tsseg: An Interactive Toolkit for Time Series Segmentation

Félix Chavelli[1]✉, Arik Ermshaus[2], Fan Yang[3], Patrick Schäfer[2], John Paparrizos[3,4], and Paul Boniol[1]

[1] Inria, ENS, CNRS, PSL
felix.chavelli@inria.fr
[2] Humboldt-Universität zu Berlin
[3] The Ohio State University
[4] Aristotle University of Thessaloniki

Abstract. Time series segmentation, encompassing both Change Point Detection (CPD) and State Detection (SD), is central to numerous applications, from activity recognition to industrial monitoring. Despite growing research interest, practitioners face a fragmented landscape: algorithms are scattered across incompatible codebases, evaluation protocols differ, and no tool enables fast comparison under a common API. In this paper, we present **tsseg**, an open-source Python library unifying 26 segmentation algorithms from four algorithmic families with 10 dedicated evaluation metrics, together with a fully interactive Gradio-based web application deployed on HuggingFace Spaces. The demo enables users to load or generate time series, run and compare multiple algorithms side by side, and evaluate results with standardized CPD and SD metrics, all without writing any code. **A video demonstrating our system is available on our web application.**

Keywords: Time series segmentation · Change point detection · State Detection · Interactive demo · Benchmarking · Open-source

1 Introduction

Time series segmentation is the task of partitioning a temporal sequence into contiguous, homogeneous segments. Two complementary formulations coexist: *Change Point Detection* (CPD), which identifies time indices where statistical properties shift, and *State Detection* (SD), which assigns a discrete label to each time step. Applications range from activity recognition [2] to healthcare monitoring [5] and industrial fault detection [6].

Despite its importance, the field remains fragmented: algorithms use one-off implementations, evaluation protocols vary, and no unified toolkit enables side-by-side comparison. `ruptures` [10] focuses on a narrow family of CPD methods, while `aeon` [8] and `sktime` [7] only provide a limited set of algorithms, with uneven representation across different families. We address this gap with:

1. a Python library providing 26 segmentation algorithms under a consistent `fit`/`predict` API with 10 evaluation metrics covering both CPD and SD.

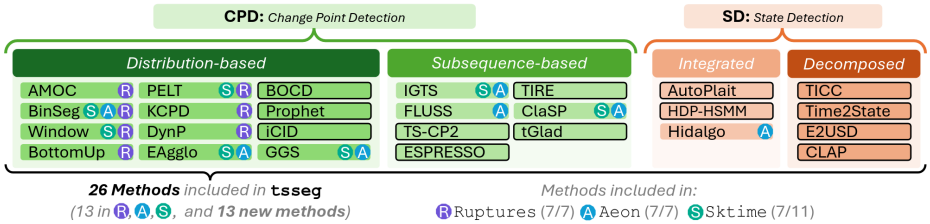

Fig. 1: Algorithms in `tsseg`, grouped by task (CPD/SD) and family. Superscripts mark methods also available in `ruptures` (R), `aeon` (A), or `sktime` (S).

2. an interactive Gradio [1] web application deployed on HuggingFace Spaces[5] for code-free exploration and benchmarking.

The library and application are released under AGPLv3 at https://github.com/fchavelli/tsseg and distributed through pip; documentation is hosted by Read the Docs at https://fchavelli.github.io/tsseg/.

2 `tsseg`: Architecture and Interface

Unlike existing time-series segmentation libraries, our system follows a two-layer architecture: the **`tsseg` library** provides algorithms, metrics, and datasets, while the associated **application** exposes them via an interactive web interface.

Library layer. The core library of our system `tsseg` provides three modules:

- **Algorithms**: 26 offline multivariate segmentation algorithms from four families (Fig. 1) are included. Thirteen are thin wrappers around existing Python libraries, while the remaining 13 (including 6 for SD) are integrations ported from C, R or MATLAB, or repackaged from reference implementations. All algorithms expose a unified `fit`/`predict` API (`aeon`/`sklearn`-style) with task-specific standardised outputs: sparse change-point indices for CPD and dense per-step state vectors for SD. Most detectors perform their own per-channel z-normalisation, while additional preprocessing is left to the user. Several methods originally restricted to either the univariate or multivariate setting have also been extended to support the converse case (see documentation). Optional GPU acceleration is available for compatible methods.
- **Metrics**: 10 evaluation metrics covering both paradigms: standard and Gaussian F1 Score, standard and bidirectional Covering Score for CPD; Adjusted Rand Index (ARI), Normalised and Adjusted Mutual Information (NMI and AMI), Weighted ARI and NMI [4], and State Matching Score [4] for SD.
- **Datasets**: built-in loader for the CMU MoCap dataset [3] and loaders for 11 labeled open source datasets commonly used in the literature.

[5] https://huggingface.co/spaces/fchavelli/tsseg

Application layer. Our web interface allows users to explore the library's capabilities. It is organised into four main frames:

- **Data**: Handles custom synthetic data generation, built-in datasets loading, or custom CSV uploads (with or without ground truth labels).
- **Segmentation**: Select algorithms and their hyperparameters with automatic parameter validation, and optionally incorporate prior knowledge (e.g., the true number of segments); each method also reports its theoretical complexity.
- **Comparison**: Visually align and display a selection of independent runs (i.e., same or different methods with different hyperparameters) side by side, acting as an empirical visual benchmark.
- **Evaluation**: Compute accuracy metrics for the selected runs, outputting bar charts and downloadable CSV logs.

Together, these tabs enable quick no-code benchmarking of multiple methods on custom datasets using diverse metrics in just a few clicks. The application is deployed on HuggingFace Spaces with 18 GB RAM, 2 CPU cores, and 50 GB of non-persistent disk space as an exploratory front-end; for large-scale runs, we recommend the pip-installable library on dedicated hardware.

3 tsseg in Practice

We illustrate the library's unified API through a simple code example provided below. The user loads a dataset, applies a segmentation algorithm, and evaluates the results using custom metrics.

```
from tsseg.data.datasets import load_mocap
from tsseg.algorithms import ClapDetector
from tsseg.metrics import StateMatchingScore

# Load a time series (from the MoCap dataset with ground truth labels)
X, y_true = load_mocap()
# Segment using CLaP algorithm, predicting state labels
segmenter = ClapDetector()
segmenter.fit(X)
y_pred = segmenter.predict(X)
# Evaluate using State Matching Score metric
score = StateMatchingScore().compute(y_true, y_pred)
```

This uniform interface, allows easy benchmarking: any of the detectors or metrics can be substituted in the snippet above, with no further code change.

4 tsseg and Related Systems

Table 1 summarises how `tsseg` compares to existing tools. `ruptures` [10] provides six CPD methods but no SD support; `aeon` [8] and `sktime` [7] are generalist time-series toolkits with 8 and 11 segmenters respectively, while `tslearn` [9] focuses on classification/clustering. To our knowledge, `tsseg` is the first library to unify CPD and SD under a single API with dedicated metrics and a web app.

Table 1: Comparison of `tsseg` with existing time series libraries.

Criterion	`tsseg`	`sktime` [7]	`aeon` [8]	`ruptures` [10]
Segmentation algorithms	**26**	11	8	6
Evaluation metrics	**10**	8	3	3
Change Point Detection (CPD)	19	6	6	6
State Detection (SD)	7	5	2	✗
Interactive application	✓	✗	✗	✗

5 Conclusion

In this paper, we present our system `tsseg`, a unified library for time series segmentation with an interactive Gradio application for code-free exploration. Both are publicly available on GitHub and HuggingFace Spaces. Future work includes expanding algorithm coverage (e.g. online/streaming methods), datasets, and integrating foundation-model-based segmenters.

Acknowledgements

We thank Michaël Thomazo for his valuable insights throughout this work.

References

1. Abid, A., Abdalla, A., Abid, A., Khan, D., Alfozan, A., Zou, J.Y.: Gradio: Hassle-free sharing and testing of ML models in the wild. CoRR **abs/1906.02569** (2019)
2. Bulling, A., Blanke, U., Schiele, B.: A tutorial on human activity recognition using body-worn inertial sensors. ACM Computing Surveys **46**(3), 1–33 (2014)
3. Carnegie Mellon University: The cmu graphics lab motion capture database. In: CMU Graphics Lab (2003)
4. Chavelli, F., Boniol, P., Thomazo, M.: Toward interpretable evaluation measures for time series segmentation. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems (2025)
5. Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G.: The ucr time series classification archive (July 2015)
6. Katser, I.D., Kozitsin, V.O.: Skoltech anomaly benchmark (skab) (2020)
7. Löning, M., Bagnall, A.J., Ganesh, S., Kazakov, V., Lines, J., Király, F.J.: sktime: A unified interface for machine learning with time series. CoRR (2019)
8. Middlehurst, M., Ismail-Fawaz, A., Guillaume, A., Holder, C., Guijo-Rubio, D., Bulatova, G., Tsaprounis, L., Mentel, L., Walter, M., Schäfer, P., Bagnall, A.: aeon: a python toolkit for learning from time series. JMLR **25**(289), 1–10 (2024)
9. Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K., Woods, E.: Tslearn, a machine learning toolkit for time series data. JMLR **21**(118), 1–6 (2020)
10. Truong, C., Oudre, L., Vayatis, N.: Selective review of offline change point detection methods. Signal Processing **167**, 107299 (2020)