EyesWeb XMI 5.2.0 – SIEMPRE Library

May 20, 2011

Part I

Description

The EyesWeb SIEMPRE Library includes a collection of software modules and patches which have been specifically designed and developed to support the requirements from the `EU ICT Siempre project`. In particular, the SIEMPRE Library integrates the EyesWeb platform with modules which are required for a distributed multimodal data recording and playback.

- The `SMPTE decoder` block received an audio signal in inputs and decodes the SMPTE timecode contained in the signal. See `SMPTE time code` and the references therin for a description of the SMPTE encoding schema.
- The `SMPTE encoder` block generates an audio signal with an encoded time code.
- The `Wave File Writer` block writes timeseries in the binary format chosen for the SIEMPRE recordings. The chosen binary format uses a small subset of the `Broadcast Wave Format` specifications.
- The `DeckLink input` block add support for the `DeckLink family of framegrabbers`, which are used in the Casa Paganini setup (see Acquisition setup at Casa Paganini) for synchronized audio/video input from High Definition (HD) videocameras.
- The `Qualisys SMPTE decoder` block which decodes the SMPTE timecode as packed in 32 bits by the software `Qualisys Track Manager` software. This block, toghether with the OSC support already available in EyesWeb, allows integration between EyesWeb and the Qualisys system. This provides the possibility to analyze, in real-time, the data tracked by the Qualisys Motion Tracking system. For the aim of the SIEMPRE project, the integration is particularly useful to be able to synchronize with the Qualisys Software to visualize the recorded data.

In the following chapters, some examples are described, which show the overall infrastructure used for many of the recordings which happens in the `EU ICT Siempre project` and the related EyesWeb patches.

Chapter 1

Example 1: String Quartet recording

The aim is to record audio, video, motion capture, and sensors data of performances of a String Quartet. The recording setup includes a Qualisys Motion Tracking system, one or more HD videocameras, ambient microphones and instrument microphones. The recording happens on three distinc PCs, and the synchronization of the data on the three computers is guaranteed by timestamping each sample with a reference SMPTE clock which is received by all computers (see Figure 1.1).

The three computers which are visible in the setup perform different operatorins:

- The black computer in the middle (A) records the Motion Tracking data. The same computer also operates as the master clock generator, i.e., it generates the SMPTE signal which is propoagated to the other computers.
- The lower-left computer (B) records audio/video data coming from the HD camera (JVC GY-HD-251 in this case).
- The lower-right computer (C) records audio data from the instruments.

EyesWeb and the SIEMPRE Library are used for several purposes in this setup.

In computer (A) a patch to generate the SMPTE signal on multiple audio channels is used. The patch si visible in Figure 1.2.

In computer (B) a patch to save audio/video data from the HD camera is used. The patch si visible in Figure 1.3.

In computer (C) a patch to save audio data from the microphones on the instruments is used. The patch si visible in Figure 1.4.

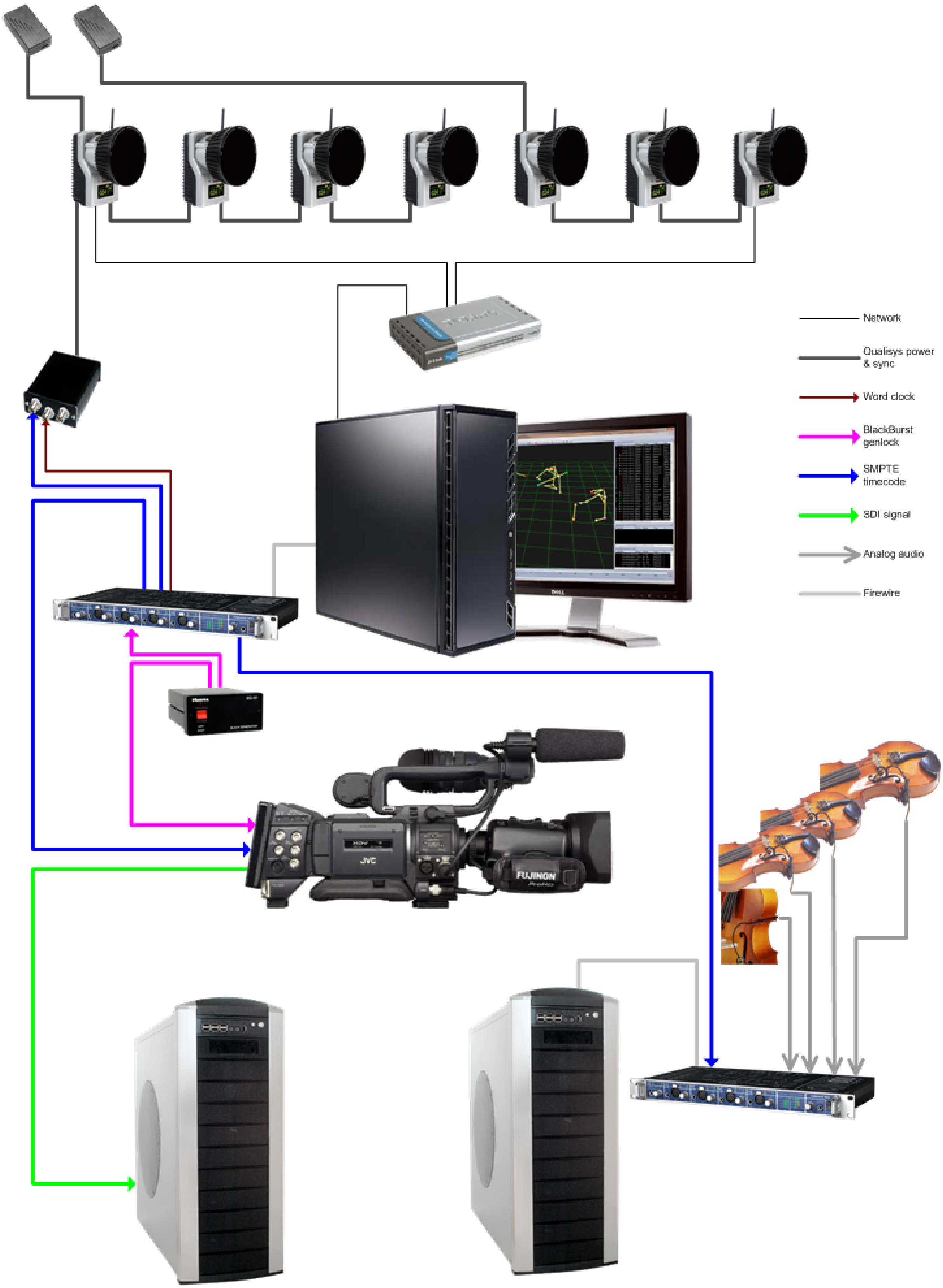
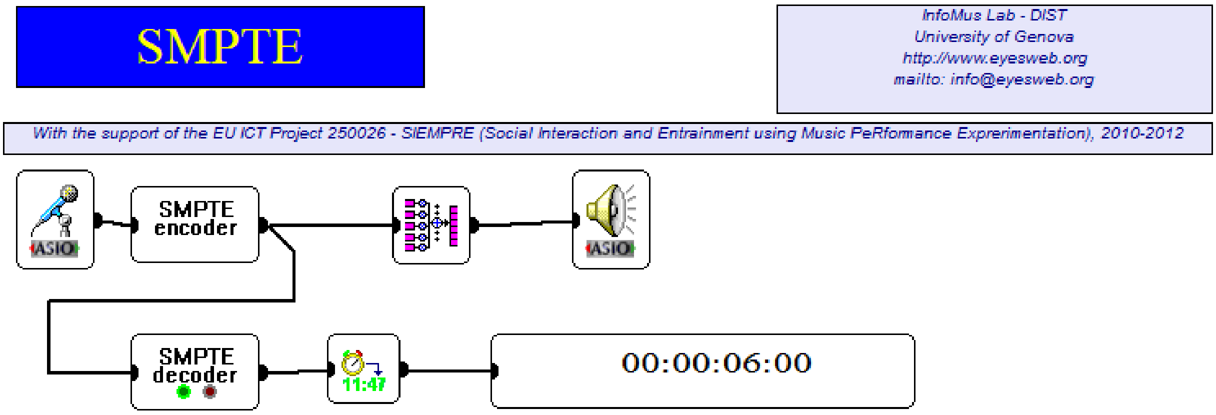


Figure 1.1: The acquisition setup installed at Casa Paganini, InfoMus Lab, Genova, Italy.

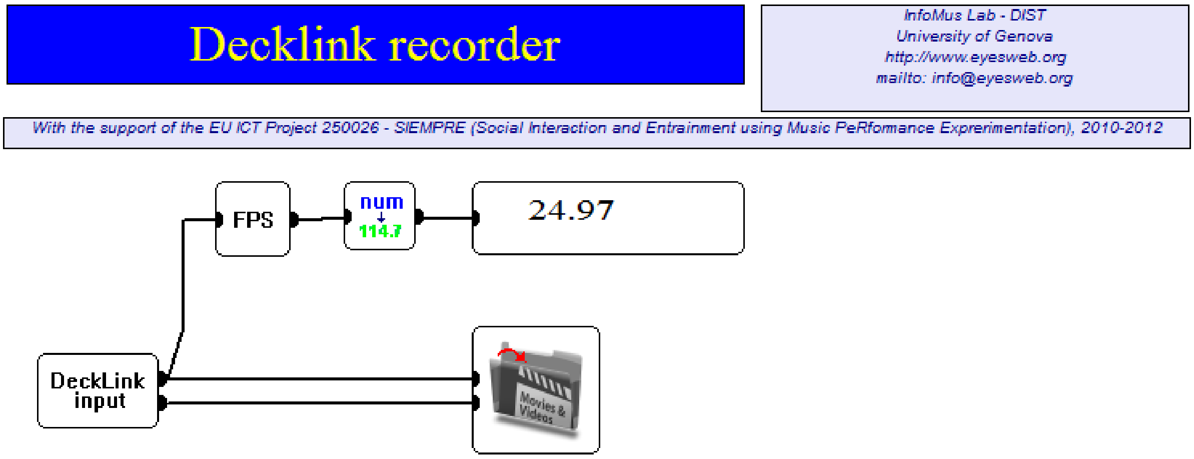


SMPTE

InfoMus Lab - DIST
University of Genova
http://www.eyesweb.org
mailto: info@eyesweb.org

With the support of the EU ICT Project 250026 - SIEMPRE (Social Interaction and Entrainment using Music PeRfomance Exprerimentation), 2010-2012

Figure 1.2: A patch to Generate the Smpte signal.

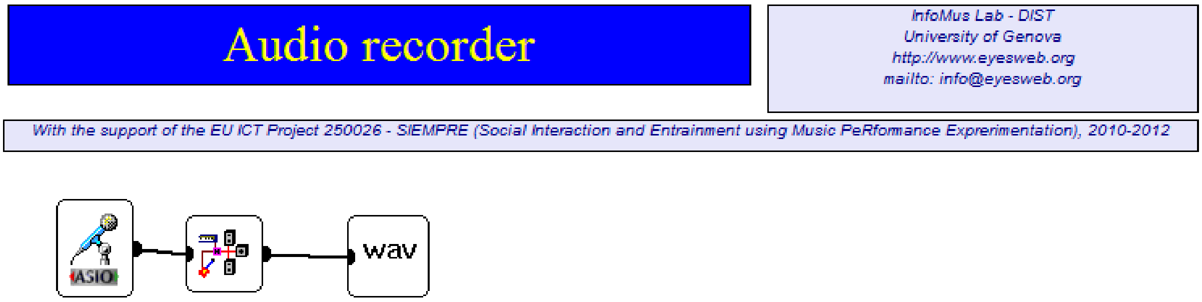


Decklink recorder

InfoMus Lab - DIST
University of Genova
http://www.eyesweb.org
mailto: info@eyesweb.org

With the support of the EU ICT Project 250026 - SIEMPRE (Social Interaction and Entrainment using Music PeRfomance Exprerimentation), 2010-2012

Figure 1.3: A patch to save the HD audio/video signal.



Audio recorder

InfoMus Lab - DIST
University of Genova
http://www.eyesweb.org
mailto: info@eyesweb.org

With the support of the EU ICT Project 250026 - SIEMPRE (Social Interaction and Entrainment using Music PeRfomance Exprerimentation), 2010-2012

Figure 1.4: A patch to save the multichannel audio signal from the microphones on the instruments.

Chapter 2

Qualisys integration

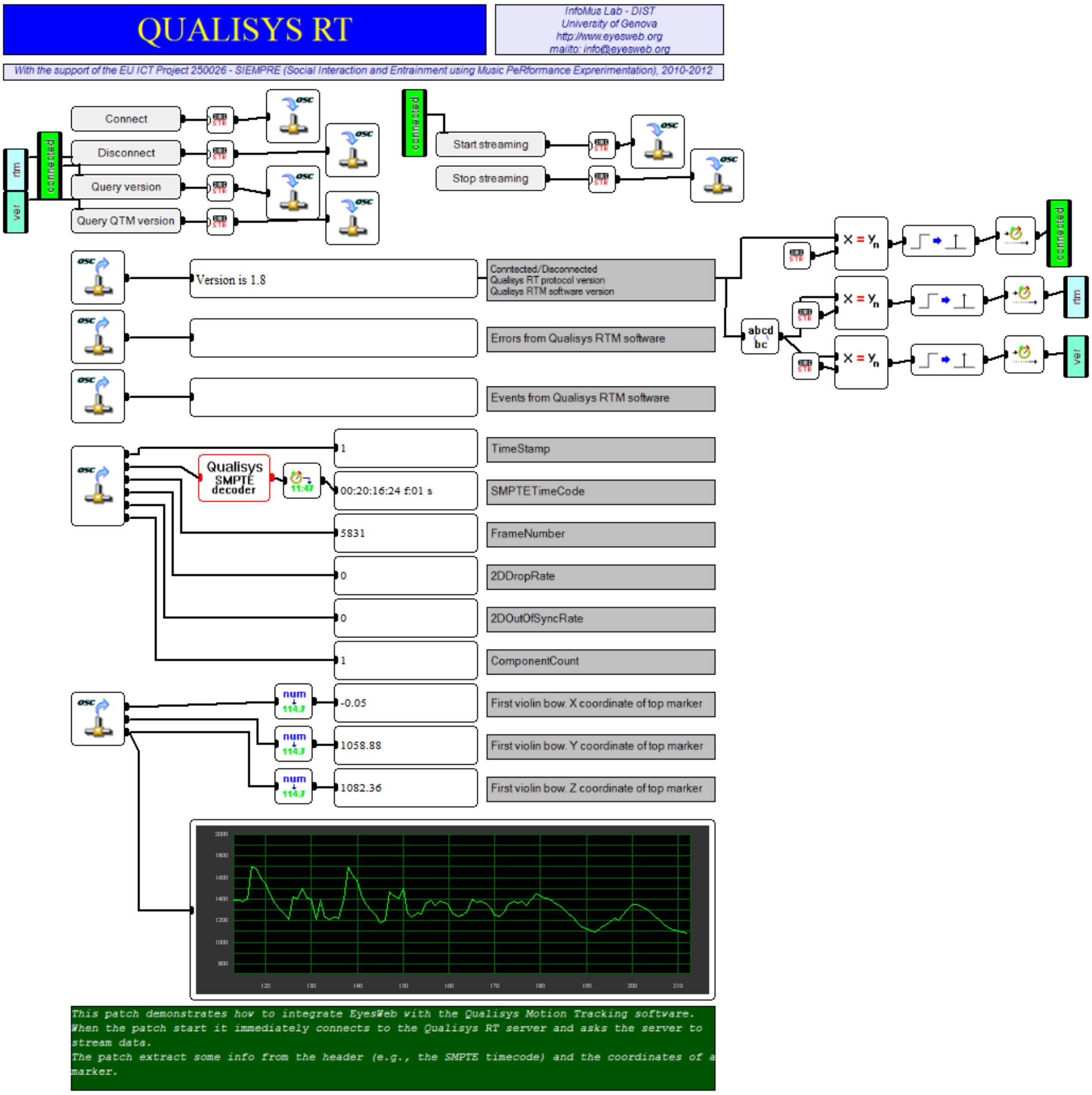
The SIEMPRE project is based on the use of the Qualisys Motion Tracking system to analyze the movements of the performers. EyesWeb has been extended in order to integrate with Qualisys and receive the tracking data in real-time. Figure 2.1 shows an example of an EyesWeb patch receiving motion tracking data in real-time. The Z coordinate of a marker placed on the top of a bow is received by the EyesWeb patch and plotted (see the graph in the lower part of the patch). Moreover, the patch also receives the SMPTE timecode from Qualisys; this timecode allows for synchronization of the processed data.

Another interesting integration example is given in Figure 2.2. In this example, the Smpte audio signal is generated by EyesWeb basing on the value received, via OSC, from the Qualisys RTM software. This mechanism may be used, for instance, to listen to the recorded audio tracks in sync with the playback of the recorded motion-tracking data.

As a matter of facts, many audio editors[1] allow for using the Smpte value as the reference to which all the audio data is synchronized. When the audio data is recorded, it is timestamped with the current Smpte value; when the same Smpte value is received in input, the corresponding audio data is played. Thus, by regenerating the Smpte signal, EyesWeb can control the playback of the recorded data in synch with the playback of the motion tracking data in the Qualisys RTM software. To use this patch, open a QTM file[2] and process it with the command *Run real time processing on file...*, which enables the Qualisys real time protocol (i.e., data streaming via OSC). Then, run the patch; you should see the same Smpte value in both the status bar of the Qualisys software and in the EyesWeb display in the top of the patch (the one with a black background and red digits). The maximum allowed difference between these values can be tuned by means of the slider. However, note that imposing a small threshold may cause the generated smpte signal to be resynched to the value streamed by Qualisys too many times. This can cause degradation of the quality of the played audio signal as the generated Smpte signal contains discontinuities.

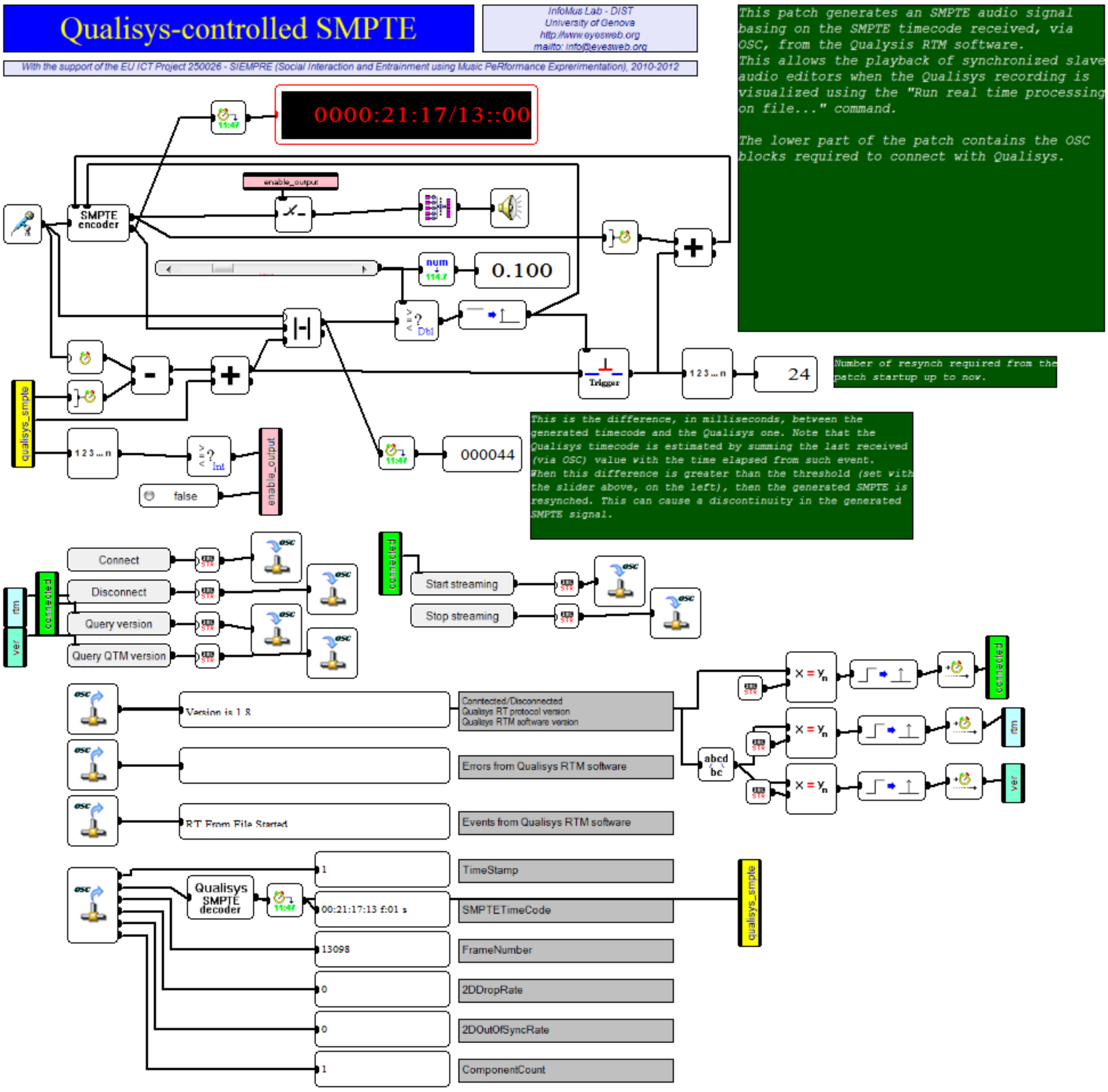
[1] The setup at Casa Paganini uses Adobe Audition as the audio editor

[2] Of course, the file must have been recorded with Smpte enabled



This patch demonstrates how to integrate EyesWeb with the Qualisys Motion Tracking software.
When the patch start it immediately connects to the Qualisys RT server and asks the server to stream data.
The patch extract some info from the header (e.g., the SMPTE timecode) and the coordinates of a marker.

Figure 2.1: A patch to show the integration between EyesWeb and the Qualisys RTM software.



Qualisys-controlled SMPTE

InfoMus Lab - DIST
University of Genova
http://www.eyesweb.org
mailto: info@eyesweb.org

With the support of the EU ICT Project 250026 - SIEMPRE (Social Interaction and Entrainment using Music PeRformance Exprerimentation), 2010-2012

This patch generates an SMPTE audio signal basing on the SMPTE timecode received, via OSC, from the Qualysis RTM software.
This allows the playback of synchronized slave audio editors when the Qualisys recording is visualized using the "Run real time processing on file..." command.

The lower part of the patch contains the OSC blocks required to connect with Qualisys.

Number of resynch required from the patch startup up to now.

This is the difference, in milliseconds, between the generated timecode and the Qualisys one. Note that the Qualisys timecode is estimated by summing the last received (via OSC) value with the time elapsed from such event. When this difference is greater than the threshold (set with the slider above, on the left), then the generated SMPTE is resynched. This can cause a discontinuity in the generated SMPTE signal.

Figure 2.2: A patch that regenerates an audio Smpte timecode based on the Smpte valued received by Qualisys via OSC.

Part II

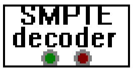
Reference

Chapter 3

SIEMPRE Catalog

3.1 Blocks

3.1.1 SmpteDecoder

bitmap	
class_name	SmpteDecoder
catalog_name	SIEMPRE
catalog_id	SIEMPRE
class_id	smpte_decoder

Decodes an SMPTE timecode from the audio signal.

Details Based on the LTC SMPTE library (http://ltcsmpte.sourceforge.net/), with minor changes to build under Win32. The derived source code is available at https://svn.infomus.org/pub

Notes With the support of the EU ICT Project 250026 - SIEMPRE (Social Interaction and Entrainment using Music PeRformance Experimentation), 2010-2012

Inputs

Audio Stream

id	input
type	Base, PCMAudioBuffer
type_id	base, pcm_audio_buffer
required	required_for_initialization required_for_execution
read_only/read_write	read_only
referred as inplace	*no*
referred as inherited	*no*

The input audio stream which should contain the SMPTE timecode track

Outputs

Timecode

id	output
type	Kernel, Time datatype (Kernel Catalog).
type_id	kernel, time
inplace_id	*no*
inherited_id	*no*

The decoded timecode

Parameters

TimeCode Track

id	time_code_track
type	Kernel, Int datatype (Kernel Catalog).
type_id	kernel, int
domain	[0, +infinity)

The zero-based index of the timecode track. If it is greater than the number of available channels, the last channel will be used.

Enable Output Offset

id	enable_output_offset
type	Kernel, Bool datatype (Kernel Catalog).
type_id	kernel, bool

In general the SMPTE timecode might be not aligned with the audio buffer. If this parameter is set to true, an additional output is generated specifing the sample offset of the SMPTE timecode with respect to the audio buffer. Note that the offset might be negative in the (very common) case when the smpte was partially contained in the previous audo buffer

Enable Output Locked

id	enable_output_locked
type	Kernel, Bool datatype (Kernel Catalog).
type_id	kernel, bool

Enable/disable an output providing info about whether the SMPTE signale has beed locked

TimeCode Framerate

id	frame_rate
type	Kernel, Int datatype (Kernel Catalog).
type_id	kernel, int
layout	Combo Box: Custom ATSC24/film (24fps) NSTC (29.97fps) PAL (25fps) ATSC30 (30fps)
domain	[0, 5)

The framerate of the SMPTE timecode, i.e., how many timecode words are available in a second. If Custom is selected a custom value can be specified in the Custom Framerate parameter, otherwise one standard value can be selected

Custom Framerate

id	custom_frame_rate
type	Kernel, Double datatype (Kernel Catalog).
type_id	kernel, double

The custom framerate of the SMPTE timecode; this parameter is used only when TimeCode Framerate is set to Custom

3.1.2 SmpteEncoder

bitmap	SMPTE encoder
class_name	SmpteEncoder
catalog_name	SIEMPRE
catalog_id	SIEMPRE
class_id	smpte_encoder

Encodes an SMPTE timecode to an audio signal.

Details Based on the LTC SMPTE library (http://ltcsmpte.sourceforge.net/), with minor changes to build under Win32. The derived source code is available at https://svn.infomus.org/pub

Notes With the support of the EU ICT Project 250026 - SIEMPRE (Social Interaction and Entrainment using Music PeRformance Experimentation), 2010-2012

Inputs

Audio clock

id	input
type	Base, Audio clock
type_id	base, audio_clock
required	required_for_initialization required_for_execution
read_only/read_write	read_only
referred as inplace	*no*
referred as inherited	*no*

The input audio clock

Outputs

Output

id	output
type	Base, PCMAudioBuffer
type_id	base, pcm_audio_buffer
inplace_id	*no*
inherited_id	*no*

The encoded timecode

Parameters

TimeCode

id	time_code
type	Kernel, Time datatype (Kernel Catalog).
type_id	kernel, time

The initial timeocde.

Reset

id	reset
type	Kernel, Trigger datatype (Kernel Catalog).
type_id	kernel, trigger

Reset the encoder to the initial timecode value

TimeCode Framerate

id	frame_rate
type	Kernel, Int datatype (Kernel Catalog).
type_id	kernel, int
layout	Combo Box: Custom ATSC24/film (24fps) NSTC (29.97fps) PAL (25fps) ATSC30 (30fps)
domain	[0, 5)

The framerate of the SMPTE timecode, i.e., how many timecode words are available in a second. If Custom is selected a custom value can be specified in the Custom Framerate parameter, otherwise one standard value can be selected

Custom Framerate

id	custom_frame_rate
type	Kernel, Double datatype (Kernel Catalog).
type_id	kernel, double

The custom framerate of the SMPTE timecode; this parameter is used only when TimeCode Framerate is set to Custom

3.1.3 WaveFileWriter

bitmap	wav
class_name	WaveFileWriter
catalog_name	SIEMPRE
catalog_id	SIEMPRE
class_id	wave_file_writer

Write the input time series as a wav file.

Notes With the support of the EU ICT Project 250026 - SIEMPRE (Social Interaction and Entrainment using Music PeRformance Experimentation), 2010-2012

Inputs

TimeSeries

id	input
type	Kernel, Generic datatype
type_id	kernel, generic_datatype
required	required_for_initialization required_for_execution
read_only/read_write	read_write
referred as inplace	*no*
referred as inherited	*no*

Time series to be saved to file

Required interfaces
Kernel, StaticTimeSeries

Parameters

Filename

id	param_filename
type	Kernel, String datatype (Kernel Catalog).
type_id	kernel, string

layout	Filename,
	MustExist=true,
	SaveMode=true,
	OverwritePrompt=true,
	Filter="Wave files (*wav)—*.wav—All files (*.*)—*.*—All files (*.*)—*.*——"

Name of the wav file. Use an empty name to stop the recording

Format

id	param_format
type	Kernel, Int datatype (Kernel Catalog).
type_id	kernel, int
layout	Combo Box: 32bit float (float) 64bit float (Double) Normalized 32bit float (float) Normalized 64bit float (Double)
domain	[0, 4)

Defines the format used to write the sample into the file : 'float' each sample is writed as 32bit floating value; 'double' each sample is writed as 32bit floating value;

Title

id	param_title
type	Kernel, String datatype (Kernel Catalog).
type_id	kernel, string

Title tag. It will be mapped to the 'INAM' Exif tag

Datetime

id	param_datetime
type	Kernel, String datatype (Kernel Catalog).
type_id	kernel, string

Datetime tag. It will be mapped to the 'ICRD' Exif tag

Comment

id	param_comment
type	Kernel, String datatype (Kernel Catalog).
type_id	kernel, string

Comment tag. Free text that will be mapped to the 'ICMT' Exif tag

Timecode

id	param_timecode
type	Kernel, String datatype (Kernel Catalog).
type_id	kernel, string

Timecode tag. It will be mapped to the 'ISMP' Exif tag.

3.1.4 DeckLinkInput

bitmap	DeckLink input
class_name	DeckLinkInput
catalog_name	DeckLink
catalog_id	decklink
class_id	decklink_input
authors	Paolo Coletta

Acquire synchronized audio/video from a DeckLink card.

Outputs

Output

id	video_output
type	Base, Image
type_id	base, image
inplace_id	*no*
inherited_id	*no*

AudioOutput

id	audio_output
type	Base, PCMAudioBuffer
type_id	base, pcm_audio_buffer
inplace_id	*no*
inherited_id	*no*

Parameters

Device Index

id	device_index
type	Kernel, Int datatype (Kernel Catalog).
type_id	kernel, int
domain	[0, +infinity)

The zero-base index of the Decklink card, in the case that more than one is installed in the system

InputMode

id	input_mode
type	Kernel, Int datatype (Kernel Catalog).
type_id	kernel, int
layout	Combo Box: Audio/Video Video only
domain	[0, 2)

Specifies whether to capture audio, or video, or both

Video Mode

id	video_mode
type	Kernel, Int datatype (Kernel Catalog).
type_id	kernel, int
layout	Combo Box: NTSC 720x486 30/1.001fps interlaced NTSC 720x486 30/1.001fps interlaced (3:2 pulldown) PAL 720x576 25fps interlaced NTSC 720x486 30/1.001fps progressive PAL 720x576 25fps progressive HD 1920x1080 24/1.001fps interlaced HD 1920x1080 24fps interlaced HD 1920x1080 25fps progressive HD 1920x1080 30/1.001fps progressive HD 1920x1080 30fps progressive HD 1920x1080 25fps interlaced HD 1920x1080 30/1.001fps interlaced HD 1920x1080 30/1.001fps interlaced HD 1920x1080 50fps progressive HD 1920x1080 60/1.001fps progressive HD 1920x1080 60fps progressive HD 1280x720 50fps progressive HD 1280x720 60/1.001fps progressive HD 1280x720 24/1.001fps progressive 2K 2048x1556 60fps interlaced 2K 2048x1556 60fps interlaced 2K 2048x1556 60fps interlaced
domain	[0, 22)

Select the resolution and framerate of the video stream

Pixel Format

id	pixel_format
type	Kernel, Int datatype (Kernel Catalog).
type_id	kernel, int
layout	Combo Box: UYVY 4:2:2 packed BGRA 4:4:4 raw
domain	[0, 2)

Select the pixel format of the video stream

Num Audio Channels

id	num_channels
type	Kernel, Int datatype (Kernel Catalog).
type_id	kernel, int
domain	[1, +infinity)

Specifies the number of audio channels to capture

Audio Sample Format

id	sample_format
type	Kernel, Int datatype (Kernel Catalog).
type_id	kernel, int
layout	Combo Box: 16bits signed 32bits signed
domain	[0, 2)

Specifies the format of each sample of audio

Part III

Appendices

Appendix A

SIEMPRE Binary Format

A.1 SIEMPRE – Binary file format

Monodimensional timeseries will be saved or converted to time-aligned Broadcast Wave Format compatible files (i.e., extended WAVE files).

The SIEMPRE project will exploit subset of the Broadcast Wave Format standard. In particular it will fix the number of channels to 1 (monodimensional data), the format to WAVE_FORMAT_IEEE_FLOAT (0x03), and the bits per sample to 32.

For a general description of Broadcast Wave Format and Wave files you may refer to the Microsoft documentation or start from the following links:

- `Audio File Format Specifications`
- `SonicSpot`

A.1.1 Format description

The WAVE (sub-)format used for SIEMPRE has the following structure:

Field	Length	Content
ChunkID	4	"RIFF"
ChunkSize	4	Size, in bytes, of the RIFF chunk. Should be equal to the length, in bytes, of the file minus 8.
WaveID	4	"WAVE"
Wave chunks	. . .	Some chunks. Each chunk is identified by a fourCC (four characters identifier) followed by the size, in bytes, of the chunk (the size does not include the FourCC and the size itself). Two mandatory chunks are the "fmt" chunk and the "data" chunk, which contains the characteristics of the file (sample rate, num channels, sample size, etc.) and the sample data respectively.

The fmt chunk and the data chunks have the following structure:

Fmt Chunk		
Field	Length	Content
ChunkID	4	“fmt ”
ChunkSize	4	For the SIEMPRE case the value is 16, a generic reader should be prepared to face different sizes too.
FormatTag	2	For the SIEMPRE case the value is WAVE_FORMAT_IEEE_FLOAT, i.e., 0x03.
NumChannels	2	For the SIEMPRE case the value is fixed to 1, i.e., monodimensional channels
NumSamplesPerSeq	4	Sampling rate (e.g., somthing around 250.0 for kinematical sensors)
AvgBytesPerSec	4	Data rate, in bytes, of the file. Can be computed as NumSamplesPerSeq * BlockAlignment
BlockAlignment	2	Size of a block (a sample for each channel). Can be computed as NumChannels * (BitsPerSample / 8). For the SIEMPRE case NumChannels is fixed to 1, BitsPerSample is fixed to 32, thus BlockAlignement is fixed to 4.
BitsPerSample	2	Number of bits for each sample. For the SIEMPRE case this is fixed to 32 (single precision floating point numbers)

Data Chunk		
Field	Length	Content
ChunkID	4	“data”
ChunkSize	4	For the SIEMPRE case the value is 4 (size of sample) multiplied by the number of samples
Data	. . .	This is the actal data: sequence of float numbers in the SIEMPRE case
Padding bytes	0 or 1	In the SIEMPRE case this is not needed as data is aligned. In the general case this is only needed if data is not aligned to a 16bits boundary

A.1.2 Application compatibility

The files produced by the `Wave File Writer` block have been tested to load in the following applications:

- Players
  - VLC media player
  - Windows Media Player
  - Quick Time player
- Music Software
  - Audacity
  - Sony Sound Force
  - Sony Vegas Pro
- Computing environments
  - Matlab: does not load additional info about the file. This is a general limitation for the WAVE format support in Matlab.