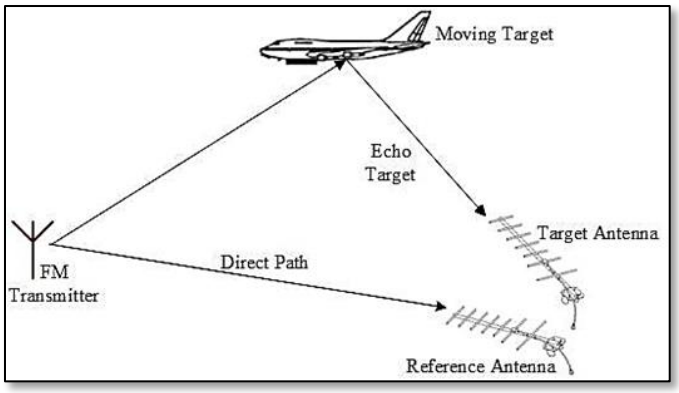
Passive Radar

Functions: FFT, linear algebra, filtering, cross-correlation, beamforming, DoA estimation
Inputs: complex-valued signal samples from radar array and reference antenna
Metrics: per-phase and total processing time

1. Overview

Passive radar is a radar mechanism where the receiver relies on an external signal source, rather than transmitting radar pulses itself. Passive radar processing is broadly similar to that of an active system, e.g. a phased array which electronically steers a radar beam using multiple antennas. However, passive radar requires an additional step to cross-correlate the external signal source (as directly received by a reference antenna) with possible reflections in the radar beam. For more high-level background, see https://apps.dtic.mil/dtic/tr/fulltext/u2/a515506.pdf or https://ieeexplore.ieee.org/document/5547434.



Passive radar detects targets by cross-correlating a directly received reference signal with its reflections in the radar beam
(source: https://qph.fs.quoracdn.net/main-qimg-a48b2c4e92ee168855d48046c244069e)

MilSpec's passive radar benchmark is based on the pyAPRiL project by Tamás Pető,[1] an open-source implementation of a full passive radar pipeline including clutter cancellation, beamforming, cross-correlation, and direction-of-arrival estimation. The MilSpec version rebuilds the baseline Python implementation in C++, incorporates the open-source libeigen[2] and fftw libraries for linear algebra and FFT processing, respectively, and adds performance instrumentation.

2. System requirements

Platform: Ubuntu 18.04 LTS with g++ 7.4.0. Code may build and run successfully on other versions/platforms, but has not been tested with them.
Storage: ~10MB for code and temporary files, plus ~200MB for sample data.
Dependencies: None.

[1] https://github.com/petotamas/APRiL
[2] https://gitlab.com/libeigen/eigen

3. Build and run

To benchmark:

- Download and extract the code zipfile from www.adacenter.org/milspec
- Ensure all input data, properly formatted as per the description in `passiveradar/src/main.cpp` (see below), are placed in the `passiveradar/dataset` directory:
    - Two elements are required: (1) the ground-truth target reference data, in `target_reference_track.txt`, and (2) the received radar returns, in one or more `dataset_[index].txt` files corresponding to sequential time indices.
    - Target reference data format:
      Line 1: `<num_rows> <num_cols>`
      Subsequent lines: `<Time index> <Timestamp> <Latitude> <Longitude> <Altitude> <Speed> <Direction> <Range> <Doppler> <Azimuth>`
    - Radar return data format:
      Line 1: `<Sampling Frequency>`
      Line 2: `<num_rows> <num_cols>`
      Subsequent lines (one per antenna element, where the first line is the reference channel, and the data in each line are the complex-valued samples collected at this file's time index): `real_1 imag_1 real_2 imag_2 ... real_num_cols imag_num_cols`
- From the `passiveradar/` directory, `make clean && make`
- From the `passiveradar/` directory, `./passive`
- Results are displayed in the terminal, as below:

```
====================== RESULTS SUMMARY ======================

Avg time to isolate channels: 0.00772775 seconds
Avg time for beamforming: 0.107594 seconds
Avg time to perform time-domain filter surveillance : 1.33224 seconds
Avg time to detect: 0.153442 seconds
Avg time to find target: 0.000795726 seconds
Avg time to estimate direction of arrival: 4.41843 seconds
Avg total time to process single time index: 8.22667 seconds

Max time to isolate channels: 0.0164568 seconds
Max time for beamforming: 0.113048 seconds
Max time to perform time-domain filter surveillance : 1.37662 seconds
Max time to detect: 0.159107 seconds
Max time to find target: 0.000853522 seconds
Max time to estimate direction of arrival: 4.44788 seconds
Max total time to process single time index: 8.28855 seconds
```

4. Code structure

Coming soon!

5. MilSpec development notes, errata, changelog

v0.9:

- Built baseline passive radar implementation:
    - Ported original pyAPRiL Python implementation to C++
    - Integrated libeigen and fftw for linear algebra and FFT computation
    - General editing/cleanup
    - Added instrumentation for per-phase and overall performance

6. Acknowledgements

MilSpec is a project under development at the University of Michigan – Ann Arbor by Pete Ehrett, Bing Schaefer, Adrian Berding, Nathan Block, John Paul Koenig, Pranav Srinivasan, Todd Austin, and Valeria Bertacco. This project is supported by the Applications Driving Architectures Center, one of six centers of JUMP, a Semiconductor Research Corporation program co-sponsored by DARPA.

The original passive radar implementation upon which the MilSpec version is based, and the sample data used for the MilSpec benchmark, was developed/collected by Tamás Pető and is available at https://github.com/petotamas/APRiL. Additional details may be found at https://www.tamaspeto.com/Vega/VEGAM20191219K4U0C0S4DVBT. The Eigen library used for linear algebra computations is available at https://gitlab.com/libeigen/eigen.