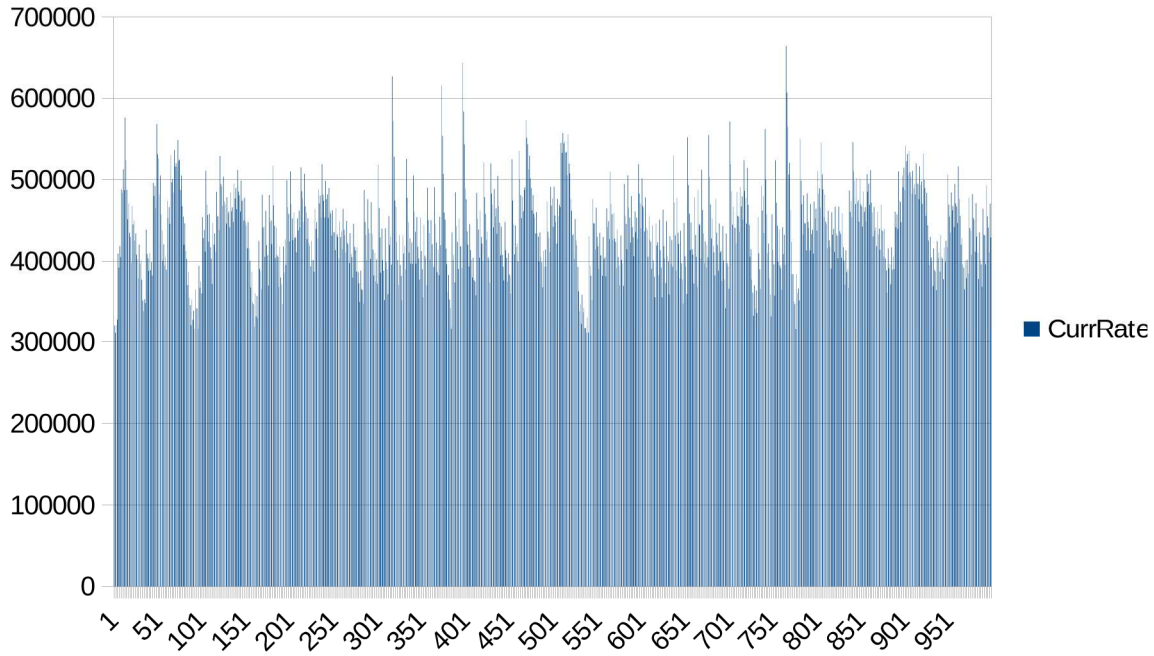

Figure 45: Resulting rate with iteration 2 rate control for Ambulance sequence (GOPR0046).

Figure 45 shows the rate control's results for the "Ambulance" sequence, a sequence recorded during the video acquisition event in Perugia. Please note this sequence was previously compressed with H.264 (Main Profile). As of yet no detailed analysis about the interaction between H.264 and HEVC is available, so the effect of the previous compression cannot be quantified. Visual examination of both the input and output videos does not show any obvious differences. Please note that the "current Rate" shown in the diagram is calculated by the rate control over a very short window size of 8 frames (1/3 second) to allow a fast reaction to changes. Evaluation with a one second window size results in a very smooth curve after the startup has been handled. The rate overrun evaluates to less than one per cent. Evaluation with more and different sequences produces similar results.

Scene complexity change detection

A “real” scene complexity change detection requires analyzing the image content of successive video frames. For reasons of complexity and run-time, this is not feasible in the scope of the current project.

However, a strategy to detect and handle scene complexity changes has been identified in the course of the work and will be implemented later on. The strategy is based on a feature of the rate control algorithm (frame size limiting) and the observed behaviour of the encoder to generally output very large frames in case of a scene complexity change.

The rate control algorithm calculates a frame size limit at the start of encoding every frame. If a frame proves to be larger than the limit for the encoded frame, the encoder will drop/skip the frame. It will be assumed that this is a scene complexity change frame.

The dropped frame results in a saving in bit rate, which is used for the next frame. It is assumed that all buffers (streaming and network) are operating at a medium fill level, i.e. the output rate of the encoder is less or equal to the network’s capability. Likewise, a large frame occurring during a scene complexity change will increase the buffer fill level considerably. Similarly, the rate under-run observed just after a scene complexity change (a large rate overrun) occurred will allow the full buffers to drain somewhat and is considered to be somewhat helpful. It also prepares the rate control to handle a scene with higher complexity that may be subsequent to the scene complexity change.

When encoding the next frame after the dropped frame, several actions are taken by the encoder logic. The next frame after a scene complexity change will be coded as temporal layer 0, in order to be available for prediction, instead of having to code the changes several times as can be observed in the diagrams for the temporal layer and rate control chapters. Also, the frame will be encoded with a slightly reduced quality to conserve bit rate.

Depending on the coded size of the frame after the scene complexity change, it may be decided to also drop/skip the next following frame.

The light-weight scene complexity change detection has no influence on the encoding speed, next to no influence on the encoder logic and shifts the bit-rate allocation in favour of a high-priority “Intra” frame (in the discussed case it is temporal layer 0).

In case of offline-encoding, which is outside the scope of the current work, dropping a frame may not be an option. It is possible to handle a scene complexity change by sacrificing encoder speed. The simple solution is to re-encode the overly large frame as a temporal layer 0 frame, as outlined above.

Conclusion

Considering the fact that the rate control algorithm does not change the QP inside a frame, but only at the end of a frame, the rate control algorithm is working reasonably well. The BigBuckBunny sequence was chosen as the major test sequence during development because it has some rather large variation in scene complexity resulting in very obvious rate changes. Checks with different sequences produced good results throughout but changes tend to be too small to be significant. There is some room for optimization, especially in handling the big scene complexity changes that currently affect three frames, but overall the results are good enough for practical use.

6 References and glossary

6.1 References

[1] ISO/IEC 23000-10:2012, Information technology – Multimedia application format (MPEG-A) – Part 10: Surveillance application format, Oct. 2012.

[2] P. Helle, S. Oudin, B. Bross, D. Marpe, M.O. Bici, K. Ugur, J. Jung, G. Clare, and T. Wiegand, "Block Merging for Quadtree-Based Partitioning in HEVC," IEEE Trans. On Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1720—1731, December 2012.

[3] HEVC test model HM-14.0, ITU-T VCEG and ISO/IEC MPEG (JCT-VC), Online available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/

[4] F. Bossen. "Common test conditions and software reference configurations," ITU-T VCEG and ISO/IEC MPEG (JCT-VC) document JCTVC-L1100, Geneva, Switzerland, January 2013.

[5] H. Schwarz, D. Marpe, T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard", in IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, no. 9, pp. 1103-1120, September 2007.

[6] http://www.bigbuckbunny.org

[7] Sun Microsystems. "XDR: External Data Representation Standard," IETF RFC 1014, June 1987.

[8] PWC, "Touching lives through mobile health, Assessment of the global market opportunity," Feb-2012. [Online]. Available: www.pwc.com/mhealth.

[9] CONCERTO consortium, "D2.1: Use cases, usage models and related requirements", version 1.0, November 2012.

[10] CONCERTO consortium, "D2.3: Cross-Layer signalling", version 2.0, April 2014.

[11] CONCERTO consortium, "D2.4: System architecture recommended definition", version 1.0, August 2014.

[12] CONCERTO consortium, "D4.2: Intermediate evaluation and specifications of content adaptation solutions", version 2.0, July 2014.

[13] CONCERTO consortium, "D4.3: Final evaluation of context-awareness solutions", version 1.0, May 2014.

[14] CONCERTO consortium, "D6.2: Specification of the demonstrator", version 1.0, August 2013.

[15] CONCERTO consortium, "D6.3: CONCERTO Simulator: Final Architecture and Numerical Result", version 1.0, December 2014.

[16] CONCERTO consortium, "D3.3: Final description and evaluation of solutions for QoE-aware image/video coding", version 1.0, July 2014.

6.2 Glossary

AVC	*Advanced Video Coding*	
BD	*Bjontegard delta*	
CTU	*Coding Tree Unit*	
CU	*Coding Unit*	
DASH	*Dynamic Adaptive Streaming over HTTP*	
DDE	*Distributed Decision Engine*	*Event dissemination framework*
ECU	*Early CU (condition)*	*HEVC fast encoding mode*
ES	*Early skip (condition)*	*HEVC fast encoding mode*
FastQT	*Fast Quadtree*	*HEVC fast encoding mode*
GOP	*Group of pictures*	
HEVC	*High Efficiency Video Coding*	
IETF	*Internet Engineering Task Force*	
LAN	*Local Area Network*	
MPD	*Media Presentation Description*	
MPEG	*Moving Picture Experts Group*	
NAL	*Network Abstraction Layer*	
NIS	*Network Information Service*	
PoA	*Point-of-Attachment*	*Access point or base station*
PU	*Prediction Unit*	
PUS	*Prediction Unit splitting*	*HEVC fast encoding mode*
QP	*Quantization Parameter*	
RD	*Rate distortion*	
RDO	*Rate distortion optimization*	
RFC	*Request for Comments*	
RTP	*Real-time Transport Protocol*	
RTSP	*Real Time Streaming Protocol*	
SAF	*Surveillance Application Format*	
SDK	*Software Development Kit*	
STP	*Spatial or temporal prediction (condition)*	*HEVC fast encoding mode*
WLAN	*Wireless Local Area Network*	