# Detecting Events in Aircraft Trajectories: Rule-based and Data-driven Approaches

Xavier Olive, Junzi Sun, Adrien Lafage and Luis Basora

The 8th OpenSky Symposium

12/13 November 2020, *from the comfort of your home*

We introduce here different

**rule-based,**
**empirical,**
**deterministic,**

and

**statistical,**
**data-driven,**
**machine-learning**

methods to detect a selection of these events

## Xavier Olive

xoolive

I like maps 🌍, code 🐍 and data visualisation 😎. I keep an eye on anything that can fly 🛩.

## Junzi Sun

junzis

Assistant Professor at TU Delft, conducts research in the areas of air traffic management, aircraft surveillance, performance modeling, and data science.

## Adrien Lafage

alafage

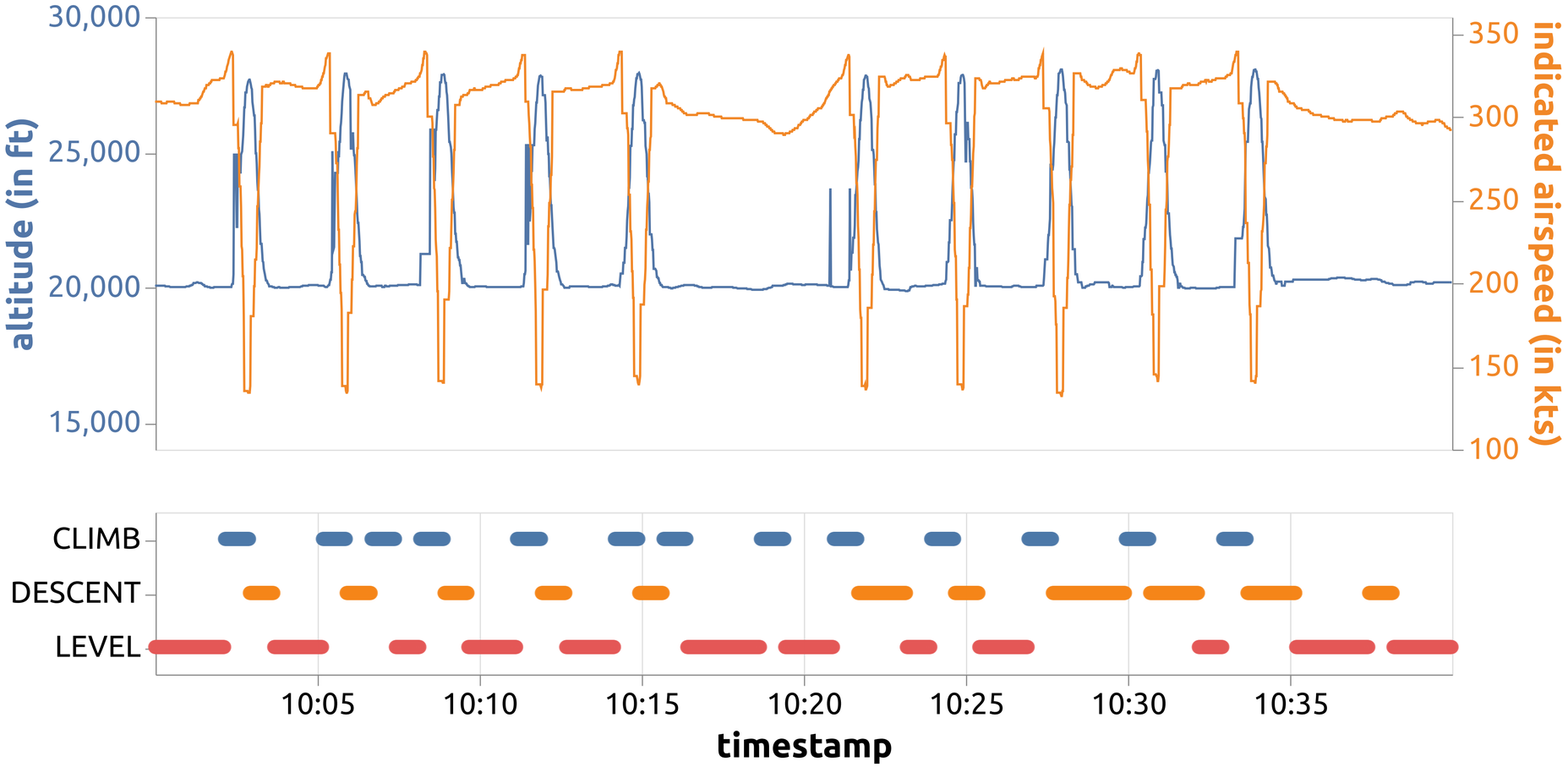
Science fiction enthusiast, I'm a French engineering student passionate by Artificial Intelligence and whatever related to it.

## Luis Basora

lbasora

Research engineer at ONERA, working in risk/performance assessment and data analysis applied to air traffic management and predictive aircraft maintenance.

# Example of a zero gravity flight



A taxonomy of events with

 specific missions

 navigational events

 flight phases

**Warning!** Different levels of granularity may overlap

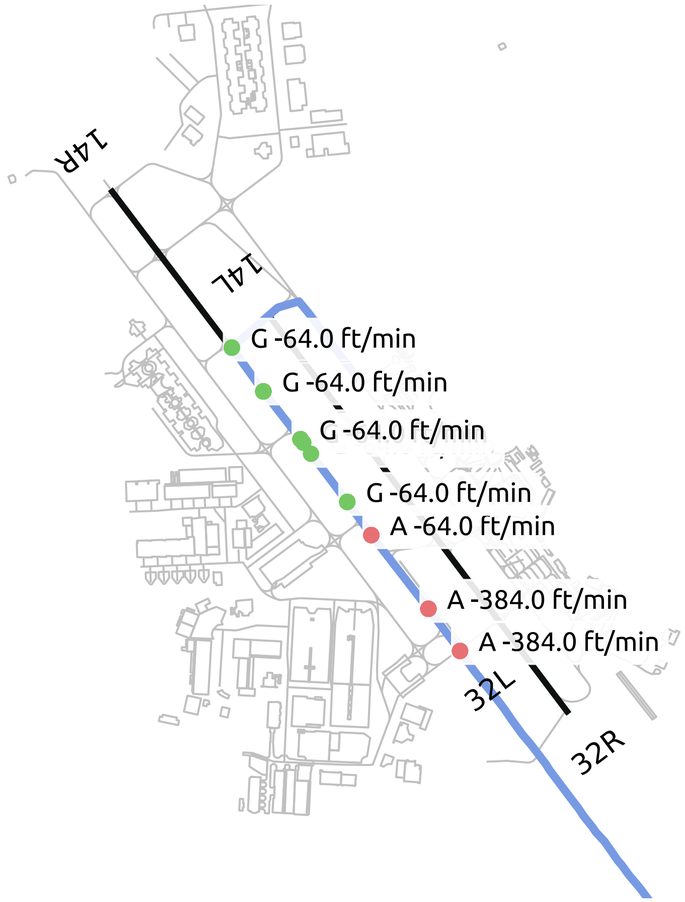
External sources of information

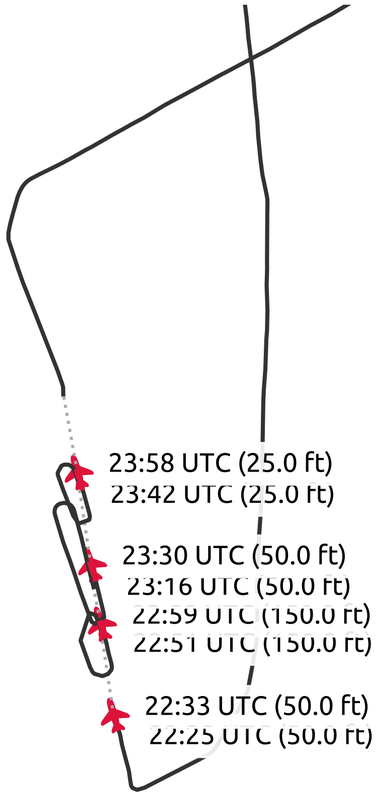
https://atmdata.github.io/

- aircraft registration databases
- structure of the airspace

*more in satellite data, newspapers, social networks, and more*

# Detecting take-off and landing



## Rule-based methods

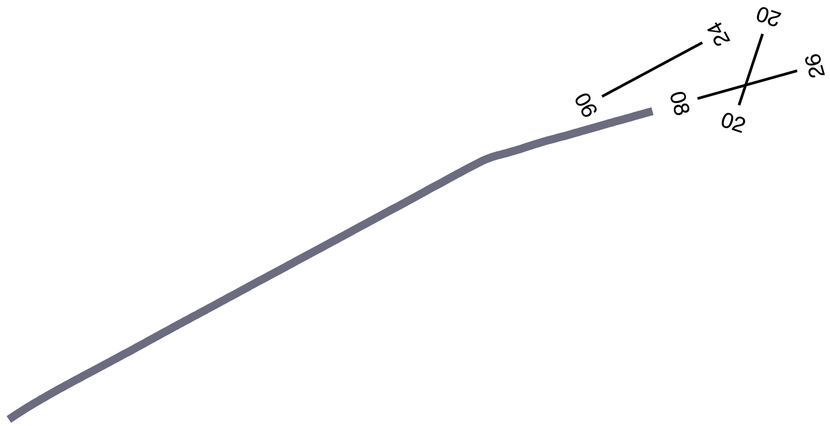
basic, direct analysis (icao24, callsign, onground)

geodetic computation for navigational events

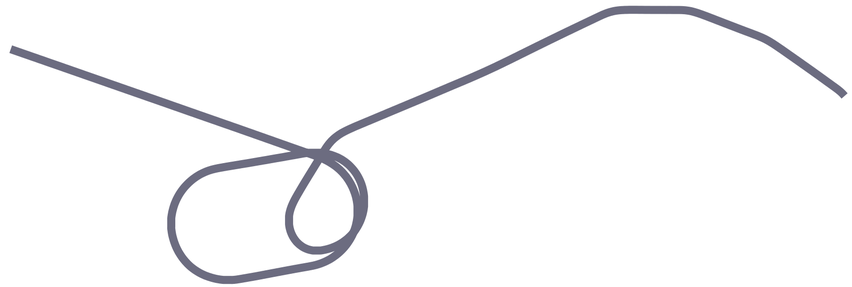
fuzzy logic for flight phases

Comparing rule-based and ML-based on two use cases

## runway changes




## holding patterns



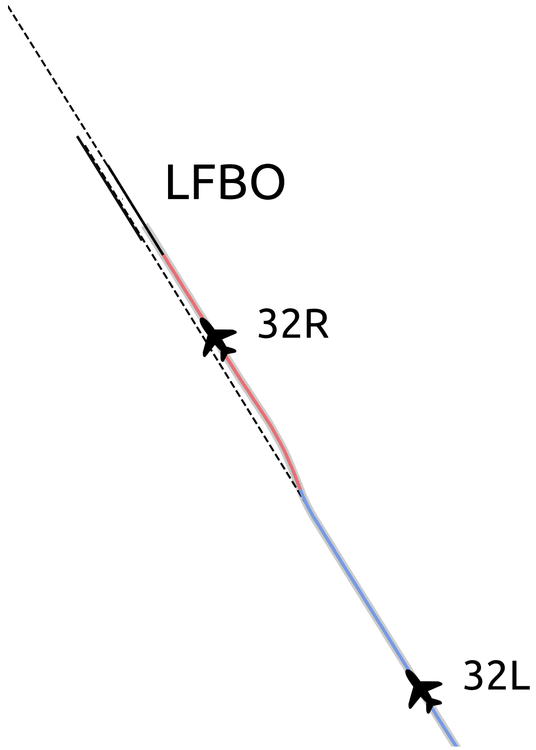
**Hard spot:** Corner cases are difficult to handle with both approaches.

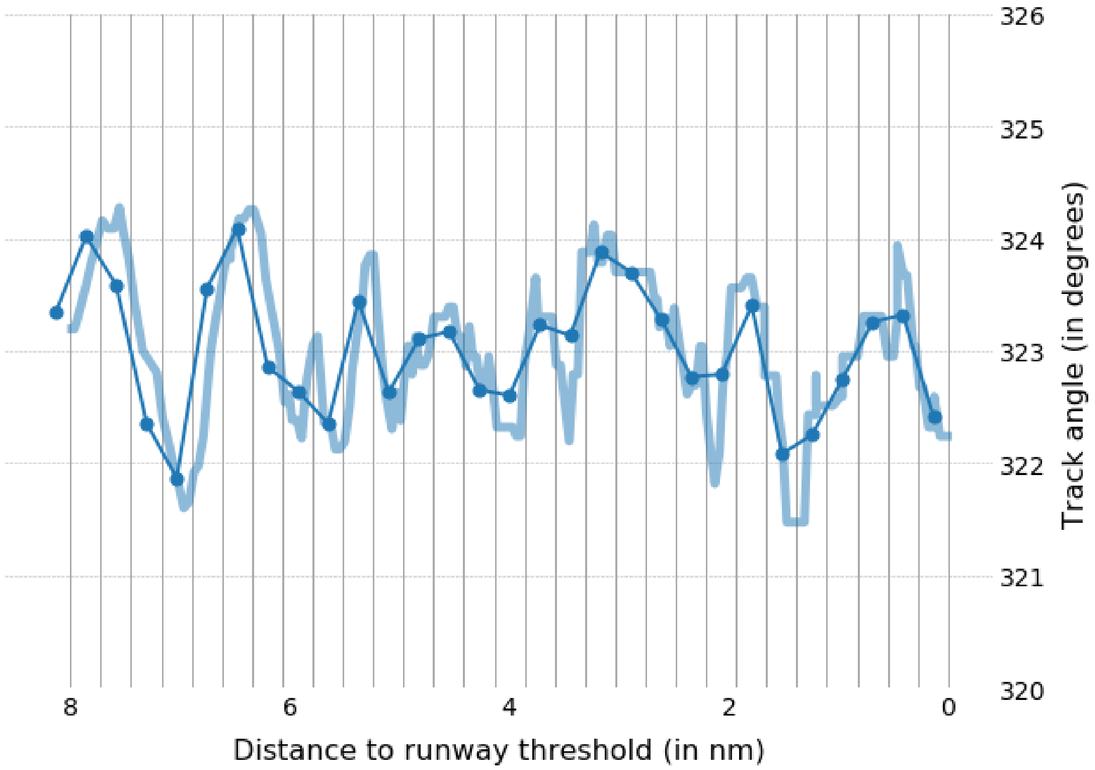
# Runway changes, rule-based

https://github.com/xoolive/traffic/blob/master/traffic/algorithms/navigation.py

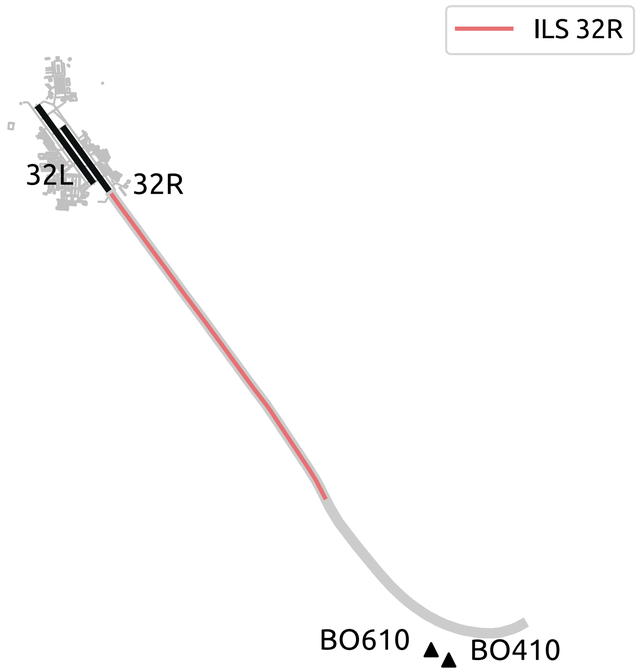
```
335     @flight_iterator
336     def runway_change(
337         self,
338         airport: Union[str, "Airport", None] = None,
339         dataset: Optional["Airports"] = None,
340     ) -> Iterator["Flight"]:
341         """Detects runway changes.
342
343         The method yields pieces of trajectories with exactly two runway
344         alignments on the same airport not separated by a climbing phase.
345
346         In each piece of yielded trajectory, the `ILS` column contains the
347         name of the runway targetted by the aircraft at each instant.
348         """

362         aligned = iter(self.aligned_on_ils(airport))
363         first = next(aligned, None)
364         if first is None:
365             return
366
367         for second in aligned:
368             candidate = self.between(first.start, second.stop)
369             assert candidate is not None
370             candidate = candidate.assign(ILS=None)
371             if candidate.phases().query('phase == "CLIMB"') is None:

379                 yield candidate
```

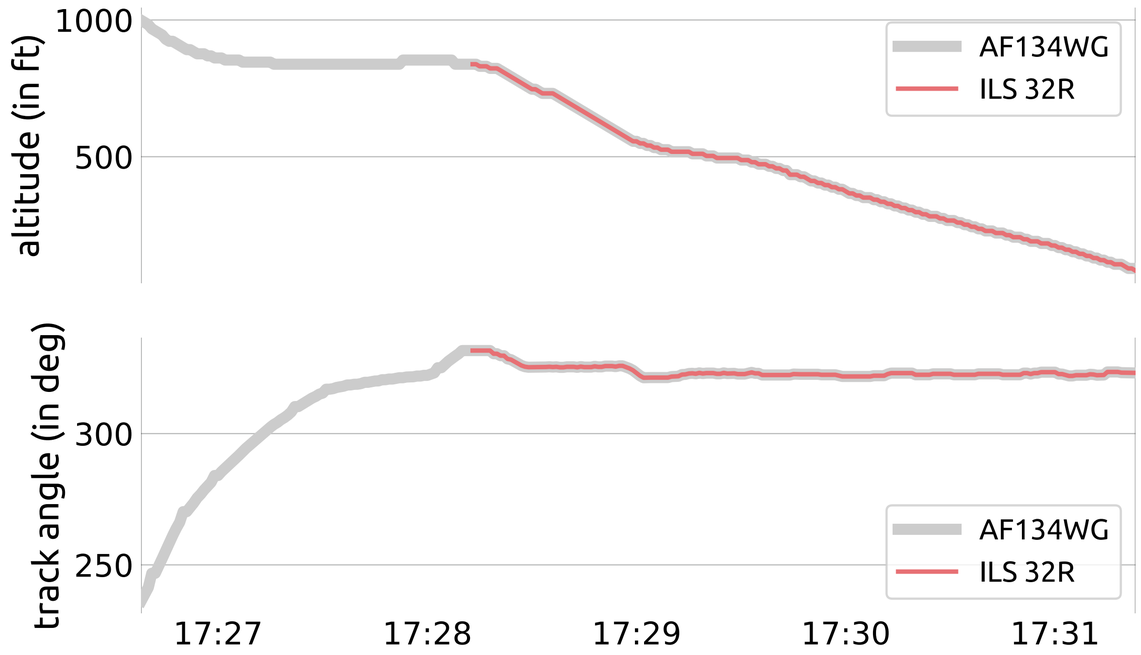
# Runway changes, f-PCA based



# Example of flight missed by the rule-based method



Detection of holding patterns

- decompose along sliding windows
- define a projection operator to a lower dimension space
- fit a classification model on this latent space.

# Detection of holding patterns

Clear definition, but wide variance around it

Rule-based implementations:
- self-intersections, parallel tracks

How to label the start and end of a holding pattern?

Large-scale analysis of trajectories requires

flight phase, pattern and

procedure identification

in order to bring useful insight to

situation awareness,

monitoring and decision-making systems

1. simple rules

1. simple rules
2. external knowledge is essential

1. simple rules
2. external knowledge is essential
3. rule-based: easy to describe, robust

1. simple rules
2. external knowledge is essential
3. rule-based: easy to describe, robust
4. ML-based: extract knowledge
4b. expertise remains essential

1. simple rules
2. external knowledge is essential
3. rule-based: easy to describe, robust
4. ML-based: extract knowledge

4b. expertise remains essential

5. compare approaches without bias

https://github.com/xoolive/traffic

All the presented methods are implemented in latest release.

The code to generate all the figures is available in

`opensky/opensky2020.ipynb`

Documentation (under continuous improvement)

https://traffic-viz.github.io/