

tangram

An open platform for modular, real-time air traffic management research

Xavier Olive Junzi Sun Xiaogang Huang Michel Khalaf

3rd/4th April 2025

History & Background

ICRAT 2020

Detecting and Measuring Turbulence from Mode S Surveillance Downlink Data

Xavier Olive
ONERA – DTIS
Université de Toulouse
Toulouse, France

Junzi Sun
Faculty of Aerospace Engineering,
Delft University of Technology,
Delft, the Netherlands

Going real-time

Idea of the paper: based on historical Mode S data, we labelled turbulence, assessed the temporal and spatial consistency in results, compared with SIGMETs, etc.

Informal chat with Bordeaux ACC: “can we try it real-time?”

- Ok... so, let's try to go real-time, we just need to get the raw data feed, decode it and run the same algorithm. “Hold my beer...”

Going real-time

Idea of the paper: based on historical Mode S data, we labelled turbulence, assessed the temporal and spatial consistency in results, compared with SIGMETs, etc.

Informal chat with Bordeaux ACC: “can we try it real-time?”

- Ok... so, let's try to go real-time, we just need to get the raw data feed, decode it and run the same algorithm. “Hold my beer...”
- Ok... not really...

Going real-time

Idea of the paper: based on historical Mode S data, we labelled turbulence, assessed the temporal and spatial consistency in results, compared with SIGMETs, etc.

Informal chat with Bordeaux ACC: “can we try it real-time?”

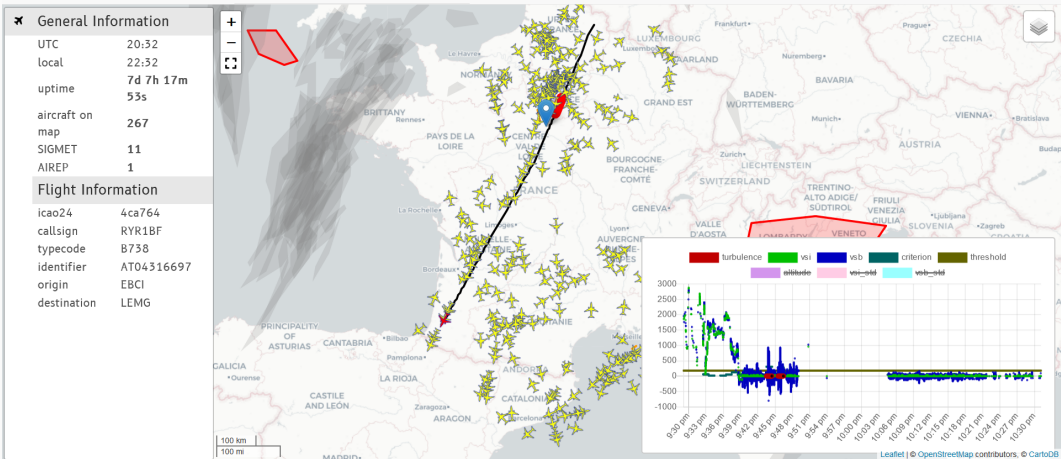
- Ok... so, let's try to go real-time, we just need to get the raw data feed, decode it and run the same algorithm. “Hold my beer...”
- **Ok... not really...**

We hired a Master student, part-time for one year (2021–2022), to develop a proof of concept.

- **Michel Khalaf**, learns about aviation, ADS-B, Python, JavaScript, concurrency, distribution, pyModeS, traffic, web services, etc. and gets a first platform running

Proof of concept

Mode S based turbulence detection Live Database History Set Parameters



“We expect a lot of turbulence today, can we try it now?”

“The proxy server could not handle the request GET /turbulence/stable. Reason: Error reading from remote server.”

“Please select ‘Disable cache’ in the Network tab of your browser’s developer tools.”

“Ok, it’s working now, but it’s very slow.”

“Let me check the server logs... oh, the CPU is at 100%.”

“So we just phoned the tower in Perpignan, the pilot says there was not much turbulence.”

ATCOs are ideal “guinea pigs” for your projects: curious, open-minded, tech-savvy and very happy to help and provide feedback.

We hit serious performance issues when more than one device was connected.

IT was constantly complaining about CPU usage on DMZ servers.

(Why do they have my phone number?)

It is very difficult to take feedback into account when working on real-time data.

Suite à ma présentation de l'avancée de nos travaux, le conseiller du directeur des opérations de la DSNA a émis l'idée que le sujet était intéressant et qu'il "devrait faire l'objet d'un accord de partenariat spécifique avec l'ONERA pour cadrer les aspects juridiques et propriété intellectuelle."

Following my presentation of our work's progress, the operations director's advisor at DSNA remarked that the topic was interesting and "should be the subject of a specific partnership agreement with ONERA to address legal and intellectual property issues."

Then Michel graduates... (2022)

- “So, what do we do now?”

Lessons learned:

- Maintaining a real-time platform is hard: we don't have time to do that
- Maybe this was not all about turbulence, but about the platform itself
- We should make it easier to develop and test new ideas
- We don't have any frontend skills, and low backend competence
- Python is... great, but “*it's complicated*” kind of relationship
- We want to test methods on real-time data, not to develop a platform
- Michel spent most of his time on the platform, not on the method

Next steps

What do we want?

- FlightRadar24 is a great piece of engineering, They can show off their ideas, and they do, but we can't...
- OpenSky Network makes it legal to use the data, also real-time, but on a low sample rate, and only state vectors
- Getting a legit real-time data feed is not easy, even with OpenSky, but we have our own antennas/data feed
- We have all the history we need with OpenSky, let's focus on **real-time only** History and real-time are two different jobs

What can we do?

- Write proposals, get funding

Project database > Tangram: open platform for modular, real-time, and data-driven aviation research

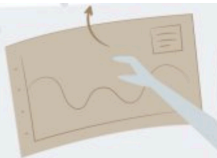
Tangram: open platform for modular, real-time, and data-driven aviation research

Tangram is an open research framework for flight surveillance data that originated as a hobby project for detecting turbulence. It has potential for various real-time aviation research topics such as GNSS jamming detection, aviation weather monitoring, emission analysis, and airport performance monitoring. The preliminary version is currently available as a web application developed in Python and JavaScript. Our final goal is to create a fully functional platform that enables real-time open data analysis on a cloud service or personal computer. Tangram will aggregate raw data, allowing researchers to incorporate their own codes for conducting open research analyses.

Open Science Fund

The Open Science Fund aims to support projects specifically designed to implement and stimulate open science practices. With this funding instrument, NWO takes a step forward towards changing the way academics are recognised and rewarded in the Netherlands.

→ Show research programme



Characteristics

Status

In progress

Duration

20 January 2024 to 30 April 2025

Research programme

Open Science Fund

File number

OSF23.1.051

Discipline

[Technology](#)

Themes

[Overig talentontwikkeling](#)

[Talent](#)

[2023-2026 \(Financieringslijnen\)](#)

Now we have a project, serious
things start (2024)

First impressions after presenting the prototype

First impressions from Junzi and Xavier:

- This prototype is really complicated to explain
- Everything is messy and slow to run

First impressions from Xiaogang (and a bit more):

- I need **containers** (have you heard of Docker?)
- We need **better tools**, libraries, framework and technologies
- tangram is **a set of components**, it should be easy to deploy them on one single container, or to distribute them over many.

Parameterization is key: tangram should work on Linux, should work on MacOS, should work at home, should work at the office, should make IT happy, should work on the cloud, work on the train, work on the plane, work at the beach, work on the Moon, etc.

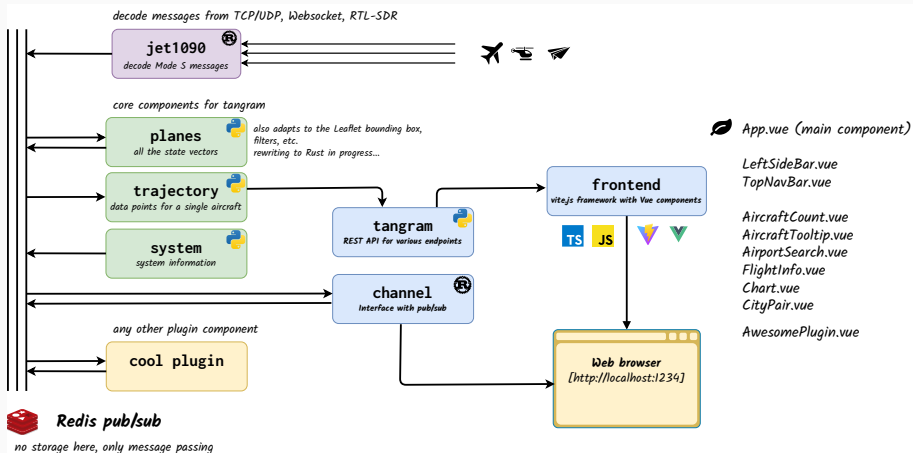
Technologies used

- Backend remains based on/documented for **Python**:
 - **FastAPI** for the REST API
 - **Redis** (pub/sub framework) passes messages between components
- Frontend is based on **Javascript** (Vite, Vue.js)
- **Podman** (\approx Docker) for containerization
- Configuration through **.env** files (environment variables)
- **Just** for task automation (\approx Makefile)
- **Rust** for performance-critical components ¹

Finding the right architecture has been an iterative process, it takes time, and it's not over


¹Do not over-engineer: Rust is great, but prototyping in Python is much easier

Overall structure (core components)



Backend: jet1090

- **jet1090** takes the best of dump1090 and pyModeS (in Rust)
- decodes real-time ADS-B and Mode S data decoding.
- Input: raw data from antennas (TCP, UDP, Websocket, TCP over ssh, RTL-SDR, etc.)
- Output: decoded data (JSON) on stdout, table view, REST API, Redis pub/sub
- <https://github.com/xoolive/rs1090>, documentation: <https://mode-s.org/jet1090>



The screenshot shows a terminal window titled "jet1090" with a dark background. It displays a table of decoded ADS-B data. The table has 20 columns: icao24, callsig, type, sqwk, lat, lon, alt, sel, gs, tas, ias, mach, vrte, trk, hdg, roll, nac, last, and first. The data is presented in a light blue monospace font. At the bottom of the terminal, there is a status bar that reads "jet1090 (685 aircraft)" and a help message: "(Esc/Q) quit | (↑/K) up | (↓/J) down | (↵/G) top".

icao24	callsig	type	sqwk	lat	lon	alt	sel	gs	tas	ias	mach	vrte	trk	hdg	roll	nac	last	first
e48df6	TAM8085	B77W	2211	49.731	-2.572	31000	=	508	482	306	0.82	384	212.7	216.7	0.175	11		21:25
e48ba8	TAM8071	B77W	2563	46.891	-5.322	32000	=	478	482	301	0.82	64	245.3	251.8	0	11		21:24
ae1178	RCH290	C17	6637			34000												21:24
abfa8e	FDX5225	B77L	4113	51.341	2.6474	32000	=	474	494	311	0.85	32	280.1	288.2	0	10		21:24
abf0c7	FDX5232	B77L	1000	49.028	2.4015	4250	100	218	232	223	0.36	2048	326.6	326.7	11.42	10		21:24
a6329d	GTI9549	B744	3532	42.833	-4.887	38175	370	494	484	272	0.85	-896	103.8	104.0	-0.52	9		21:24
a53013	UPS251	B752	2003	51.283	1.5045	33025	=	471	462	284	0.8	0	100.7	92.10	-0.17	11		21:24
a50c4b	UPS273	B752	2061	51.465	9.2144	22750	120	487	438	317	0.72	-2112	240.8	249.0	-4.21	11		21:24
89655b	UAE4	A388	2247	51.242	1.1701	22350	290	450	428	322	0.72	1440	99.31	93.51	0	10		21:24
896538	ETD5GZ	B789	3146	50.256	9.7858	36950	=	515	484	275	0.84	-128	121.5	108.8	-0.52	9		21:24
8964b0	UAE28Y	A388	4615	49.809	8.3023	37000	=	522	494	284	0.86	0	112.1	97.38	-0.35	10		21:24
8964a2	UAE8T	B77W	3015	47.550	8.5364	6325	120	224	226	209	0.35	2848	81.91	73.12	-2.81	9		21:24
896471	UAE7L	A388	0410	45.439	10.466	28300	330	438	446	301	0.76	1280	102.9	92.98	-0.70	10		21:24

- **Redis** is a fast, open-source, in-memory key-value data store for use as a database, cache, message broker, and queue.
- Here, **Redis** is used as a pub/sub framework to pass messages between components.

Components may:

- send messages with a topic on a Redis channel;
- subscribe to specific topics and receive messages matching it.

- **channel** makes the interface between the Redis pub/sub and the browser.
- On the browser side, **channel** serves as a Websocket server.
- <https://github.com/emctoo/channel>

The main idea behind Websockets is to maintain a **persistent, bidirectional connection** between client and server: real-time data exchange with minimal overhead (esp., no query mechanism).

Rule of thumb: if you need to query something regularly, it may be preferable to switch to a Websocket based mechanism.

- **tangram** provides a simple RestAPI end point, based on FastAPI
- For example, it serves the full trajectories for a single aircraft, useful on the map and for the plot charts.

Frontend: Vue components

Vue components are reusable **self-contained pieces** of user interface:

- a `<template>` section with HTML and placeholders,
- a `<script>` section with Javascript,
- a `<style>` is CSS (can be scoped)

```
<template>
  <div class="component">
    <h1>{{ title }}</h1>
  </div>
</template>
```

```
<script>
export default {
  name: "HelloWorldComponent",
  data() {
    return {
      title: "Hello Vue!",
    };
  }
}
```

Frontend: a Vite/Vue based framework

The general architecture is described in `App.vue`:

- web pages are made of components;

In **tangram**:

- we distinguish **core components** (regular Vue files);
- and **plugin components**
(can be added, replaced, overridden and placed outside the tangram codebase)
- Variables can be passed between components through a “store”
from the state management library Pinia <https://pinia.vuejs.org/>

Demonstration and live coding

Overview



463 [1271] 11:10:24 Z
visible aircraft 2:02:12

✈ AFR25GN

B:3985a3 A320

Registration: F-HBND

LFPO

LFMN

Paris

Nice

EOBT

ETOT

ETOA

10:35

10:38

11:52

AOBT

ATOT

ATOA

10:26

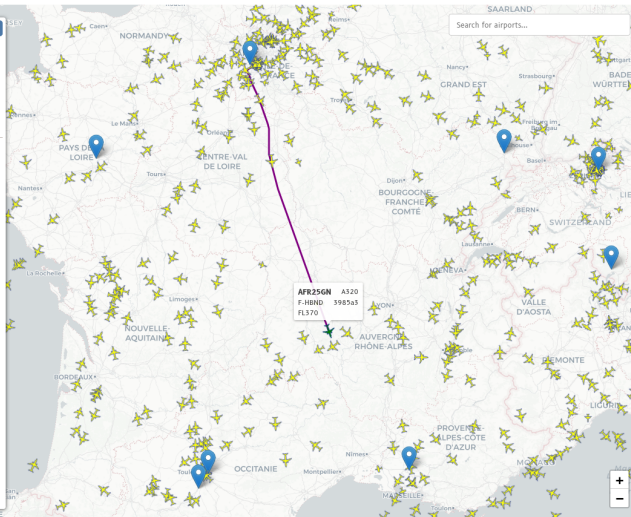
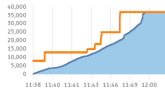
10:38

11:45

Flight data

Altitude (in ft)

— Selected altitude — Barometric altitude



Extension:

Query an external service for current SIGMET and display them on the map.

Steps:

- Build a backend service to produce the proper structure and serve through tangram
- Write a Vue component to display the SIGMETs on the map

Extension:

Highlight aircraft tracked by more than four sensors (useful for multilateration)

Steps:

- Listen to messages on Redis, maintain a structure counting sensors tracking each aircraft
- Publish the results on Redis
- Write a Javascript component to store the state of each aircraft
- Write a new CSS for highlighted aircraft

Extension:

Highlight pieces of trajectory creating contrails according to the Schmidt–Appelman criterion

Steps:

- Get a trajectory from the backend
- Get meteo data from fastmeteo
- Compute the Schmidt–Appelman criterion
- Write a new Vue component to display the contrails

Conclusion

Key take-aways

Feedback is welcome

Contributions are open

(“*Limited*”) support can be offered

Warning:

- tangram is still in alpha
- documentation is still very preliminary