Ramp Detection Algorithm

Author: Luke Bray

This Python program is an automated signal analysis tool designed to process large raw radio signal files (`.cf32` format) to identify and characterize the initial "turn-on" transient of a transmission. Its core functionality is to efficiently locate the precise start of a signal burst within a massive dataset (potentially millions or billions of samples) and then perform targeted time-domain and frequency-domain analysis on that specific event. This automated approach replaces the tedious and inefficient process of manual inspection, enabling rapid analysis of extensive signal captures.

The program operates using a highly efficient two-stage detection methodology, followed by a multi-faceted analysis phase.

Methodology: Transient Detection

To avoid a time-consuming linear scan of the entire file, the program employs a two-stage search algorithm that combines a high-speed coarse search with a high-precision local scan.

Stage 1: Coarse Localization via Bisection Search

The primary challenge is to quickly find the general area of the transmission without reading the entire file. This is achieved using a bisection (or binary) search algorithm, which operates on the principle of "divide and conquer."

1. **File Scoping:** The program first determines the total number of samples in the file. This defines the initial search space, from sample `0` to `N-1`.

2. **Activity Check:** The core of the bisection search is a function (`_check_if_signal_active`) that can quickly determine if a signal is likely "active" or "inactive" within a given block of samples.

    - It reads a small block of samples (e.g., 50,000) from a specified location in the file.

    - It calculates the instantaneous amplitude (`sqrt(I² + Q²)`) for this block.

    - The **standard deviation** of the amplitude is then computed. A signal consisting of only background noise will have a very low, stable amplitude and thus a low standard deviation. An active signal, with its significant power variations, will have a much higher standard deviation.
    - This standard deviation is compared against a pre-defined `ENERGY_THRESHOLD`. If it exceeds the threshold, the block is considered "active"; otherwise, it is "inactive" (noise).
3. **Iterative Refinement:** The bisection algorithm iteratively narrows the search space:
    - It tests the "activity" at the midpoint of the current search space.
    - If the midpoint is **active**, it implies the transient must have occurred *before* this point. The algorithm then discards the second half of the search space.
    - If the midpoint is **inactive**, the transient must have occurred *after* this point. The algorithm discards the first half of the search space.
    - This process repeats, halving the search space with each iteration until the region of interest is narrowed down to a manageable size (e.g., a few hundred thousand samples). This logarithmic search is exceptionally fast, reducing a search space of billions to thousands in just a few dozen steps.

Stage 2: Precise Detection via Linear Ramp Scan

Once the bisection search has isolated a small, high-probability region, the program switches to a more computationally intensive but highly precise linear scan to pinpoint the exact start of the power-up ramp.

1. **Sliding Window:** The algorithm moves a small "window" (e.g., 30 samples wide) one sample at a time across the localized data region.
2. **Linear Regression:** For each position of the window, the program performs a first-degree polynomial fit (linear regression) on the amplitude data within it. This calculation yields the best-fit straight line for that data segment, providing a **slope**.
3. **Slope Thresholding:** The calculated slope is the key indicator:
    - In a region of noise, the slope will be nearly zero.

- During the power-up transient, the amplitude increases linearly, resulting in a steep, positive slope.
- The algorithm compares this slope to a `SLOPE_THRESHOLD`. The first window where the calculated slope exceeds this threshold is flagged as the beginning of the ramp. The absolute sample index of the start of this window is returned as the definitive transient start point.

Methodology: Post-Detection Analysis

Upon successful detection of the transient start sample, the program automatically initiates three distinct analysis routines, focusing on the 100 samples immediately following the detected event to characterize its properties.

1. **Amplitude vs. Time Analysis (`plot_amplitude_vs_time`):** This function plots the instantaneous amplitude against time (in seconds, calculated using the provided sample rate). This provides a clear visual confirmation of the detected ramp and the signal's subsequent behavior.
2. **I/Q Data Analysis (`plot_cf32`):** This plot visualizes the fundamental structure of the signal by displaying the raw In-phase (I) and Quadrature (Q) components against the sample number. This is useful for observing the phase and modulation characteristics at the moment of turn-on.
3. **Spectral Analysis (`plot_spectral_bins`):** This function provides a frequency-domain snapshot of the signal at the transient.
    - It applies a **Hanning window** to the data segment to minimize spectral leakage, an artifact of performing an FFT on a finite data set.
    - It computes the Fast Fourier Transform (FFT) to convert the signal from the time domain to the frequency domain.
    - It visualizes the magnitude of the **seven frequency bins** centered around the signal's DC component. This bar plot clearly shows the distribution of signal energy at and immediately around the carrier frequency during power-up.

By integrating these functions, the program provides a comprehensive, automated, and highly efficient solution for locating and characterizing transient signal events in large-scale radio data captures.