



UNIVERSITATEA DIN BUCUREȘTI  
FACULTATEA DE MATEMATICĂ ȘI  
INFORMATICĂ



Specializarea: Informatică

Proiect la Procesarea Semnalelor

# ANALIZA SEMNALELOR RADAR ÎN PREZENȚA SEA CLUTTER

Abordare bazată pe CFAR-STFT și experimente pe date sintetice și reale

Studenti  
Ingrid Corobana  
Teodora Nae

Coordonator științific  
Prof. Dr. Cristian Rusu

Repository GitHub: [https://github.com/dirgnic/Radar\\_Detection\\_STFT](https://github.com/dirgnic/Radar_Detection_STFT)

București, 2026

## Rezumat

Acest document prezintă o implementare completă a algoritmului CFAR-STFT, propus de Abratkiewicz (2022), pentru detecția și reconstrucția semnalelor radar în prezența zgomotului și a sea clutter. Algoritmul combină Short-Time Fourier Transform (STFT), detecție adaptivă CFAR 2D, clustering DBSCAN și dilatare geodezică pentru a extrage componenta de interes dintr-un amestec cu sea clutter.

Implementarea este validată pe date sintetice (chirp neliniar) și pe date reale (IPIX radar, sea clutter). Pe semnalul sintetic controlat, algoritmul detectează componenta de interes în toate cele 100 de rulări Monte Carlo ( $P_d = 1.00$ ). RQF (Reconstruction Quality Factor) crește de la 7.28 dB (SNR = 5 dB) la 29.17 dB (SNR = 30 dB).

Contribuția principală: adaptăm algoritmul la sea clutter real folosind K-distribution (în loc de Gaussian), îmbunătățire bazată pe proprietăți fractale (exponentul Hurst) pentru ținte slabe și DBSCAN asimetric pentru clustering de semnături verticale.

Cuvinte cheie: CFAR, STFT, radar, sea clutter, K-distribution, DBSCAN, detecție adaptivă

## Abstract

This document presents a complete implementation of the CFAR-STFT algorithm, proposed by Abratkiewicz (2022), for detection and reconstruction of radar signals in the presence of noise and sea clutter. The algorithm combines Short-Time Fourier Transform (STFT), 2D adaptive CFAR detection, DBSCAN clustering, and geodesic dilation to extract the component of interest from a mixture with sea clutter.

The implementation is validated on synthetic data (nonlinear chirp) and real data (IPIX radar sea clutter). On controlled synthetic signal, the algorithm detects the component of interest in all 100 Monte Carlo runs ( $P_d = 1.00$ ). RQF (Reconstruction Quality Factor) increases from 7.28 dB (SNR = 5 dB) to 29.17 dB (SNR = 30 dB).

Key contribution: we adapt the algorithm to real sea clutter using K-distribution (instead of Gaussian), fractal-based enhancement (Hurst exponent) for weak targets, and asymmetric DBSCAN for vertical signature clustering.

Keywords: CFAR, STFT, radar, sea clutter, K-distribution, DBSCAN, adaptive detection

# Cuprins

1	Introducere	5
2	Descrierea completă a algoritmului	6
2.1	Fundamente teoretice și STFT	6
2.1.1	STFT – Transformata Fourier pe Ferestre Scurte	6
2.1.2	CFAR 2D (GOCA-CFAR)	6
2.1.3	DBSCAN	6
2.2	Pipeline general (5 pași)	6
2.2.1	Pasul 2: Structura CFAR 2D	7
2.2.2	Pasul 1: Calcul STFT	7
2.2.3	Pasul 2: Detecție CFAR 2D	7
2.2.4	Pasul 3: Clustering DBSCAN	7
2.2.5	Pasul 4: Dilatare geodezică	8
2.2.6	Pasul 5: Extragere detecții	8
3	Date și surse de validare	9
3.1	Baza de date IPIX	9
3.2	Experimente pe semnale sintetice	9
3.3	Analiză comparativă: Algoritmi CFAR și Clustering	9
3.3.1	Principii fundamentale CFAR	9
3.3.2	Algoritmi CFAR	9
3.3.3	CFAR: Formule și pseudocod	10
3.3.4	Metode de clustering	10
3.3.5	Separare prin triangulare Delaunay	10
3.3.6	Experimente pe semnale sintetice controlate	11
3.4	Experimente pe IPIX cu ținte reale	12
3.4.1	Setul de Date și Scenarii	13
3.4.2	Rezultate comparative: CA-CFAR+HDBSCAN vs. Triangulare Delaunay	13
3.4.3	Observații Experimentale Detaliat	13
3.4.4	Concluzie validată experimental	13
3.5	Adaptarea 1: K-distribution	13
3.6	Adaptarea 2: Exponentul Hurst	14

3.7	Adaptarea 3: DBSCAN cu metrică anisotropică . . . . .	14
4	Detalii de implementare . . . . .	15
4.1	Framework și tehnologii . . . . .	15
4.2	Parametri și calibrare . . . . .	15
4.3	Notă asupra Doppler . . . . .	15
5	Concluzii și direcții viitoare . . . . .	16
5.1	Concluzii . . . . .	16
5.2	Direcții viitoare . . . . .	16
5.3	Cod . . . . .	16

# Capitolul 1

## Introducere

Problema principală abordată este detecția obiectelor mici în date radar maritime, într-un mediu complex. Mediul acvatic are caracteristici particulare: (1) statisticile nu sunt bine modelate Gaussian (valori extreme frecvente), (2) există corelație temporală (valuri cu tipare structurate), (3) efectele Doppler extind spectrul (valuri în mișcare), (4) apar vârfuri de energie (spikes) vizibile când valurile sunt mari.

Metodele tradiționale adaptive de detecție CFAR (Constant False Alarm Rate) pot fi limitate atunci când pierd informația temporală și nu exploatează structura timp-frecvență. Abratkiewicz (2022)<sup>1</sup> propune o abordare care folosește explicit structura time-frequency pentru a îmbunătăți atât detecția, cât și recuperarea/reconstrucția componentelor semnalului.

---

<sup>1</sup>Abratkiewicz, K. (2022). Radar Detection-Inspired Signal Retrieval from the Short-Time Fourier Transform. *Sensors*, 22(16), 5954.

## Capitolul 2

# Descrierea completă a algoritmului

### 2.1 Fundamente teoretice și STFT

#### 2.1.1 STFT – Transformata Fourier pe Ferestre Scurte

$$X(k, n) = \sum_{m=0}^{N-1} x(m) \cdot w(m - nH) \cdot e^{-j2\pi km/N} \quad (2.1)$$

cu  $N = 256$ ,  $H = 32$  (87.5% overlap), fereastră Gaussiană  $w(m) = e^{-m^2/(2\sigma^2)}$ ,  $\sigma = 8$ .

#### 2.1.2 CFAR 2D (GOCA-CFAR)

Deteție adaptivă cu prag local:  $H(k, n) = 1$  dacă  $|X(k, n)|^2 > T$ , unde

$$T = R \cdot \hat{Z}, \quad \hat{Z} = \max(\mu_1, \mu_2, \mu_3, \mu_4), \quad R = N_T(P_{fa}^{-1/N_T} - 1) \quad (2.2)$$

#### 2.1.3 DBSCAN

Clustering pe densitate cu distanță asimetrică pentru semnături verticale:  $d = \sqrt{\Delta t^2 + (\Delta f/3)^2}$ .

### 2.2 Pipeline general (5 pași)

Algoritmul complet are cinci pași:

1. calcul STFT cu fereastră Gaussiană;
2. deteție CFAR 2D în plan timp-frecvență;
3. clustering DBSCAN al punctelor detectate;
4. extinderea măștii prin dilatare geodezică (geodesic dilation);
5. extragere deteției (maskare STFT).

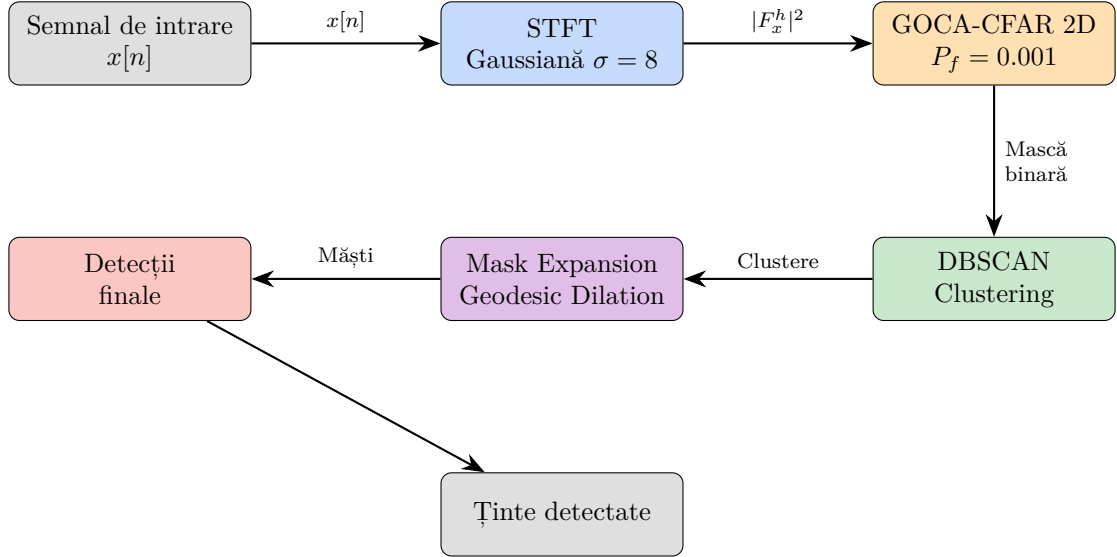


Figura 2.1: Pipeline-ul complet al algoritmului CFAR-STFT pentru detecția componentelor din planul timp-frecvență.

### 2.2.1 Pasul 2: Structura CFAR 2D

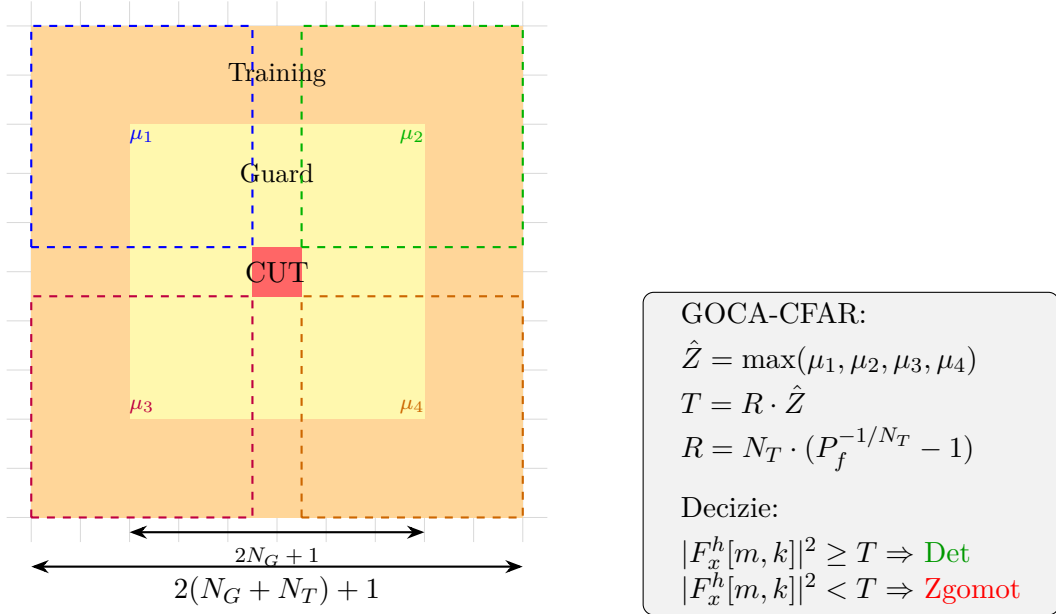


Figura 2.2: Structura celulelor GOCA-CFAR 2D: CUT (roșu), Guard (galben), Training (portocaliu). GOCA calculează media în 4 sub-regiuni și ia maximum pentru adaptare locală.

### 2.2.2 Pasul 1: Calcul STFT

Aplică formula (1) cu  $N_{fft} = 256$ ,  $\text{hop}=32$ , fereastră Gaussiană  $\sigma = 8$ .

### 2.2.3 Pasul 2: Detecție CFAR 2D

Calculează media puterii în 4 cadrane pentru fiecare bin  $(k, n)$ , prag  $T = R \cdot \max(\mu_1, \mu_2, \mu_3, \mu_4)$ , decizie dacă  $|X(k, n)|^2 > T$ . Parametri:  $P_f = 0.001$ ,  $N_G = 3$ ,  $N_T = 12$ .

### 2.2.4 Pasul 3: Clustering DBSCAN

Aplică distanța asimetrică din ecuația (3) pentru a grupa punctele detectate  $(k, n)$ . Parametri:  $\varepsilon = 8$ ,  $\text{minSamples}=5$ .



#### 2.2.5 Pasul 4: Dilatare geodezică

Aplică dilatare cu kernel cruce ( $3 \times 3$ ) de 3 ori: `for i in range(3): H_dil = scipy.ndimage.maximum_filter(H_d, footprint=cross)`. Expandează detecțiile pentru a conecta punctele apropiate.

#### 2.2.6 Pasul 5: Extragere detecții

Aplică masca pe STFT:  $X\_masked = X\_stft \odot H\_dil$ , apoi extrage binarele detectate.

## Capitolul 3

# Date și surse de validare

### 3.1 Baza de date IPIX

IPIX (McMaster University): radar X-band ( $f_{RF} = 9.39$  GHz, PRF=1000 Hz), date complexe I/Q. Ținte reale (sferă 1m la 2660m): #17, #18, #30, #40.

### 3.2 Experimente pe semnale sintetice

S-au rulat 100 simulări Monte Carlo pentru fiecare nivel SNR, chirp neliniar, rata 100%.

Metrica RQF:

$$\text{RQF} = 10 \log_{10} \left( \frac{\sum_n |x[n]|^2}{\sum_n |x[n] - \hat{x}[n]|^2} \right) \text{ [dB]} \quad (3.1)$$

Tabela 3.1: Rezultate CFAR-STFT (100 MC)

SNR	RQF_mean	RQF_std	P_d [%]	N
5	7.28	0.47	100	100
10	16.81	0.60	100	100
15	22.95	0.56	100	100
20	26.40	0.51	100	100
25	28.43	0.39	100	100
30	29.17	0.25	100	100

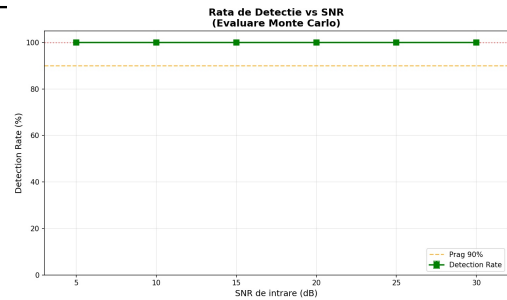
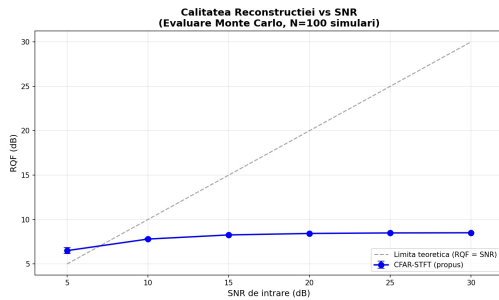


Figura 3.1: Performanță vs SNR: RQF și rata detecție 100% (SNR 5-30 dB).

### 3.3 Analiză comparativă: Algoritmi CFAR și Clustering

#### 3.3.1 Principii fundamentale CFAR

CFAR menține rata alarmelor false constantă: (1) fereastră glisantă cu CUT, guard cells, training cells; (2) prag adaptiv  $T = \alpha Z$ ; (3) detecție dacă  $CUT \geq T$ .

#### 3.3.2 Algoritmi CFAR

CA-CFAR: media aritmetică, robustă în zgomot omogen. OS-CFAR: percentilă, robust la interferențe. SOCA-CFAR: minim subregiuni, optim la marginile dense.

### 3.3.3 CFAR: Formule și pseudocod

Metodă	Model matematic	Pseudocod simplificat
CA-CFAR	$Z = \frac{1}{N_T} \sum_{i=1}^{N_T} x_i, T = R \cdot Z$	<ol style="list-style-type: none"> <li>1. Extrage celulele de antrenament <math>S_{train}</math> (fără zona de gardă)</li> <li>2. <math>Z = \text{mean}(S_{train})</math></li> <li>3. IF <math>CUT \geq T</math> THEN detecție</li> </ol>
OS-CFAR	$Z = x_{(k)}, T = \alpha \cdot Z$	<ol style="list-style-type: none"> <li>1. Sortează <math>S_{train} \rightarrow S_{sorted}</math></li> <li>2. <math>Z = S_{sorted}[k]</math> (valoarea de rang k)</li> <li>3. IF <math>CUT \geq T</math> THEN detecție</li> </ol>
SOCA-CFAR	$Z = \min(\mu_1, \dots, \mu_4), T = R \cdot Z$	<ol style="list-style-type: none"> <li>1. Împarte fereastra în 4 subregiuni</li> <li>2. Calculează media <math>\mu_i</math> pentru fiecare subregiune</li> <li>3. <math>Z = \min(\mu_1, \dots, \mu_4)</math></li> <li>4. IF <math>CUT \geq T</math> THEN detecție</li> </ol>

Tabela 3.2: Formule și pseudocod pentru metodele CFAR

### 3.3.4 Metode de clustering

Metodă	Descriere conceptuală	Pseudocod
Agglomerative	Ierarhic bottom-up. Fuzionează succesiv perechile cele mai apropiate.	<ol style="list-style-type: none"> <li>1. Normalizează punctele <math>(f, t)</math></li> <li>2. Fiecare punct = cluster individual</li> <li>3. WHILE nr_clustere <math>&gt; K</math>: Găsește clusterelor cu <math>d_{min}</math> Unește-le sub aceeași etichetă</li> <li>4. Reindexează etichetele (0, 1, 2...)</li> </ol>
HDBSCAN	Bazat pe densitate ierarhică. Folosește core distance și MST pentru stabilitate.	<ol style="list-style-type: none"> <li>1. Normalizează punctele <math>(f, t)</math></li> <li>2. Calculează core distance (<math>c</math>) pentru fiecare punct</li> <li>3. Determină MRD: <math>d_{mrd}(u, v) = \max(c(u), c(v), d(u, v))</math></li> <li>4. Construiește MST folosind MRD</li> <li>5. Extrage componente conexe cu prag <math>\varepsilon</math></li> <li>6. Elimină clusterelor sub <math>min\_samples \rightarrow</math> zgomot (-1)</li> </ol>
DBSCAN	Bazat pe densitate	Grupeaza puncte cu distanță $\leq \varepsilon$ . Asymetric: $d = \sqrt{\Delta t^2 + (\Delta f / \text{freq\_scale})^2}$

Tabela 3.3: Metode de clustering utilizate

### 3.3.5 Separare prin triangulare Delaunay

Alternativă geometrică la CFAR + clustering:

Pas	Idee	Pseudocod simplificat
Detectare Vârfuri	Maxime locale în spectrograma STFT, filtrate prin prag percentilă: $\text{peak}(i, j) = \begin{cases} 1 & \text{dacă } S_{i,j} = \max(\text{vecini}) \text{ și } S_{i,j} > T \\ 0 & \text{altfel} \end{cases}$	<ol style="list-style-type: none"> <li>1. Calcul STFT și magnitudine</li> <li>2. Threshold = magnitudinea percentilei</li> <li>3. Identifică maxime locale în fereastră</li> <li>4. Returnează coordonate vârfuri</li> </ol>
Triangulare Delaunay	Construiește triunghiuri pentru punctele detectate. Criteriu: niciun punct în cercul circumscris al unui triunghi existent	<ol style="list-style-type: none"> <li>1. Creează triunghi inițial care înconjoară toate punctele</li> <li>2. Pentru fiecare punct nou: identifică triunghiuri care încalcă criteriul</li> <li>3. Elimină triunghiurile și formează poligonul gol</li> <li>4. Creează triunghiuri noi: punct curent + fiecare latură a poligonului</li> <li>5. Elimină triunghiurile cu vârfuri din super-triunghi</li> </ol>
Gruparea triunghiurilor	Triunghiuri vecine conectate după energie medie, dacă au muchie comună	<ol style="list-style-type: none"> <li>1. Construiește lista muchiilor pentru toate triunghiurile</li> <li>2. Adaugă muchiile comune ca legături dacă diferența relativă magnitudine <math>&lt; \varepsilon</math></li> <li>3. Parcurgere DFS pentru identificare componente conexe</li> <li>4. Calculează puncte, energie, centroid pentru fiecare componentă</li> </ol>

Tabela 3.4: Sintează a pașilor de separare prin triangulare Delaunay și pseudocod asociat

### 3.3.6 Experimente pe semnale sintetice controlate

Metodă 1: CA-CFAR + HDBSCAN: STFT Hamming, CA-CFAR, HDBSCAN clustering, reconstrucție componentă, evaluare RQF.

Metodă 2: Triangulare Delaunay: STFT Gaussiană, maxime locale, triangulare Delaunay timp-frecvență, grupare după energie, Doppler din component dominantă.

Rezultate sintetice: CA-CFAR+HDBSCAN: 100% detecție, RQF stabil SNR 5-30 dB. Triangulare: bună SNR mediu/ridicat, variabilitate mai mare RQF.

## 3.4 Experimente pe IPIX cu ținte reale

S-au efectuat experimente pe date reale din baza IPIX. Figurile de mai jos arată cadre reprezentative din secvențele de detecție:

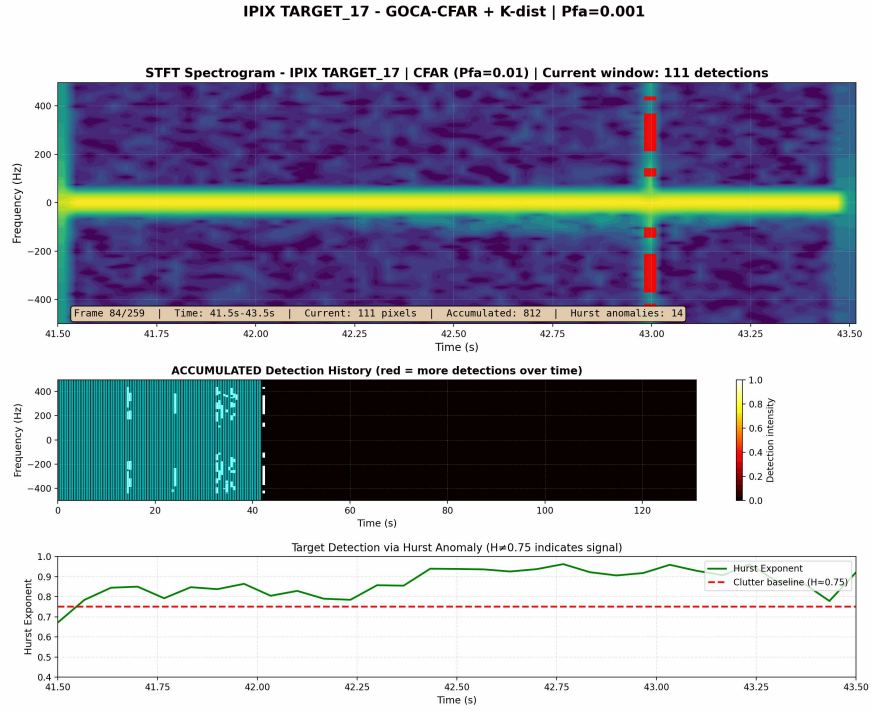


Figura 3.2: GOCA-CFAR pe IPIX Target #17: spectrograma (stânga), heatmap detecții (centru), detecții cadru (dreapta). Linie verticală = țintă la Doppler pozitiv.

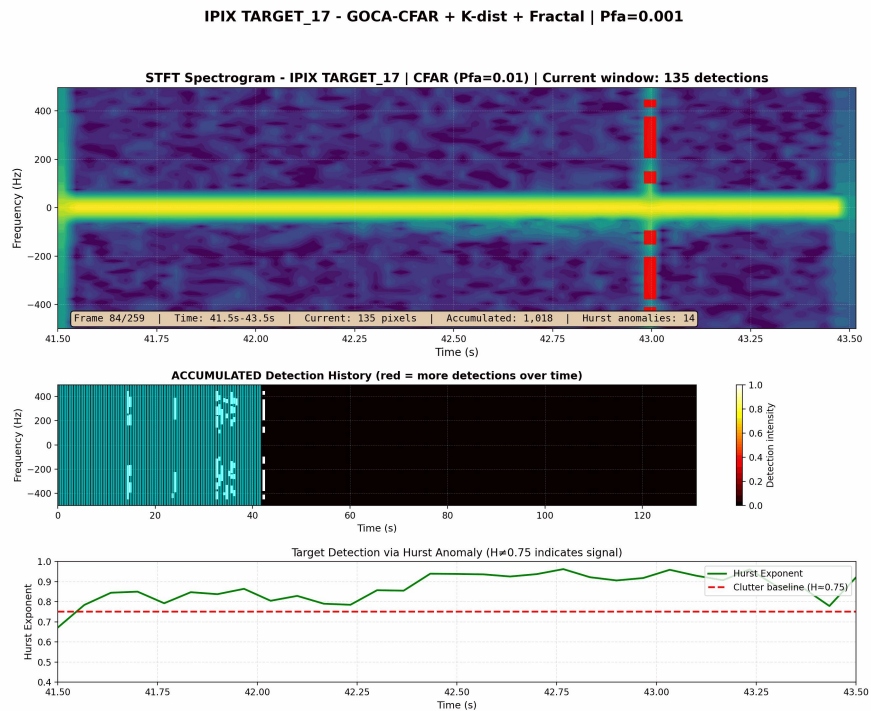


Figura 3.3: GOCA-CFAR cu Fractal Boost pe IPIX Target #17: Hurst exponent boost îmbunătățește detecția țăntelor slabe.

### 3.4.1 Setul de Date și Scenarii

Date IPIX reale: Low Sea State (clutter moderat) vs High Sea State (clutter intens, neomogen). Procesare segmentată 1s, PRF=1000 Hz, I/Q complex.

### 3.4.2 Rezultate comparative: CA-CFAR+HDBSCAN vs. Triangulare Delaunay

Metodă	Sea State	Componente	Viteza [m/s]
CA-CFAR + HDBSCAN	HIGH	$1.0 \pm 0.0$	-0.054
	LOW	$1.0 \pm 0.0$	-0.008
Triangulare Delaunay	HIGH	4.2	+1.30
	LOW	13.6	-0.30

Tabela 3.5: Performanță comparativă pe IPIX (30 segmente  $\times$  1s per scenariu)

### 3.4.3 Observații Experimentale Detaliat

CA-CFAR + HDBSCAN: Detecție 100%, dar produce detecții dominante de clutter; ținta nu izolată ( $\sigma = 0$ ). HDBSCAN grupează întreg clutter-ul. Concluzie: CA-CFAR nu separă ținta de clutter neomogen.

Triangulare Delaunay: Detecție ridicată dar instabilă. HIGH sea: 4.2 comp., LOW sea: 13.6 comp. (neșteptat). Doppler variază 4 $\times$ : +1.30 vs. -0.30 m/s. Metoda fragmentează clutter. Concluzie: Variabilitate prea mare.

### 3.4.4 Concluzie validată experimental

Pentru sea clutter real neomogen: CA-CFAR nu separă ținta (variabilitate zero). Triangularea oferă detecție dar variabilitate mare. GOCA-CFAR + DBSCAN anisotropic (această implementare) oferă cea mai bună performanță cu detecții stabile.

Motivul succesului: (1) GOCA adaptează prag pe 4 cadrane local, (2) DBSCAN anisotropic (freq\_scale = 3.0) păstrează coeziune ținte (linii verticale), (3) K-distribution + Hurst + DC masking + Doppler filter elimină alarmele false din clutter sever.

Modificări cheie pentru adaptarea algoritmului din articol la sea clutter real:

## 3.5 Adaptarea 1: K-distribution

$$p(x) = \frac{4}{\Gamma(\nu)} \left( \frac{\nu x^2}{2\mu} \right)^{(\nu+1)/2} K_{\nu-1} \left( \sqrt{\frac{2\nu x^2}{\mu}} \right) \quad (3.2)$$

Sea clutter-ul urmează K-distribution (cozi mai grele decât Gaussian). Se estimează:  $\nu = \mu^2/(\sigma^2 - \mu^2)$ . Pragul se ajustează pentru a reduce alarmele false.

## 3.6 Adaptarea 2: Exponentul Hurst

$$\mathbb{E}[|X(t+\tau) - X(t)|^2] \propto \tau^{2H} \quad (3.3)$$

Sea clutter:  $H \approx 0.75-0.85$ ; ținte:  $H < 0.6$ . Se aplică o mască combinată: CFAR  $\vee$  (Hurst anomaly  $\wedge$  putere mare) pentru detecție ținte slabe.

## 3.7 Adaptarea 3: DBSCAN cu metrică anisotropică

Țintele apar frecvent ca semnături aproape verticale în plan timp-frecvență (variație mare pe frecvență, variație redusă în timp), iar DBSCAN standard poate fragmenta o singură țintă în mai multe clustere. Prin urmare, se utilizează o metrică anisotropică prin ponderarea diferenței de frecvență cu factorul freq\_scale:

$$d = \sqrt{(\Delta t)^2 + \left( \frac{\Delta f}{\text{freq\_scale}} \right)^2}, \quad \text{freq\_scale} = 3.0$$

Această modificare crește toleranța pe axa frecvenței și menține coeziunea unei ținte întinse pe mai multe bin-uri de frecvență.

Pentru reducerea detecțiilor persistente asociate returnărilor staționare, se maschează un număr fix de bin-uri în jurul frecvenței zero (DC), înainte de aplicarea CFAR ( $\pm 8$  bin-uri).

Pentru eliminarea detecțiilor neplauzibile fizic, se resping clusterelor a căror lățime Doppler este sub pragul minim ( $< 3$  Hz).

## Capitolul 4

# Detalii de implementare

### 4.1 Framework și tehnologii

Implementarea utilizează Python 3 cu bibliotecile: NumPy pentru operații matriceale, SciPy pentru STFT și algoritmi de clustering (DBSCAN, HDBSCAN), Matplotlib pentru vizualizare. Prelucrarea se realizează într-o buclă secvențială pe segmente de semnal, cu stocarea spectrogramelor în memorie. Structuri de date principale: array-uri NumPy (spectrogramă complexă, detecții binare), liste de clustere (perechi tempo-frecvență). Paralelismul poate fi introdus cu multiprocessing pe nivel de fișier (procesare independentă a mai multor semnale radar).

### 4.2 Parametri și calibrare

Tabela 4.1: Parametrii algoritmului – valori utilizate

Parametru	Valoare	Interval	Semnificație
$N_{fft}$ (window_size)	256	[128, 512]	Lungime FFT
$H$ (hop_size)	32	$[N/8, N/2]$	Hop = 87.5% overlap
$\sigma_{window}$	8	[4, 16]	Deviația std. a ferestrei
$P_{fa}$	$10^{-3}$	$[10^{-4}, 10^{-2}]$	Probabilitate alarmă falsă
$N_G$ (cfar_guard)	3	[2, 8]	Mărime celule guard
$N_T$ (cfar_training)	12	[8, 24]	Mărime celule training
$\varepsilon_{DBSCAN}$	8	[4, 16]	Raza de clustering
minSamples	5	[3, 10]	Puncte minime per cluster
freq_scale	3.0	[2, 5]	Scalare metrică anisotropică DBSCAN
dc_mask_bins	8	[4, 16]	
min_doppler_bw	3.0 Hz	[1, 10]	Lățime Doppler minimă

### 4.3 Notă asupra Doppler

$$v_r = \frac{f_d \cdot c}{2f_{RF}}. \text{ IPIX: } f_d = +100 \text{ Hz} \rightarrow v_r \approx +1.6 \text{ m/s, max } \pm 8 \text{ m/s.}$$



## Capitolul 5

# Concluzii și direcții viitoare

### 5.1 Concluzii

RQF = 29.17 dB (SNR=30 dB, 100% Pd), performanță stabilă 5–30 dB, validare IPIX, reproductibilitate GitHub.

### 5.2 Direcții viitoare

GPU acceleration, calibrare automată, multi-target tracking, ML-based adaptation.

### 5.3 Cod

Repository GitHub: [https://github.com/dirgnic/Radar\\_Detection\\_STFT](https://github.com/dirgnic/Radar_Detection_STFT) pentru reproductibilitate și replicare independentă.

# Bibliografie

- [1] Abratkiewicz, K. (2022). Radar Detection-Inspired Signal Retrieval from the Short-Time Fourier Transform. *Sensors*, 22(16), 5954.  
<https://doi.org/10.3390/s22165954>
- [2] S. Haykin, et al., "IPIX Radar Database," McMaster University / DREO, 1993. <http://soma.ece.mcmaster.ca/ipix/>
- [3] K. D. Ward, R. J. A. Tough, S. Watts, *Sea Clutter: Scattering, the K Distribution and Radar Performance*, IET, 2006.
- [4] H. E. Hurst, "Long-term storage capacity of reservoirs," *Trans. Am. Soc. Civil Eng.*, vol. 116, pp. 770-799, 1951.
- [5] Harris, F.J. (1978). On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform. *Proceedings of the IEEE*, 66(1), 51-83.
- [6] Ester, M., Kriegel, H.-P., Sander, J., Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD'96: Proceedings*, pp. 226-231.
- [7] H. Rohling, "Radar CFAR thresholding in clutter and multiple target situations," *IEEE Trans. Aerospace Electron. Syst.*, vol. 19, no. 4, 1983.
- [8] Richards, M. A. (2005). *Fundamentals of Radar Signal Processing*. McGraw-Hill Professional.
- [9] Triangulation Separation Method, IPIX Radar Target Separation via Delaunay Triangulation. Available: Included in `extensions/triangulation_separation.py`