Record, Review, Share and Analyze Behavioral Data from 6DoF-XR Experiments with PLUME

Charles JAVERLIAT*, Sophie VILLENAVE*, Pierre RAIMBAUD, Guillaume LAVOUÉ

École Centrale de Lyon, CNRS, LIRIS UMR5025, ENISE

Motivations

Understand user experience using data

- Self-reported data: questionnaires, semi-structured interviews
- Behavioral data: inputs, movements, interactions
- Physiological data: EEG, ECG, EDA, EOG, HR, etc.

Motivations

Self-reported data has intrinsics limitations

- Question framing bias
- Response subjectivity bias
- Memory bias

Motivations

Behavioral and physiological data acquisition is challenging

- Tedious to acquire
- Hard to synchronize
- Performance-intensive
- Heterogeneous format among studies
- Replicability and reproducibility issues due to partial recording.

Motivations

Existing tools* fall short of requirements

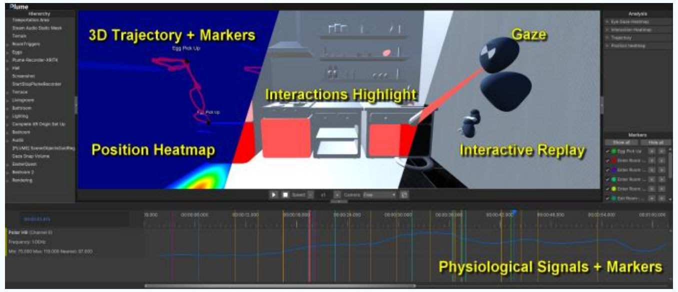
- Time consuming installation
- Tightly couples the tool with the project
- Limited compatibility with physiological hardware
- Limited compatibility with head mounted displays
- Performance-intensive
- Proprietary license

* VR Scientific Toolkit (VRSTK), Cognitive3D, Tobii Ocumen, Vizard, MIRIA, NVIDIA VCR, …
(Full comparative table available in the paper)

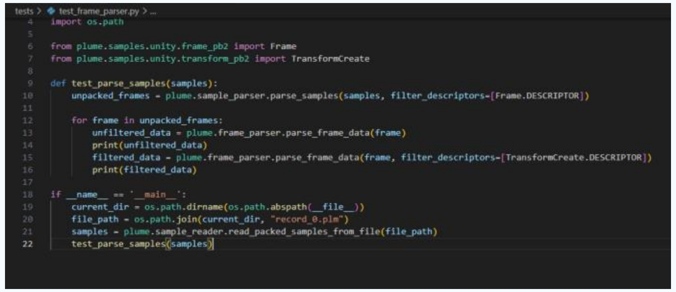
Plume

Plume Recorder

Plume Viewer



Plume Python



```python
import os.path

from plume.samples.unity.frame_pb2 import Frame
from plume.samples.unity.transform_pb2 import TransformCreate

def test_parse_samples(samples):
    unpacked_frames = plume.sample_parser.parse_samples(samples, filter_descriptors=[Frame.DESCRIPTOR])

    for frame in unpacked_frames:
        unfiltered_data = plume.frame_parser.parse_frame_data(frame)
        print(unfiltered_data)
        filtered_data = plume.frame_parser.parse_frame_data(frame, filter_descriptors=[TransformCreate.DESCRIPTOR])
        print(filtered_data)

if __name__ == '__main__':
    current_dir = os.path.dirname(os.path.abspath(__file__))
    file_path = os.path.join(current_dir, "record_0.plm")
    samples = plume.sample_reader.read_packed_samples_from_file(file_path)
    test_parse_samples(samples)
```

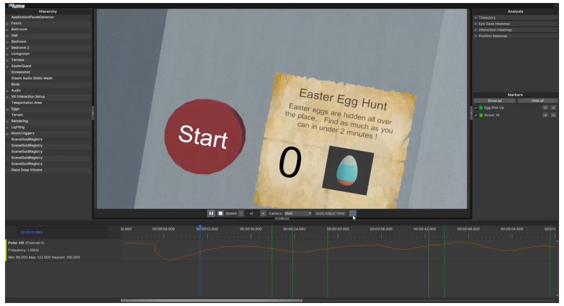
PLUME Overview

Plume Recorder

Plugin installed in application

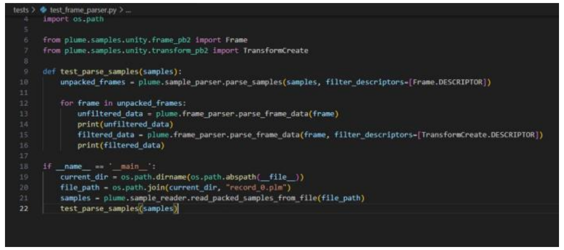
PLM file
Self-contained
record file

Plume Viewer



In-situ replay and analysis

Plume Python



```
tests > test_frame_parser.py > ...
import os.path

from plume.samples.unity.frame_pb2 import Frame
from plume.samples.unity.transform_pb2 import TransformCreate

def test_parse_samples(samples):
    unpacked_frames = plume.sample_parser.parse_samples(samples, filter_descriptors=[Frame.DESCRIPTOR])

    for frame in unpacked_frames:
        unfiltered_data = plume.frame_parser.parse_frame_data(frame)
        print(unfiltered_data)
        filtered_data = plume.frame_parser.parse_frame_data(frame, filter_descriptors=[TransformCreate.DESCRIPTOR])
        print(filtered_data)

if __name__ == '__main__':
    current_dir = os.path.dirname(os.path.abspath(__file__))
    file_path = os.path.join(current_dir, "record_0.plm")
    samples = plume.sample_reader.read_packed_samples_from_file(file_path)
    test_parse_samples(samples)
```

Ex-situ analysis

PLUME File Format (.plm)

Self-contained

No dependency on project files, easy to share.

Raw data

Low-level, timestamped, data (transforms, components properties, physiological signals) for maximum reproducibility and data repurposing.

Easy parsing

Platform-neutral and language-neutral format using Protobuf. Easy data wrangling thanks to PLUME Python.

Plume

Recorder

Plume Recorder

Plug-and-play Unity Plugin

1min setup.

Record as much as possible by default

Positions, meshes, spatialized audio, eye-gaze, inputs, event markers, physiological signals, …

Fast and lightweight

Using IL weaving for hooks injection in assemblies and Burst-compiled routines.

No strings attached

No changes required in the project scripts. Uninstalling is as easy as installing.

Plume Recorder

Cross-platforms (PC-VR and autonomous)
Windows, iOS, Android, PC-VR.

OpenXR compatibility for XR specific data
Eye gaze, controlers inputs, interactions, etc.

Individual or collaborative scenarios

Large compatibility with physiological sensors
Compatibility layer with the LabStreamingLayer.

Plume
Viewer

Plume Viewer

Standalone desktop 3D Viewer

Doesn't require the Unity project to replay .plm files.

Post-experiment visual inspection

Detect issues, post-experiment debrief with participants.

In-situ analysis of synchronized multi-modal data

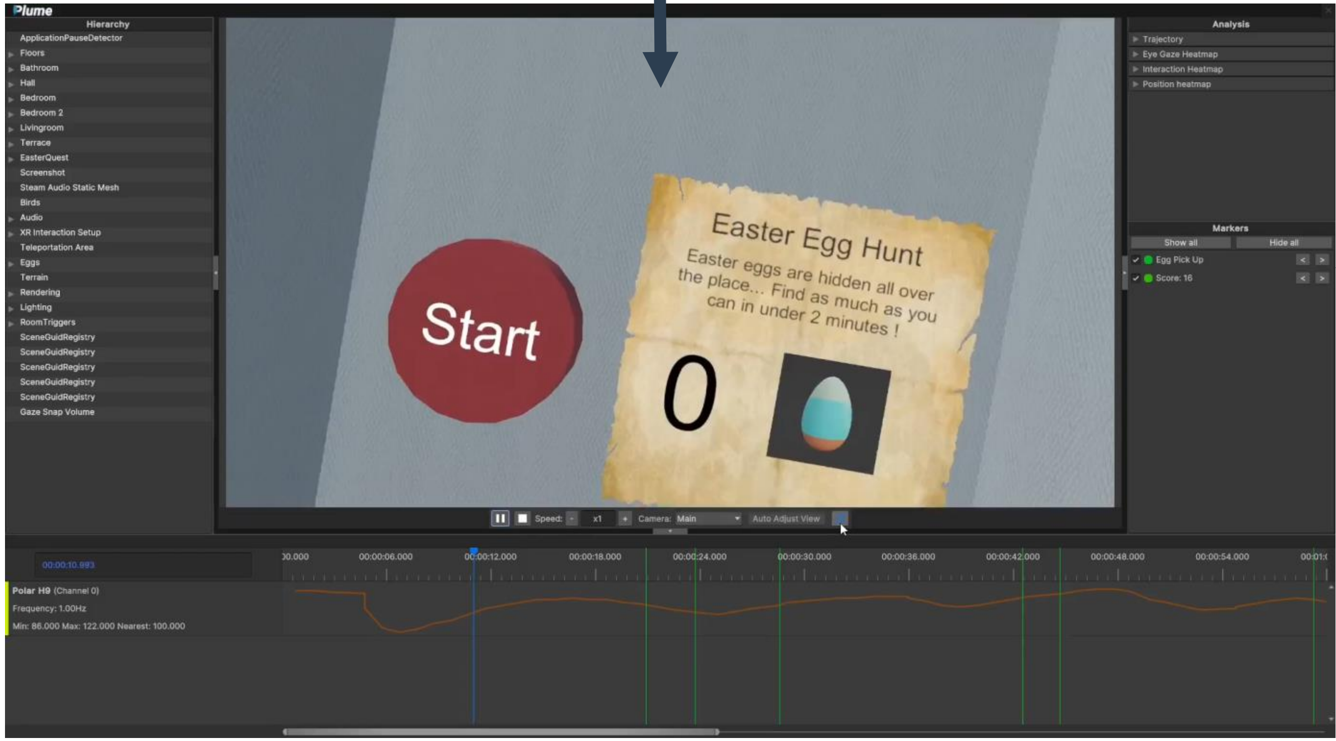
Run custom analysis modules directly on 3D data, quickly find correlations between scene state and physiological state.

Plume Viewer

PLM file

Self-contained record file

3D interactive player



Easter eggs are hidden all over the place... Find as much as you can in under 2 minutes !

☑ ● Egg Pick Up

☑ ● Score: 16

Speed: x1

Camera: Main

Polar H9 (Channel 0)

Frequency: 1.00Hz

Min: 86.000 Max: 122.000 Nearest: 100.000

Analysis modules

Event Markers

Timeline with physiological tracks

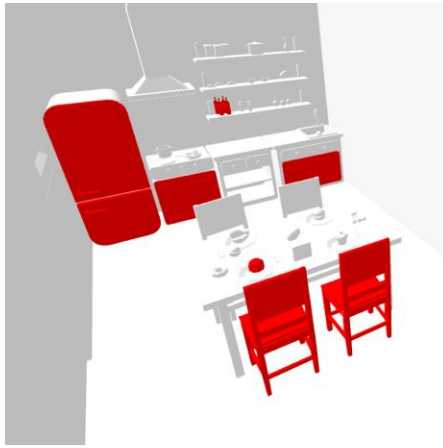
Plume Viewer

Builtin in-situ analysis modules

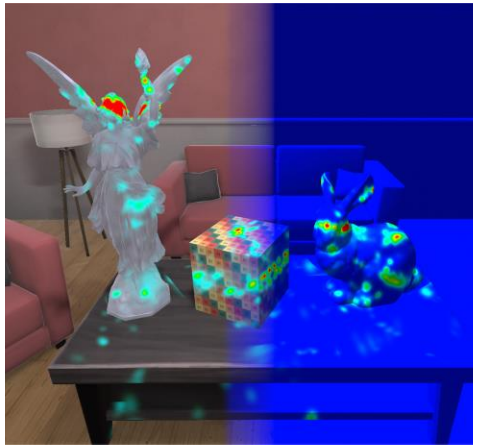
Quickly create trajectories, interaction highlights, position heatmaps for any object, eye gaze heatmaps, etc.

Egg Pick Up
Egg Pick Up
Egg Pick Up

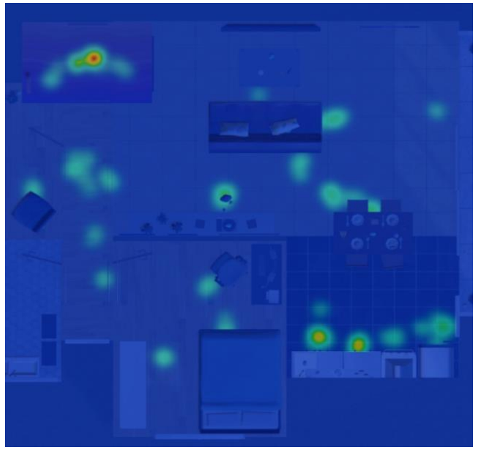
Trajectories (with markers)



Interaction highlights



Eye gaze heatmaps

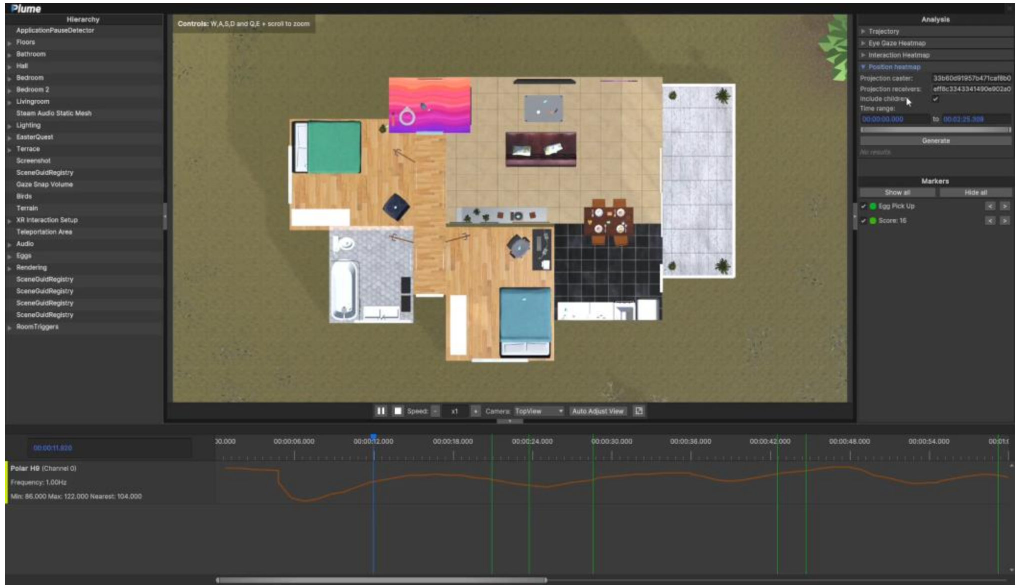


Position heatmaps

Plume Viewer

Create your own modules

Use the raw data and the 3D context to build insightful spatiotemporal visualizations.



Projection caster: 33b60d91957b471caf8b0

Projection receivers: eff8c334334149 0e902a0

Include children: ✓

Time range: 00:00:00.000 to 00:02:25.388

Speed: x1

Camera: TopView

00:00:11.920

Polar H9 (Channel 0)

Frequency: 1.00Hz

Min: 86.000 Max: 122.000 Nearest: 104.000

Plume

Python

Plume
Python

Quick export to other formats

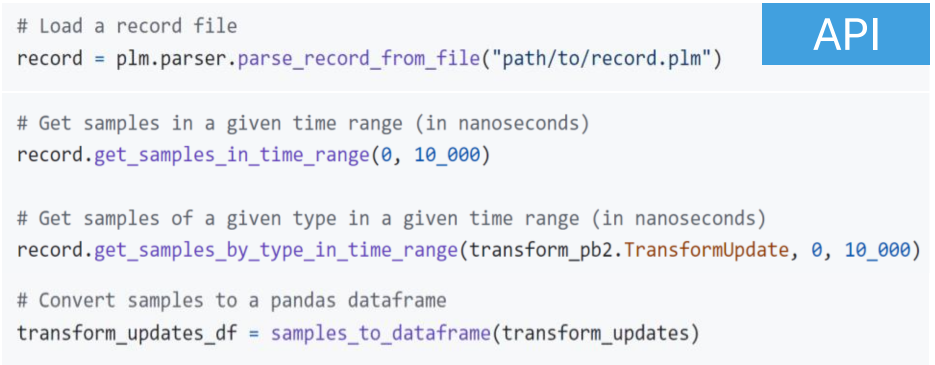
Export your data as CSV and XDF using the Python CLI

Easy data wrangling

Quickly load the .plm file as a dataframe for direct use in traditional analytical pipelines like pandas or R.

Plume
Python

PLM file
Self-contained
record file

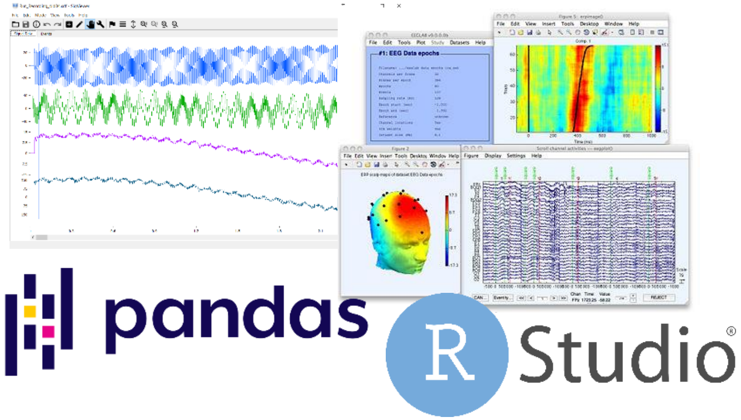


```
# Load a record file
record = plm.parser.parse_record_from_file("path/to/record.plm")

# Get samples in a given time range (in nanoseconds)
record.get_samples_in_time_range(0, 10_000)

# Get samples of a given type in a given time range (in nanoseconds)
record.get_samples_by_type_in_time_range(transform_pb2.TransformUpdate, 0, 10_000)

# Convert samples to a pandas dataframe
transform_updates_df = samples_to_dataframe(transform_updates)
```



Integration in well-established ecosystems
(pandas, R, EEGLAB, SigViewer)

Plume

PLUME is free and open-source!

GitHub

https://github.com/liris-xr/PLUME

Discord

https://discord.gg/QMnKCvhBCf

Case Study

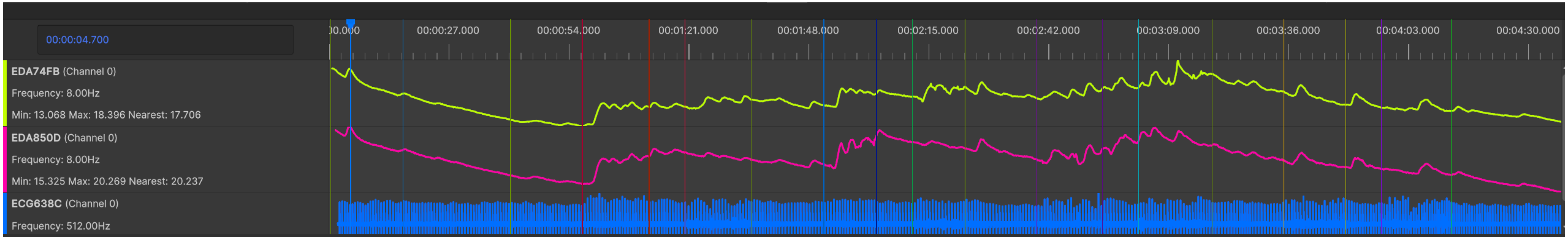
PLUME to Evaluate Physiological Signal Quality

(Part of the *Renforce* ANR Project)

Evaluating bio-signals sensors for VR

<u>Goal:</u> Measuring the impact of typical VR movements on signal quality depending on electrode placement

- **Recorder** - Synchronization of VR tasks and physiological signal using markers (at beginning and end of tasks)
- **Viewer -** Inspect records to understand discrepancies in the data and empirically correlate movements and noise
- **Python -** Compute signals quality using Pearson Correlation for EDA and Template Matching for ECG



00:00:04.700

EDA74FB (Channel 0)
Frequency: 8.00Hz
Min: 13.068 Max: 18.396 Nearest: 17.706

EDA850D (Channel 0)
Frequency: 8.00Hz
Min: 15.325 Max: 20.269 Nearest: 20.237

ECG638C (Channel 0)
Frequency: 512.00Hz

Plume

- How to ? Install Plume Recorder in a Unity Project
- Hands-on Plume Python to compute ex-situ analysis
- Hands-on Plume Viewer to produce in-situ visualizations

LIRIS
CENTRALE LYON ENISE
cnrs
La Région Auvergne-Rhône-Alpes
anr agence nationale de la recherche

Plume

https://liris-xr.github.io/PLUME/ > Learn > Learn the basics

https://github.com/liris-xr/PLUME-Tutorial-Basics/releases/ > Download record1.plm, record2.plm and plume_bundle.zip

LIRIS CENTRALE LYON ENISE cnrs La Région Auvergne-Rhône-Alpes anr agence nationale de la recherche