Design a minimal towed-array processing pipeline

Here's a **minimal, practical towed-array (passive) processing pipeline** that you can build end-to-end, starting from multichannel time-series and ending with bearings + tracks. I'll keep it "minimum moving parts," but still real.

0) Inputs & assumptions

- **x[m, n]**: M hydrophones, sampled at **Fs** (you said 10 kHz before), spacing **d** (you said 1 m), sample index **n**
- **Sound speed c** (use ~1500 m/s as a first constant)
- Array is (approximately) **linear** over short windows

1) Acquisition → sanity → calibration

Goal: make channels comparable.

- Remove DC, check clipping, drop dead channels.
- **Time sync / delay calibration**: compensate fixed per-channel delays (measured in lab or estimated from known pinger).
- **Gain/phase calibration** if you have it (optional for "minimal," but helps).

2) Band selection & conditioning

Goal: isolate useful band and stabilize noise.

- **Bandpass filter** to the band where targets live (example: 50–1500 Hz, depends on your scenario).
- Optional: **notch** ship tones (50/60 Hz + harmonics).
- Optional: **prewhitening** (simple: estimate PSD and divide; or use 1st-order difference).

Output: conditioned multichannel **y[m, n]**.

3) Frame → FFT (STFT)

Goal: operate in time-frequency blocks.

- Choose frame length **Nfft** (e.g., 1024 @ 10 kHz → 102.4 ms)
- Overlap 50–75%

- Window (Hann)

Compute **Y[m, f, t]**.

4) Noise PSD estimate (per frequency)

Goal: normalize changing sea state / self-noise.

- Track noise with a **running median** or **minimum statistics** over time:
  - $P_N[f] \leftarrow \alpha P_N[f] + (1-\alpha)\,\mathrm{median}_t(|Y|^2)$

This enables stable detection + weighting.

5) Narrowband beamforming (the core)

Goal: produce a bearing-time (or bearing-freq-time) map.

For a linear array, steering vector at frequency f for bearing θ:

- $a_m(f,\theta) = e^{-j2\pi f \tau_m(\theta)}$
- $\tau_m(\theta) = \frac{(m-(M-1)/2)\,d\sin\theta}{c}$

Delay-and-sum beam power (minimal and robust):

- $B(f,\theta,t) = \left|\sum_{m=0}^{M-1} w_m\, Y[m,f,t]\, a_m^*(f,\theta)\right|^2$

Pick **weights** w_m = uniform or Hann across sensors.

Angle grid: e.g., −90°..+90° in 0.5–1° steps.

6) Broadband integration (turn the cube into a map)

Goal: a single bearing-time heatmap.

- Select frequency bins in band of interest
- Weight by SNR:
  - $\mathrm{BT}(\theta,t) = \sum_{f\in\mathcal{F}} \frac{B(f,\theta,t)}{P_N[f]+\epsilon}$

Now you have **bearing-time (BT)**.

7) Detection on BT

Goal: pick candidate bearings per time.

Minimal detection:

- For each time t, find **peaks in BT(θ,t)**
- Threshold using **CFAR**-style rule:

- threshold = k × median(BT over θ) or percentile
- Keep top-K peaks + their SNR.

Output: observations **z_t = { (θ_i, score_i) }**.

8) Tracking (very minimal, but useful)

Goal: smooth bearings, maintain multiple contacts.

Two easy options:

A) Simple track smoothing

- For strongest peak each frame: θ̂(t)
- Apply median filter + Kalman filter on θ(t) with constant-rate model.

B) Multi-target tracks (still minimal)

- Maintain track list with predicted θ
- Associate peaks via nearest-neighbor + gating
- Initiate if detected consistently N times, delete if missed M times.

Output: stable **bearing tracks**.

9) (Optional, but common) DEMON / tonals classification

If you want classification cues:

- Extract the beamformed time series at θtrack (or a small θ band).
- Run **tonal detection** (spectrogram line tracking) or **DEMON** (envelope spectrum) for propeller shaft/blade rates.

Minimal "build order" (what to implement first)

1. STFT per channel
2. Delay-and-sum beamformer → **BT map**
3. Peak pick + threshold
4. Simple tracker

That alone gives you a usable contact picture.

Key parameter cheatsheet (for Fs=10 kHz, M=50, d=1 m)

- Nfft: **1024–4096** (102–409 ms). Start with **2048**.
- Overlap: **75%**.
- Angle step: **1°** (faster) then **0.5°** (better).
- Band: start **50–1500 Hz** (adjust).
- Beware spatial aliasing: for linear arrays, aliasing starts when **d > λ/2** → $f > c/(2d)$.
  - With d=1 m, $f_{alias} \approx 1500/(2 * 1) = 750\ \mathrm{Hz}$.
  - So above ~750 Hz, your bearings can alias unless you handle it (limit band, or accept ambiguity, or use more advanced methods).

If you want, I can give you:

- a compact **math** + **pseudocode** version,
- or a **C#/Python implementation skeleton** (including steering vector + FFT framing),
- or how to extend this to **MVDR/Capon** and **adaptive nulling** once the minimal pipeline works.