MAX-PLANCK-GESELLSCHAFT

User friendly signal processing web services for annotators in AVATecH and AUVIS

Eric Auer
The Language Archive – Max Planck Institute for Psycholinguistics
Nijmegen, The Netherlands

AVATecH and AUVIS

- Joint Max Planck / Fraunhofer project 2009 – 2012
- Assist with annotation of audio and video recordings
- User friendly ELAN editor integration for annotators
- Basic and versatile building blocks for developers
- Multiple recognizers available, with CMDI metadata
- AUVIS project will add advanced recognizers

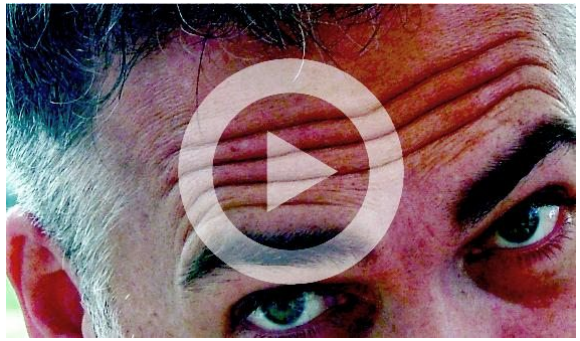
http://www.clarin.eu/cmdi

CLARIN
Common Language Resources and Technology Infrastructure

AVATecH building blocks

- Binary files: Audio, video, other (send, receive)
- Text files: Annotation tier, timeseries (XML, CSV)
- New: XML multitier (independent tiers in one XML)
- Numerical parameter: Min, max
- Text parameter: Choices or free
- All items have name / description

metadata and all parameters have defaults

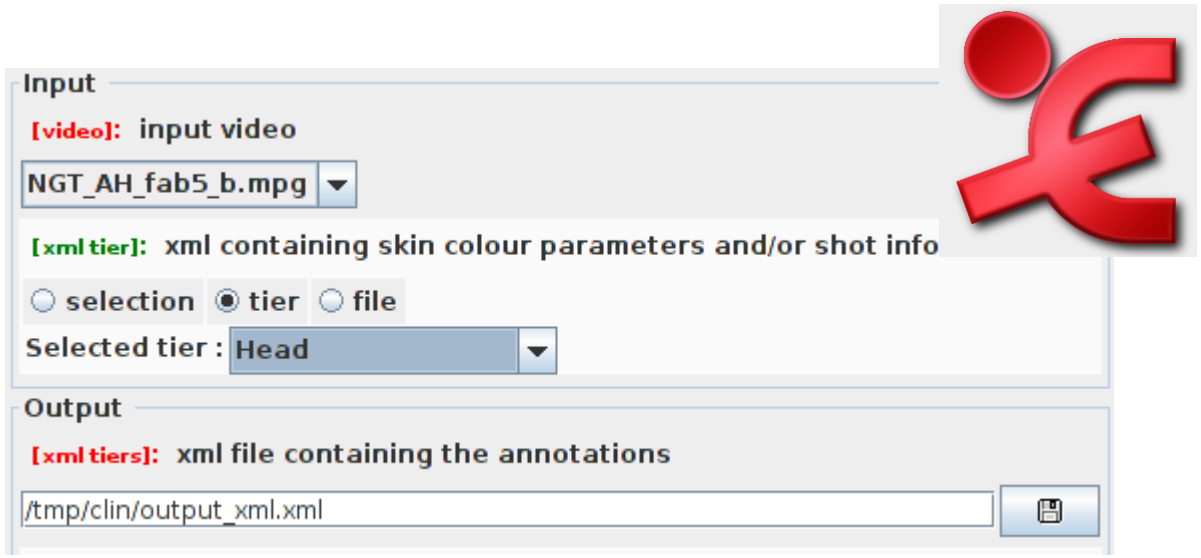


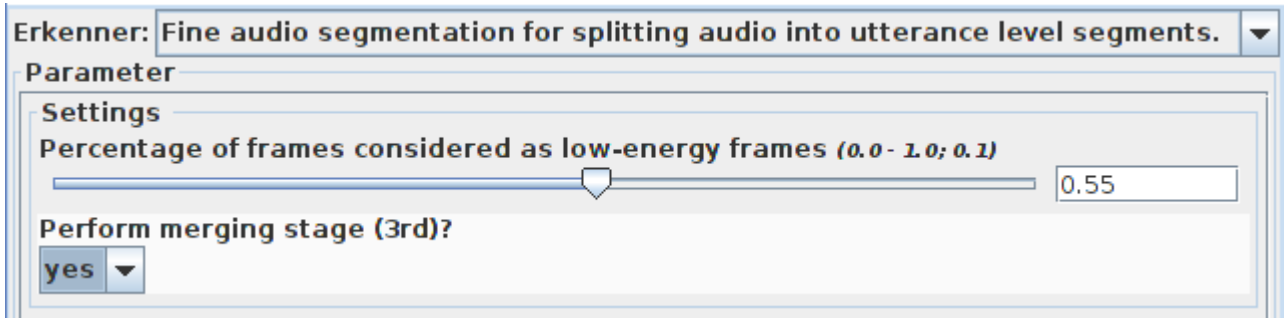
Metadata examples for developers

- <input type="video" optional="false" mimetypes="video/mpeg video/mp4" level="basic" info="video to scan for motion">source_video</input>
- <output type="csvtimeseries" optional="true" mimetypes="text/csv" level="advanced" info="amount of motion over time">motion_curve</output>
- <output type="multitier" optional="false" mimetypes="text/xml" level="basic" info="motion annotation">motion_anno</output>
- <numparam min="23" max="42" default="33" level="basic" info="sensitivity (higher triggers more easily)">motion_threshold</numparam>
- <textparam convoc="yes no automatic" default="automatic" level="advanced" info="ignore movement in the background">background_suppression</textparam>
- <recognizer recognizerType="local" runWin="motionAnnotator.exe /z" runLinux="/opt/avatech/bin/motionAnnotator -z" info="Human motion analysis">motion_recognizer</recognizer>

What the annotator sees in ELAN

- Binary files:
- Text files:
- Multitier:
- Recognizer:
- Number:
- Choice:

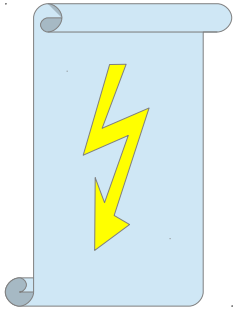




Recognizer invocation under the hood

- <PARAM> <param name="source_video">
  /home/eric/gebarentaal/NGT_AH_fab5_b.mpg</param>
- <param name="skin_segment_info">/tmp/head_21432.xml</param>
- <param name="motion_anno">/tmp/motion_21432.xml</param>
- <param name="motion_threshold">33</param>
- <param name="background_supppression">no</param>
- <param name="InvocationContext">motion_recognizer
  2013-01-15 15:00:01+01:00</param> </PARAM>

Recognizer action



- PARAM XML block is pipelined to recognizer
- Recognizer starts computation, reads user files
- Recognizer creates output files and pipelines logs
- Tagged logs: DEBUG: INFO: WARN: ERROR: RESULT:
- RESULT: DONE. (or FAILED.) – ELAN imports results
- ELAN could show logs (with syntax highlighting)

  during computation. New: PROGRESS: 42% Mogrify

AVATecH webservices with CLAM

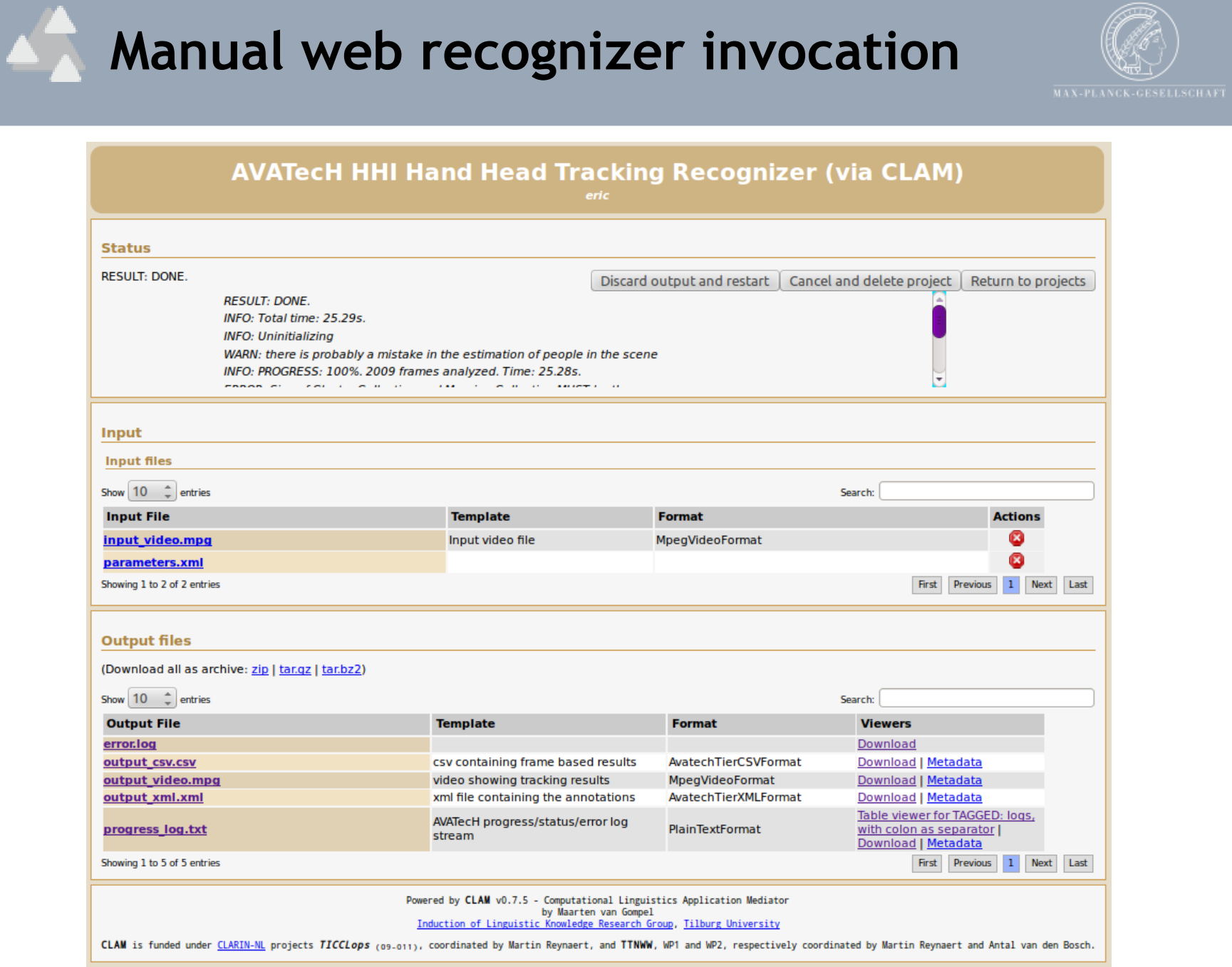
- Free open source (GPL) – http://ilk.uvt.nl/clam/
- Generic command line wrap by Maarten van Gompel
- Straightforward REST webservice interface
- CLAM-specific XML metadata and messages

- Smart CSS / XSLT for manual webapp style invocation
- AVATecH proxy client mimicks recognizers – sole command line args: metadata file, webservice URL

Webservice lifecycle under the hood

- Init - create temporary workspace: PUT http://catalog.clarin.eu/avatech/clin/E487ED5110BBA772 (random name generated by proxy helper)
- Send files: POST http://.../...772/ input/skin_segment_info.xml multipart/form-data, 1. name="inputtemplate" (value: skin_segment_info) 2. name="file"; filename="skin_segment_info.xml" Content-Type: text/xml (value: file content)
- Start computation: POST http://.../...772/ application/x-www-form-urlencoded, e.g. motion_threshold=33 background_suppression=automatic …
- Monitor computation: GET http://.../...772/ returns CLAM XML, e.g. … <status code="..." … /> … (repeat / poll until computation is done or aborts)
- Fetch log: GET http://.../...772/output/progress_log.txt (fixed name here, log can be "tailed" during computation with HTTP HEAD and GET Range requests)
- Fetch output files: GET http://.../...772/output/motion_annotation.xml
- Finally remove workspace: DELETE http://.../...772/ (this also aborts computations in case they were still running)

Manual web recognizer invocation



AVATecH HHI Hand Head Tracking Recognizer (via CLAM)

eric

Status

RESULT: DONE.

Discard output and restart | Cancel and delete project | Return to projects

RESULT: DONE.
INFO: Total time: 25.29s.
INFO: Uninitializing
WARN: there is probably a mistake in the estimation of people in the scene
INFO: PROGRESS: 100%. 2009 frames analyzed. Time: 25.28s.

Input

Input files

Show 10 entries Search:

Input File	Template	Format	Actions
input_video.mpg	Input video file	MpegVideoFormat	
parameters.xml			

Showing 1 to 2 of 2 entries First | Previous | 1 | Next | Last

Output files

(Download all as archive: zip | tar.gz | tar.bz2)

Show 10 entries Search:

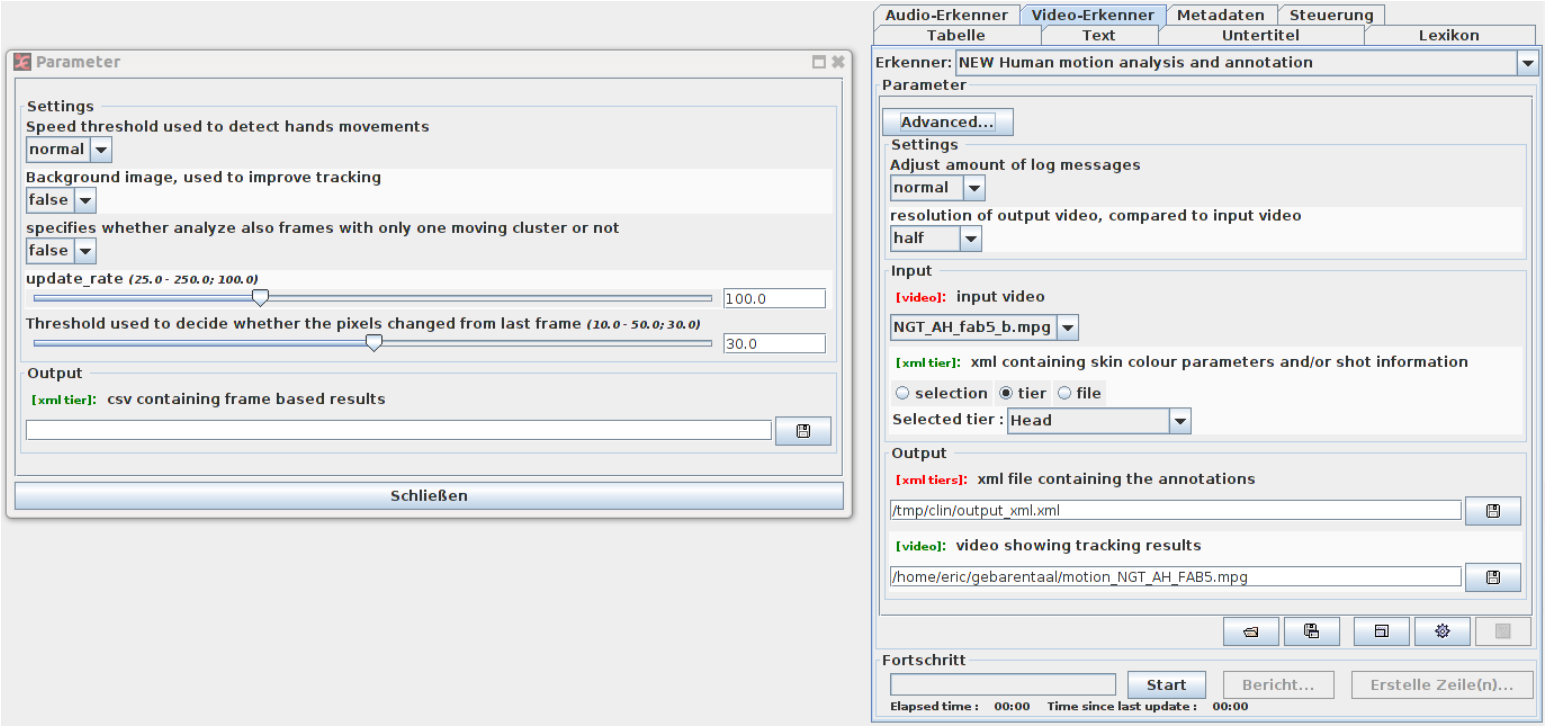
Output File	Template	Format	Viewers
error.log			Download
output_csv.csv	csv containing frame based results	AvatechTierCSVFormat	Download | Metadata
output_video.mpg	video showing tracking results	MpegVideoFormat	Download | Metadata
output_xml.xml	xml file containing the annotations	AvatechTierXMLFormat	Download | Metadata
progress_log.txt	AVATecH progress/status/error log stream	PlainTextFormat	Table viewer for TAGGED: logs, with colon as separator | Download | Metadata

Showing 1 to 5 of 5 entries First | Previous | 1 | Next | Last

Powered by **CLAM** v0.7.5 - Computational Linguistics Application Mediator
by Maarten van Gompel
Induction of Linguistic Knowledge Research Group, Tilburg University

CLAM is funded under CLARIN-NL projects ***TICCLops*** (09-011), coordinated by Martin Reynaert, and **TTNWW**, WP1 and WP2, respectively coordinated by Martin Reynaert and Antal van den Bosch.

ELAN 4.5 web recognizer invocation



- "Same as local recognizer" but proxy has to transfer files – still server can be faster or have more tools
- Advanced settings in (normally closed) pop-up window

Recognizers accessible via CLAM

- Fine and Standard Audio Segmentation
- Model Based Speech Segmentation
- Relative Silence Recognizer
- Speaker Diarization (who speaks when)

Fraunhofer IAIS

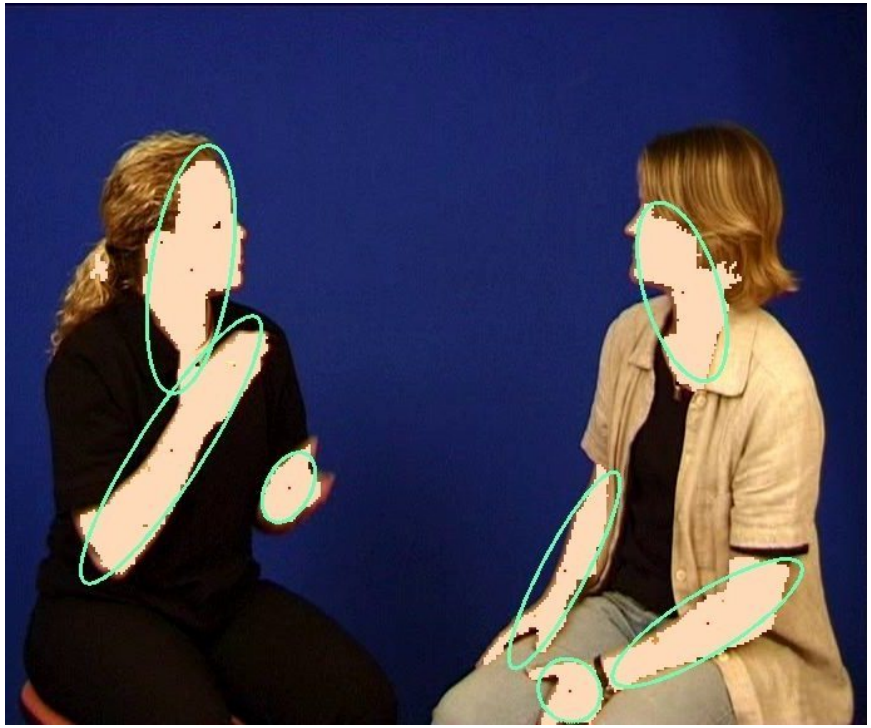
- Sphinx Model Based Speech Alignment
- Praat Based Tag Vowel Segmentation

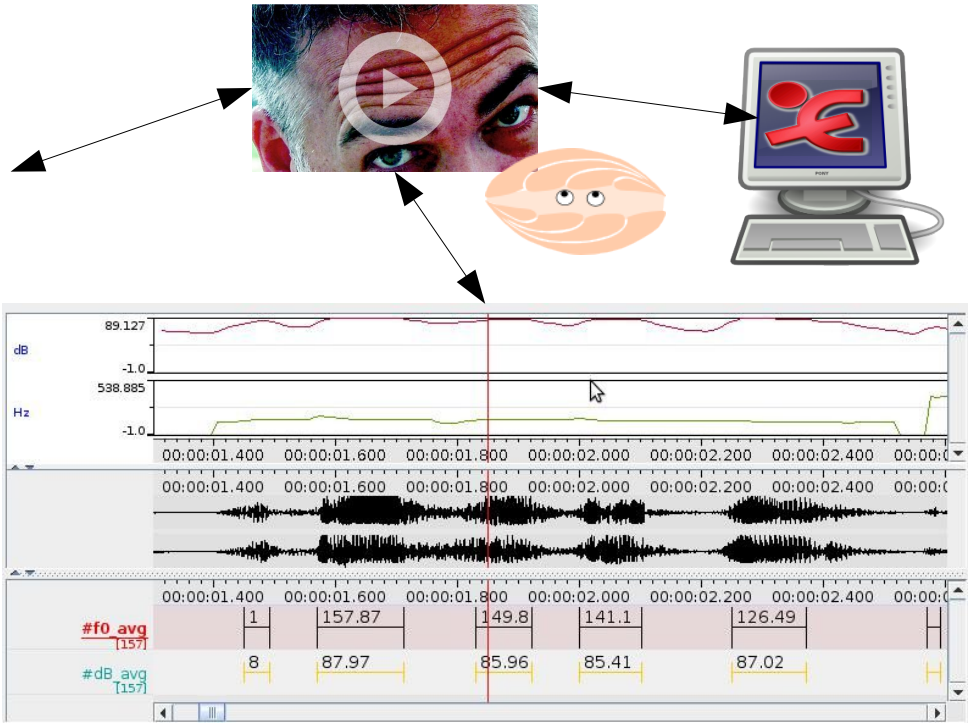
(demos)

- Video Hand Head Tracking
- Video Key Frame Extraction
- Video Skin Tone Estimator

Fraunhofer HHI

Recognizer output examples



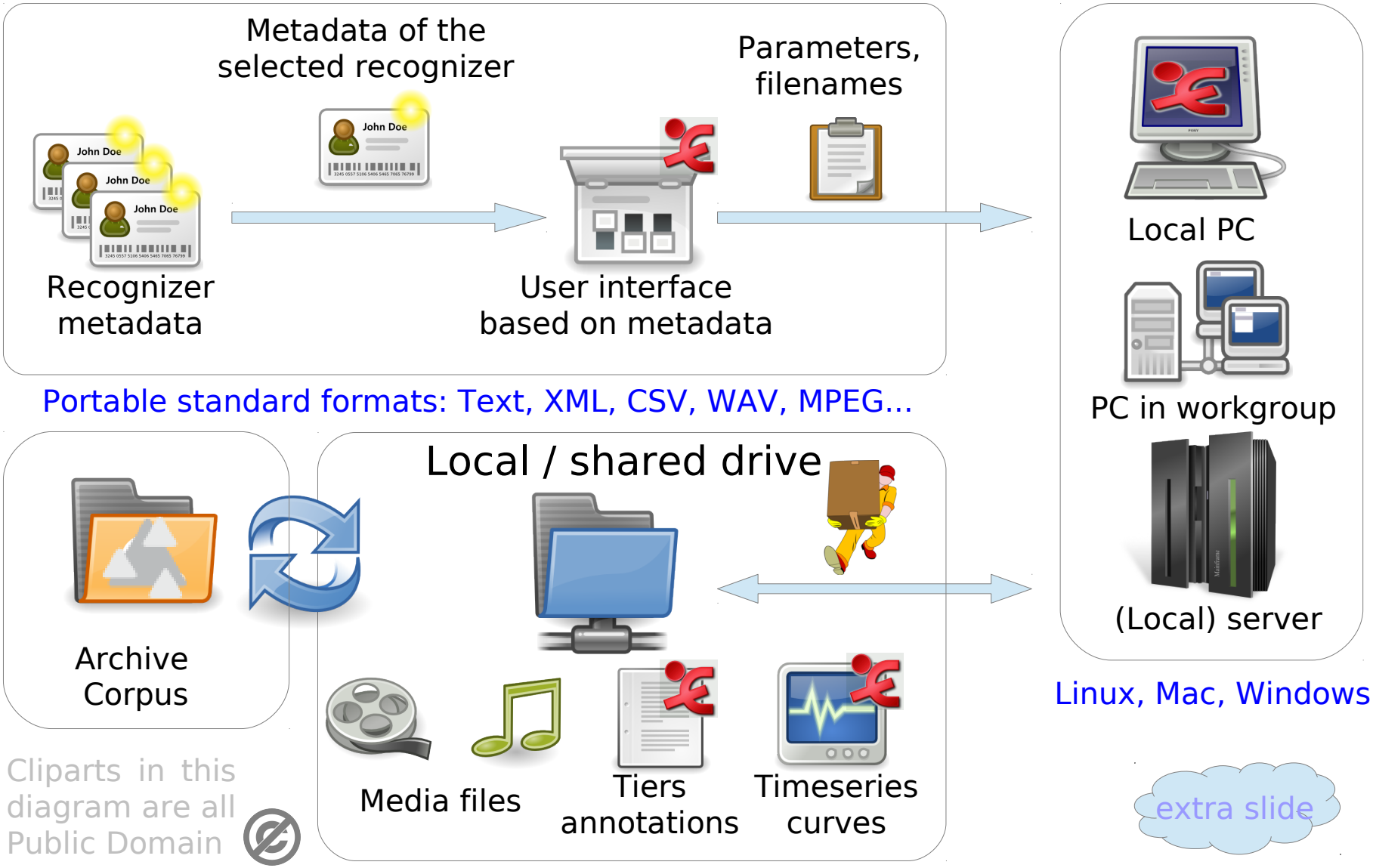


- Video can be added to media list or reviewed manually
- Timeseries to view in ELAN or use in math software
- ELAN helps annotators to send and receive tiers

Questions! And references.

- AVATecH: http://tla.mpi.nl/projects_info/avatech/
- AUVIS: http://tla.mpi.nl/projects_info/auvis/
- ELAN: http://tla.mpi.nl/tools/tla-tools/elan/
- CLAM: http://proycon.github.com/clam/

Answers? The AVATecH framework.



Apache and WSGI config snippets

- WSGIScriptAlias /model_speech_alignment \
  "/opt/clam/config/model_speech_alignment.wsgi/"
  WSGIDaemonProcess model_speech_alignment \
  user=avatech group=users home=/opt/clam threads=5 \
  maximum-requests=42
  WSGIProcessGroup model_speech_alignment

- #!/usr/bin/env python
  import os
  import sys
  CLAMPARENTDIR = '/opt' # directory with 'clam' subdirectory
  sys.path.append(CLAMPARENTDIR)
  os.environ['PYTHONPATH'] = CLAMPARENTDIR
  import clam.config.model_speech_alignment
  import clam.clamservice
  application = clam.clamservice.run_wsgi(
  clam.config.model_speech_alignment)

extra slide

Wrapper shell script excerpt

```
STATUSFILE=$1; INDIR=$2 ; OUTDIR=$3 ; MODEL=$4
ln -s $STATUSFILE $OUTDIR/progress_log.txt
# … create $INDIR/parameters.xml to pass the values:
# audio = $INDIR/audio.wav model = $MODEL
# text = $INDIR/text.wav output = $OUTDIR/output.csv …
if /opt/recognizers/msa/ModelSpeechAligner.sh \
  < $INDIR/parameters.txt >> $STATUSFILE
then # make sure that progress log contains RESULT: DONE.
else # make sure that progress log contains RESULT: FAILED.
fi
grep '^RESULT: DONE.' < $STATUSFILE > /dev/null
exit # return status of the grep: success iff result == done
```

extra slide

CLAM config and metadata excerpt 1

```
SYSTEM_ID = "mpimodelspeechalignment"
SYSTEM_NAME = "Sphinx-based Speech Alignment"
SYSTEM_DESCRIPTION = "CLAM-wrapped Sphinx..."
ROOT = "/tmp/model_speech_alignment/" # for temp workspaces
CLAMDIR = "/opt/clam/" # … where CLAM is installed
HOST = "catalog.clarin.eu"
URL = "http://catalog.clarin.eu/"
URLPREFIX = "avatech/model_speech_alignment"
COMMAND = CLAMDIR + "wrappers/model-speech-alignment.sh
  $STATUSFILE $INPUTDIRECTORY $OUTPUTDIRECTORY $PARAMETERS"
PARAMETERS =  [
    ('Main', [ # Boolean Choice Integer Float String Text Static …
        ChoiceParameter(id='model', name='Language Model',
            description='Which CMU Sphinx language model?',
            choices=['English','German','Dutch'],
            default='English', required=True, paramflag='' ),
    ] )
]
```

extra slide

CLAM config and metadata excerpt 2

```
PROFILES = [
    Profile(
        InputTemplate('audio', WaveAudioFormat,"Input audio file",
            filename='audio.wav', # rename on upload, fixed name
            extension='.wav',
        ),
        InputTemplate('text', PlainTextFormat, "Input text to align",
            StaticParameter(id='encoding',name='Encoding',
                description='Character encoding', value='utf-8'),
            filename='text.txt', # rename on upload, fixed name
            extension='.txt',
        ),
```

extra slide

CLAM config and metadata excerpt 3

```
        OutputTemplate('csvtier', AvatechTierCSVFormat, 'Aligned text',
            SetMetaField('encoding','utf-8'),
            SemicolonTableViewer(), # or SimpleTableViewer(),
            filename='csvtier.csv', # default was input+extension
        ),
        OutputTemplate('progress_log', PlainTextFormat, 'Log stream',
            SetMetaField('encoding','utf-8'),
            AvatechLogViewer(), # or SimpleTableViewer(),
            filename='progress_log.txt', # default was input+extension
        ),
    )  # end of Profile( ...
] # end of PROFILES = [ ...
```

extra slide

Custom CLAM file format definitions

- Excerpts from CLAM common/formats.py additions:

```
class AvatechTierCSVFormat(CLAMMetaData):
    """AVATecH Tier CSV format, no attributes, always UTF-8.
    Use headers: "#starttime";"#endtime";"YOURCOLUMN"
    Times in seconds. Example row: 0.00;1.23;"Hello world" (more columns ok)"""

    attributes = { 'encoding':'utf-8' }
    mimetype = "text/csv"
    # validate(self) always returns True for now
    # httpheaders(self) always returns a fixed value:
    # … yield ("Content-Type", self.mimetype + "; charset=" + self['encoding'])
```

```
class AvatechTierXMLFormat(CLAMMetaData):
    """AVATecH Tier XML format, UTF-8, see specs at www.mpi.nl/avatech"""

    attributes = { 'encoding':'utf-8' }
    mimetype = "text/xml"
    scheme = " # scheme known, but CLAM does not yet validate schemes
    # httpheaders following the same pattern as AvatechTierCSVFormat
```

extra slide

Custom CLAM viewer definitions

- Simplistic extension for CLAM common/viewers.py:

- class AvatechLogViewer(AbstractViewer):

```
    id = 'avatechlogtableviewer' # View logs as HTML
    name = "Table viewer for TAGGED: logs"

    # would be better to process only the first ":"
    # could add syntax highlighting / per-tag styles
    def view(self,file,**kwargs):
        render = web.template.render('templates')
        return render.crudetableviewer( file, ":")
```

last slide!