Change Detection Algorithm with Hardware Constraints

Prof. Ingo Sander[1], Postdoc Marcello Costa[2]
September 16, 2024 — [1]KTH Royal Institute of Technology, [2]Cisb-Saab

Contents

1. **Elementary bitemporal Change Detection:** → rapid data analysis, low complexity, power consumption, and memory requirements, parallelized extension[1]
2. **Time-series methods**[3] → ARIMA, PCA, SVM, Random Forest, NN, K-means, GLM, EM, SVD, XGboost, LSTM,..
3. **Synchronous MoC and Parallelism** → Distributed data processing architectures for runtime, memory and energy saving[2]
4. **Particular case: Ultra-wideband (UWB) SAR** → stable scatters in time, random motions of targets in the clutter as a first-order autoregressive AR(1) process

---

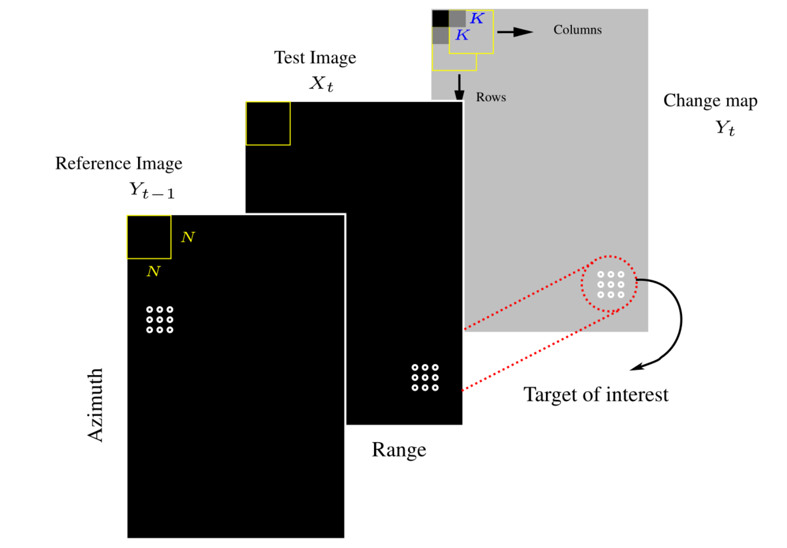
[1] Wu, Suya, et al. "Quickest Change Detection for Unnormalized Statistical Models," *IEEE Trans. on Inf. Theory*, v. 70, n. 2, pp. 1220-1232, 2024.

[2] Rabhi *et al.*, "Patterns and skeletons for parallel and distributed computing," Springer Science & Business Media, 2003.

[3] Bayer, F.M., *et al.*, "A Novel Rayleigh Dynamical Model for Remote Sensing Data Interpretation," *IEEE Trans. on Geoscience and Remote Sensing*, v. 58, n. 7, pp. 4989-4999, 2020.

Time series Prediction Model

$$\hat{Y}_t = f(X_t, Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p}) + \epsilon_t, \quad (1)$$



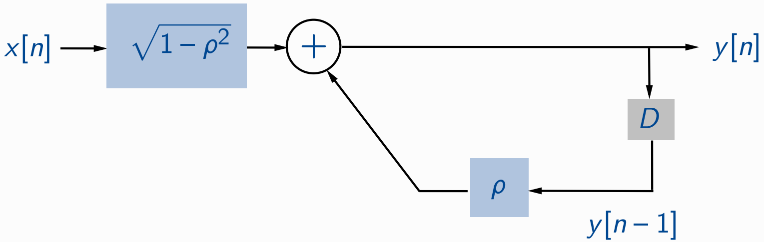
Combinational solution

X_t	Y_{t-1}	Change
0	0	No change
0	1	No change
1	0	Change
1	1	No change

Note: Anomaly = 1, background = 0.

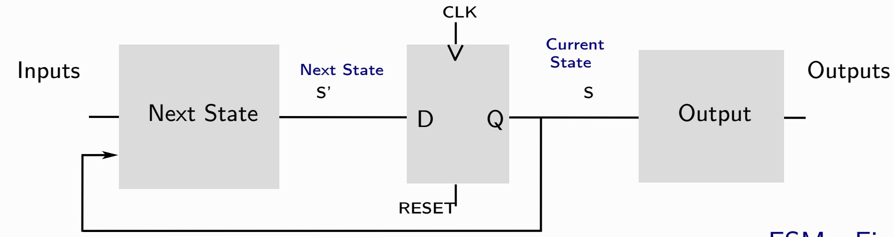
Auto-regressive first order Model (AR(1)): Detects uncorrelated samples between y_{t-1} and x_t as a predicted anomaly to compose the change map y_t, which describes the degree of retained memory of previous states.

$$\underbrace{\hat{y}_t}_{\text{changes}} = \underbrace{\rho}_{\text{corr.}} \underbrace{y_{t-1}}_{\text{ref. signal}} + \sqrt{1-\rho^2} \underbrace{x_t}_{\text{test signal}}, \tag{2}$$

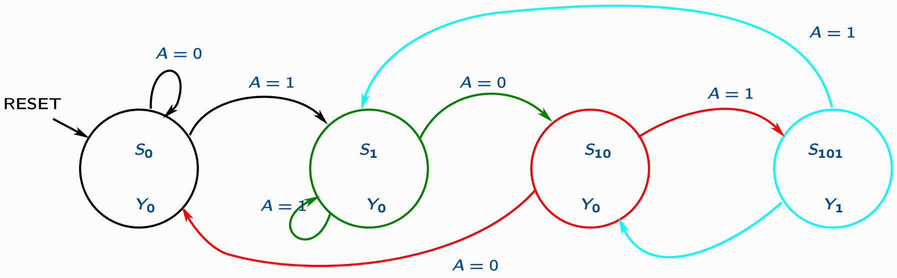


where $\rho \to [0, 1]$.

Synchronous Model of Computation (MoC)



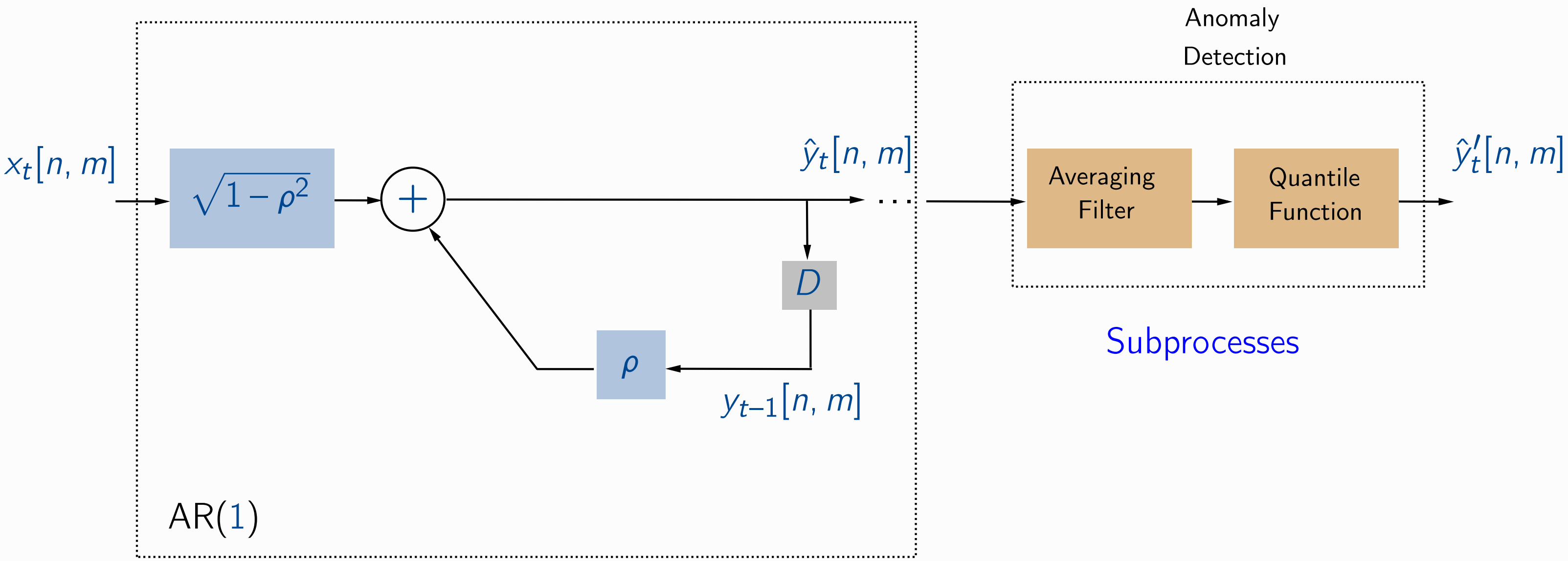
FSM - Finite State Machine



Structure for a first order IIR filter.

Change Detection Algorithm: 2D-AR(1) on Synchronous MoC

Time-Spatial Implementation:



Synchronous
Model of Computation

Subprocesses

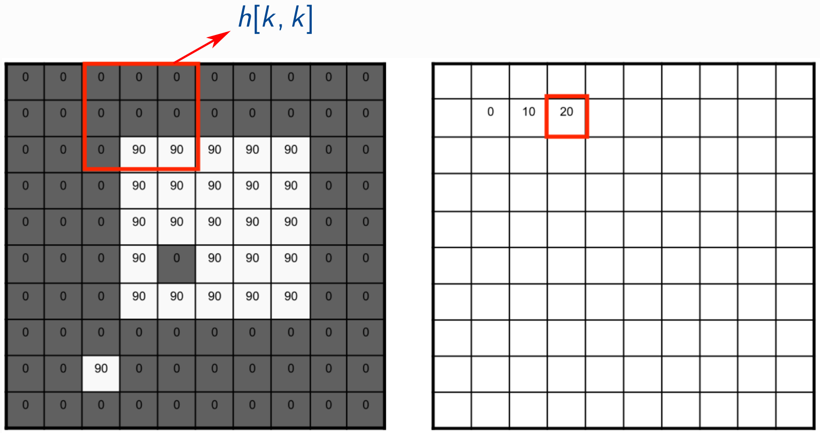
Spatial Filtering with Moving Average (FIR)

The smoothed function $g[n, m]$ under the neighborhood influence is determined by the kernel elements h through the original signal $f[n, m]$, i.e.,

$$\begin{aligned} g[n, m] &= (f * h)[n, m] \\ &\sum_{k,\ell} f[k, \ell]h[n-k, m-\ell] \end{aligned} \tag{3}$$

For a generic stencil kernel

$$h = \frac{1}{k}\begin{bmatrix} 1 & 1 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & k \end{bmatrix}_{k\times k} \tag{4}$$



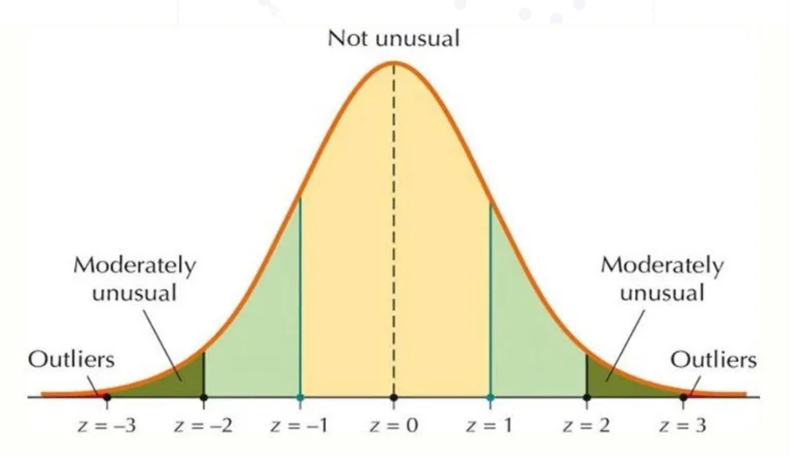
$f[n, m] = \hat{y}_t[n, m]$ $\qquad$ $g[n, m] = \hat{y}'_t[n, m]$

Anomaly Detection

The empirical cumulative distribution function (ECDF), $\hat{F}(\cdot)$, is obtained from $[\hat{y}'_1, \ldots, \hat{y}'_n]$ i.i.d samples of $\hat{y}'$, making

$$\hat{F}(\hat{y}') = \frac{1}{n}\sum_{i=1}^{n} I(\hat{y}'_i \leq \hat{y}') \qquad (5)$$

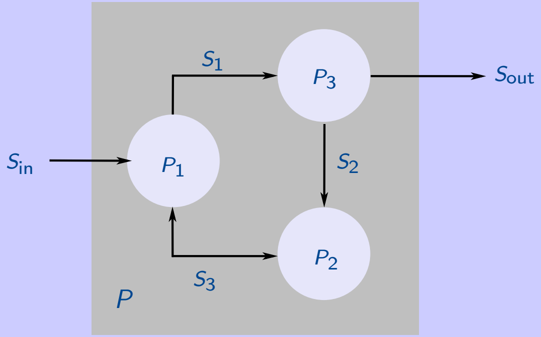
$$\hat{y}'(p) = \hat{F}^{-1}(p),$$



$\Rightarrow$ $\hat{y}'(p)$ establishes the threshold on $\hat{y}'_t[n, m]$ when $x_t[n, m]$ has an anomaly of interest (potential target) against the lower clutter anomalies due to the state $y_{t-1}[n, m]$

Synchronous MoC[5]**:** Synchronized parallel components, running in successive computation steps, where all components perform some quantum of computation.

1. Output is synchronous with input
2. Internal actions are instantaneous



[5] Albert Benveniste and Gérard Berry. "The synchronous approach to reactive and real-time systems," *Proceedings of the IEEE*, 79(9):1270–1282, September 1991.

Change Detection with AR Model

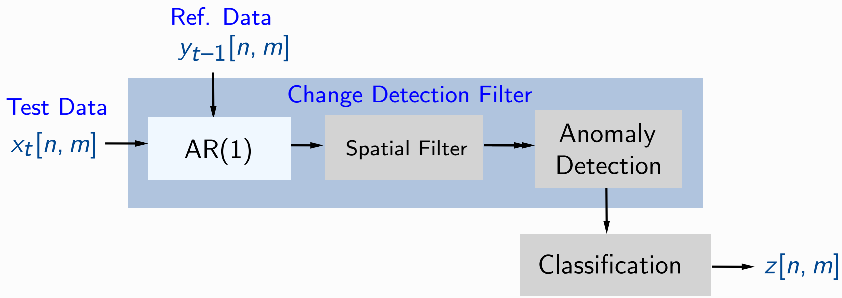
▶ ForSyDe Framework[5]/Haskell

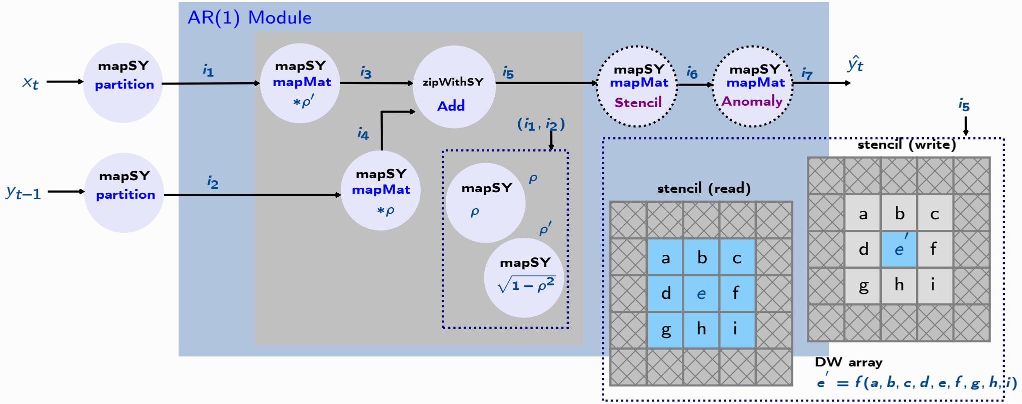
Synchronous MoC constructors:

- Combinational → `zipWithSY`, `mapSY`
- Delay → `delaySY`
- Sequential → `mooreSY`, `mealySY`

Parallel stencil computation

- Data.Massiv.ArrayFramework[6]



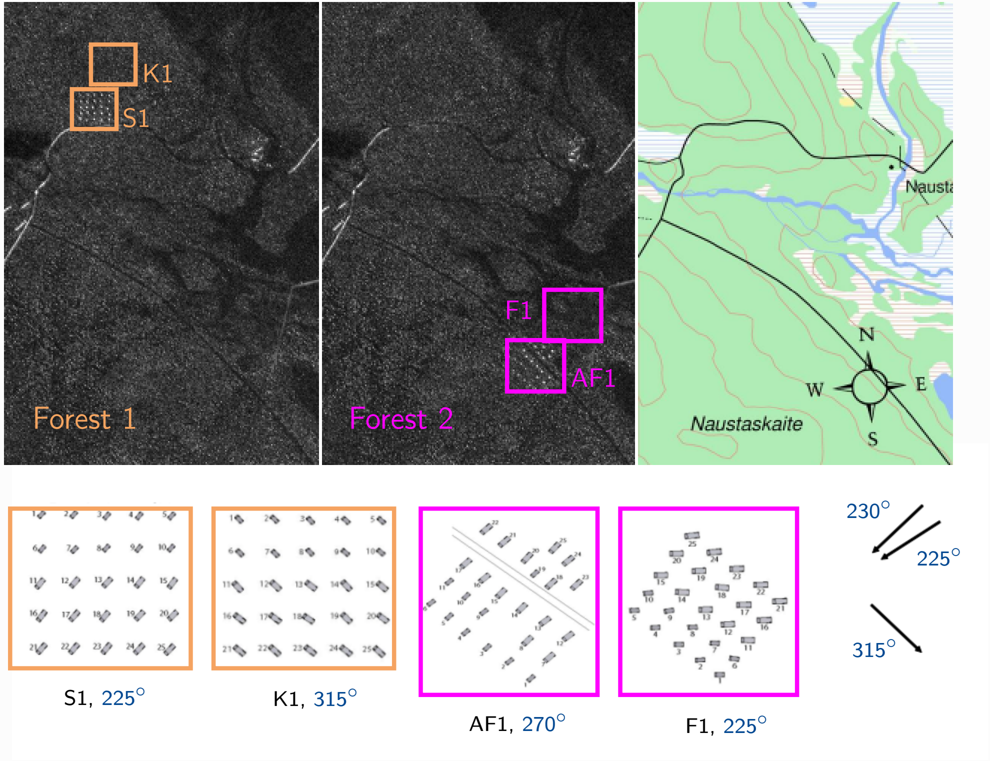


[5] https://forsyde.github.io/
[6] https://hackage.haskell.org/package/massiv

VHF-UWB SAR Dataset: CARABAS-II

CARABAS-II dataset[7]:

- 24 images: magnitude VHF ($20-90$ MHz) SAR HH-polarized with 1 m resolution (2×3 km^2)
- 4 Targets position: S1, K1, F1, and AF1
- 6 passes: 3 Flight directions under ground RFI sources ON/OFF
- Application: Through-foliage detection

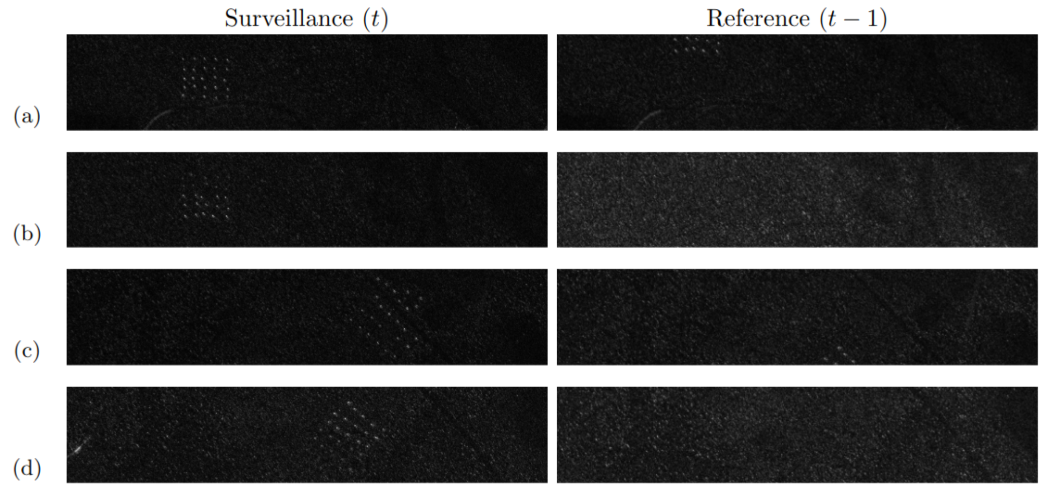


[7] Ulander, L.M, et al. “Change detection for low-frequency sar ground surveillance,” *IEE Proceedings-Radar, Sonar and Navigation* , 152(6), pp. 413–420, 2005.

VHF-UWB SAR Dataset: CARABAS-II

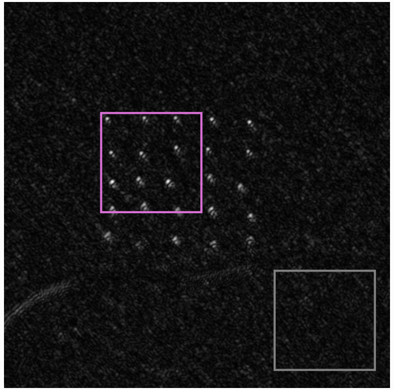
	Exp. No.	Surveillance image		Reference image		Known Targets	Area [km^2]
		Mission	Pass	Mission	Pass		
(a)	1	2	1	3	1	25	6
(b)	2	3	1	4	1	25	6
(c)	3	4	1	5	1	25	6
(d)	4	5	1	2	1	25	6
	5	2	2	4	2	25	6
	6	3	2	5	2	25	6
	7	4	2	2	2	25	6
	8	5	2	3	2	25	6
	9	2	3	5	3	25	6
	10	3	3	2	3	25	6
	11	4	3	3	3	25	6
	12	5	3	4	3	25	6
	13	2	4	3	4	25	6
	14	3	4	4	4	25	6
	15	4	4	5	4	25	6
	16	5	4	2	4	25	6
	17	2	5	4	5	25	6
	18	3	5	5	5	25	6
	19	4	5	2	5	25	6
	20	5	5	3	5	25	6
	21	2	6	5	6	25	6
	22	3	6	2	6	25	6
	23	4	6	3	6	25	6
	24	5	6	4	6	25	6
	Total					600	144

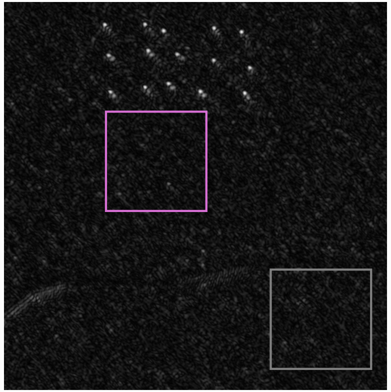
Campaign sample data: 2×0.5 km^2 (15%)



Step 1: Block size selection

surveillance x_t reference y_{t-1}

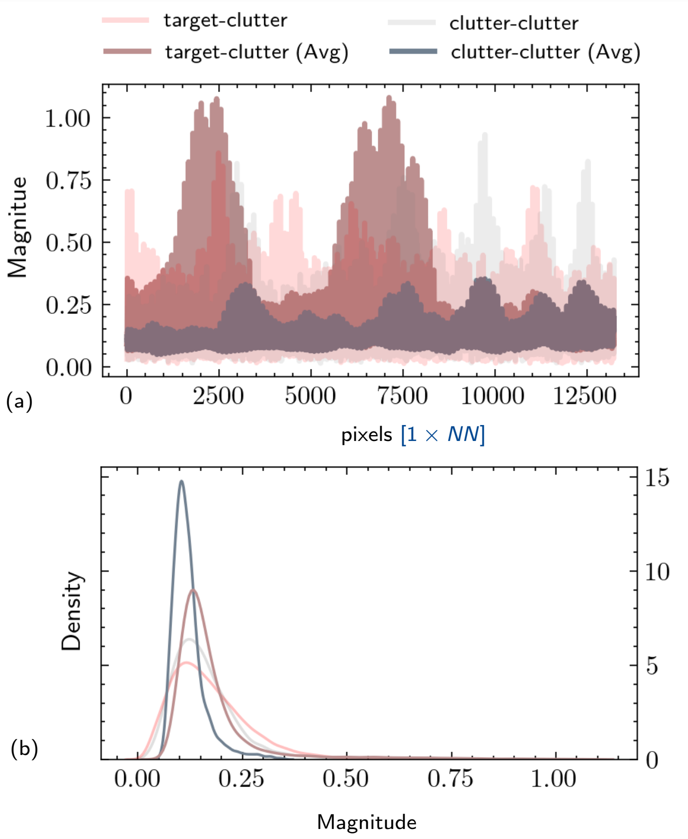




sample 115×115 : target-clutter

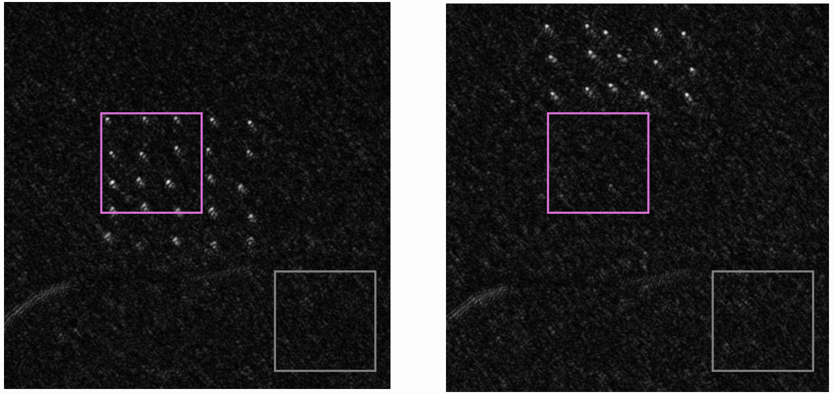
sample 115×115 : clutter-clutter

AR output / Estimated pdf



Step 1: Block size selection

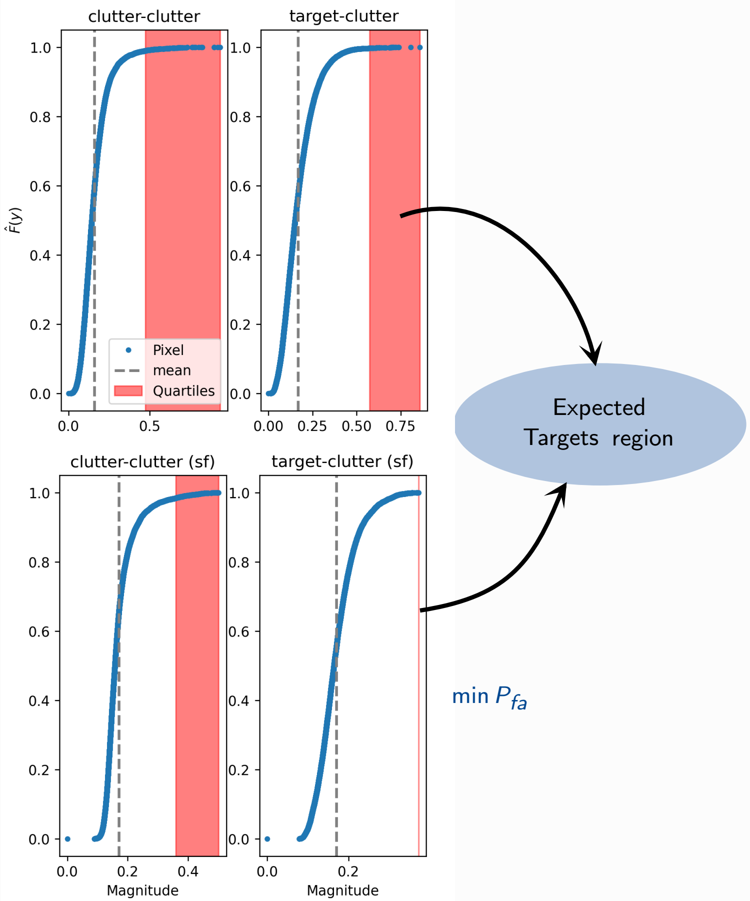
surveillance x_t reference y_{t-1}



sample 115×115 : target-clutter

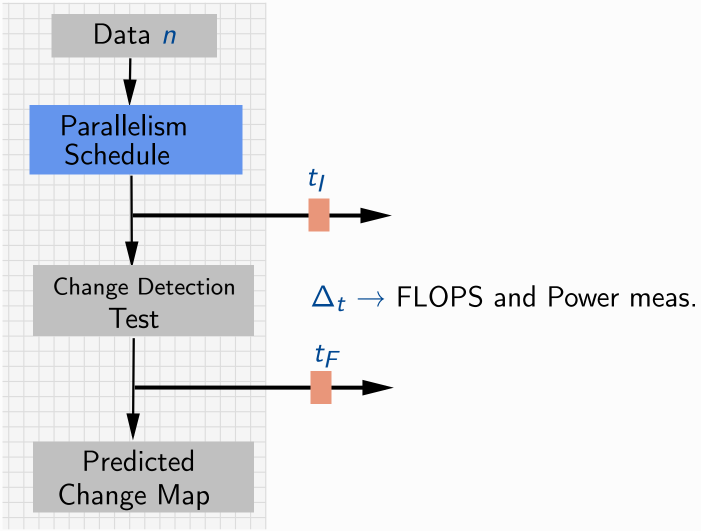
sample 115×115 : clutter-clutter

q-ECDF



Step 2: HW measurements

- Scenarios: Sample dataset 15%
- Tests: [AR(1), GLRT*]
- Analytical block N: [40,40]
- Parallelism scheduling: [4,8] cores
- Convolution block k: [9,9].
- Linux-based perfs[9]: `htop`, `powerstat`



*GLRT → generalized likelihood-ratio test for Gaussian Distribution[8].

[8] M. Lundberg, , et al., "A challenge problem for detection of targets in foliage," in Algorithms for Synthetic Aperture Radar Imagery XIII, p. 62370K, May 2006.

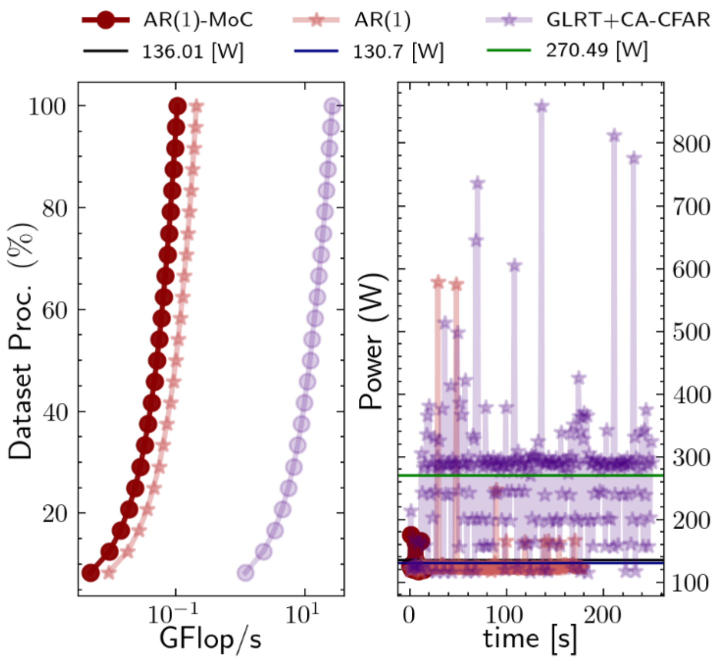
[9] The CPU processor Intel Core i9-12900KF, 16-Core, 24-threads, clock of 3.2 GHz and 32 GB DDR5 of memory.

Energy-Complexity Performance

	GLRT		2D-AR(1)	
Instance	GLRT	CFAR	sf	q-ECDF
Complexity	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2K^2)$	$\mathcal{O}(N^2k^2)$	$\mathcal{O}(N^2 \log N^2)$
Runtime (sec)	8.47	242.60	10.65	7.00

Note: sf = spatial filtering stage with averaging filter.

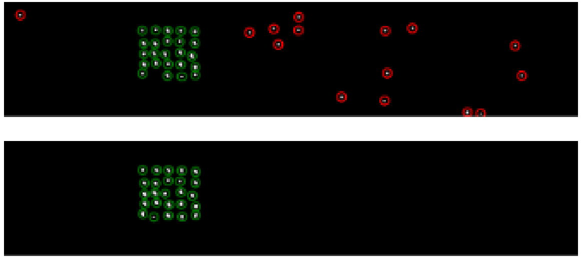
$\Rightarrow$ GLRT demands 65% additional power consumption by tested pair.



VHF-UWB SAR Dataset: CARABAS-II

$2 \times 0.5\ \text{km}^2$ (15%)

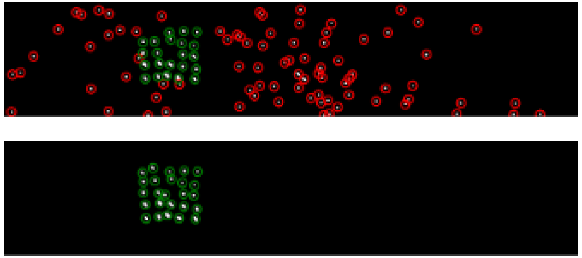
Forest 1 Pass 1 Mission S1



GLRT + CA-CFAR: 24, 15, 1

2D-AR(1): 25, 0, 0

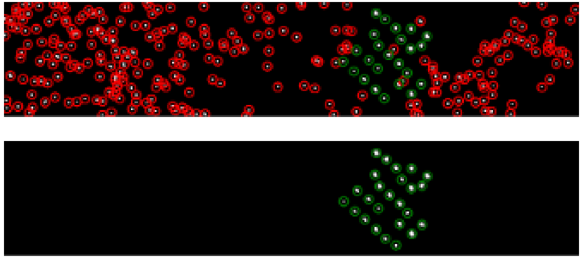
Forest 1 Pass 1 Mission K1



GLRT + CA-CFAR: 21, 84, 4

2D-AR(1): 25, 0, 0

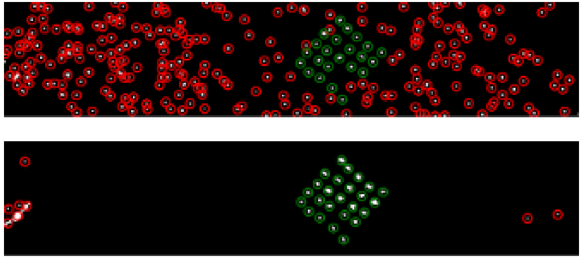
Forest 2 Pass 1 Mission F1



GLRT + CA-CFAR: 24, 325, 1

2D-AR(1): 25, 0, 0

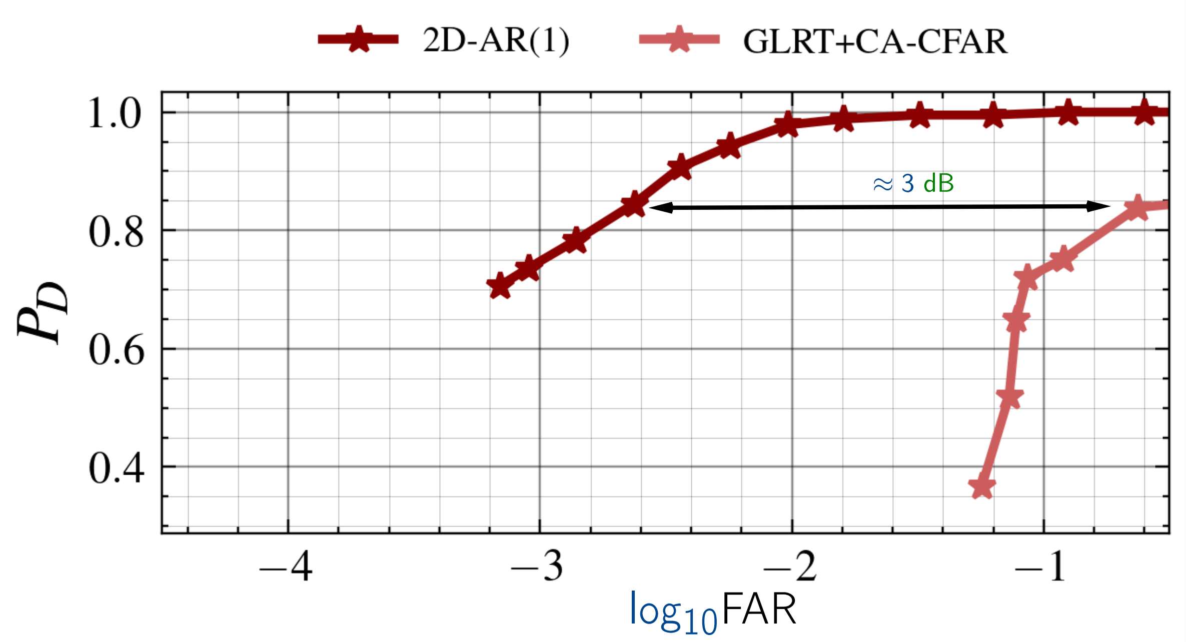
Forest 2 Pass 1 Mission AF1



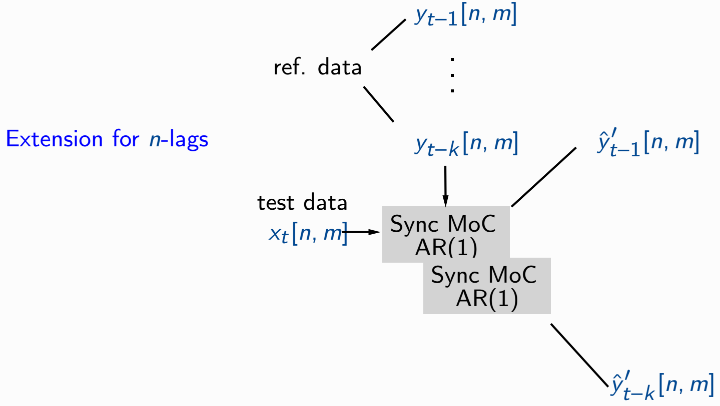
GLRT + CA-CFAR: 23, 345, 2

2D-AR(1): 25, 8, 0

Step 3 - ROC over $24[2 \times 3\ \text{km}^2\ (100\%)]$



- Hardware efforts $\times$ detection performance are balanced over high-parallelized Sync MoC structure by $[N, k]$ selection, enabling fast data analysis.
- Extension for a Markov chain in multitemporal (n-lags) improves the detection performance (considering relevant statistical dependencies), replicating the elementary MoC structure.

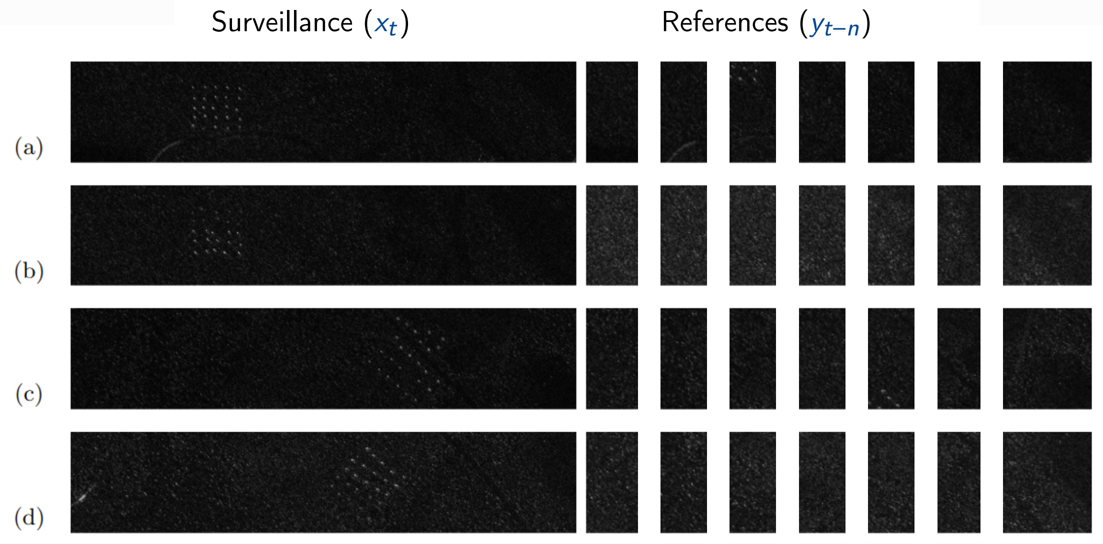


Extension for n-lags

Exp. No.	Surveillance image		Reference image		Known Targets	Area [km^2]
	Mission	Pass	Mission	Pass		
1	2	1	3	1	25	6
2	3	1	4	1	25	6
3	4	1	5	1	25	6
4	5	1	2	1	25	6
5	2	2	4	2	25	6
6	3	2	5	2	25	6
7	4	2	2	2	25	6
8	5	2	3	2	25	6
9	2	3	5	3	25	6
10	3	3	2	3	25	6
11	4	3	3	3	25	6
12	5	3	4	3	25	6
13	2	4	3	4	25	6
14	3	4	4	4	25	6
15	4	4	5	4	25	6
16	5	4	2	4	25	6
17	2	5	4	5	25	6
18	3	5	5	5	25	6
19	4	5	2	5	25	6
20	5	5	3	5	25	6
21	2	6	5	6	25	6
22	3	6	2	6	25	6
23	4	6	3	6	25	6
24	5	6	4	6	25	6
Total					600	144

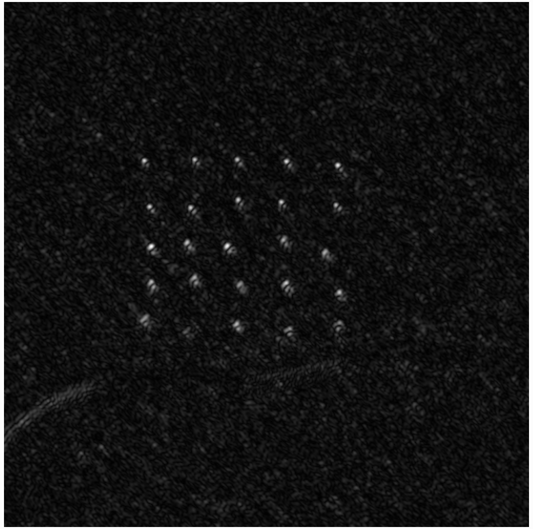
Include all lags y(t-n) to improve the prediction model

Campaign sample data: 2×0.5 km (15%)

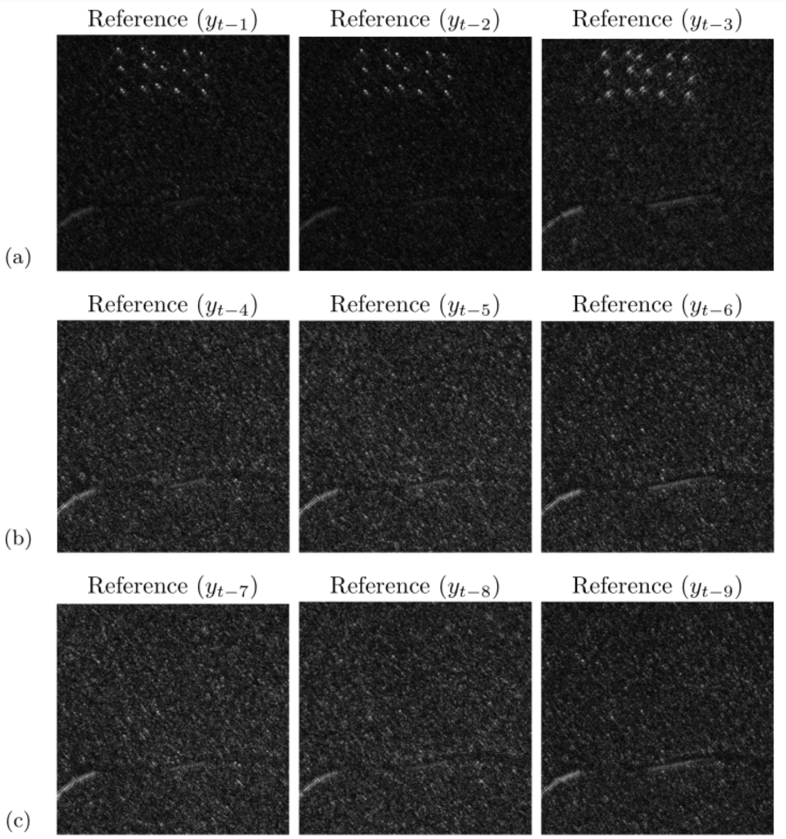


Extension for n-lags

Surveillance x_t

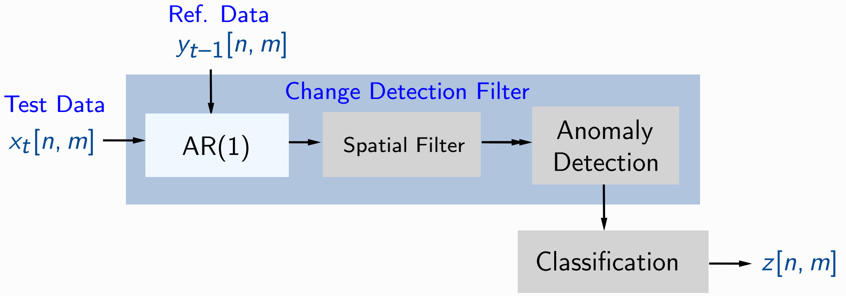


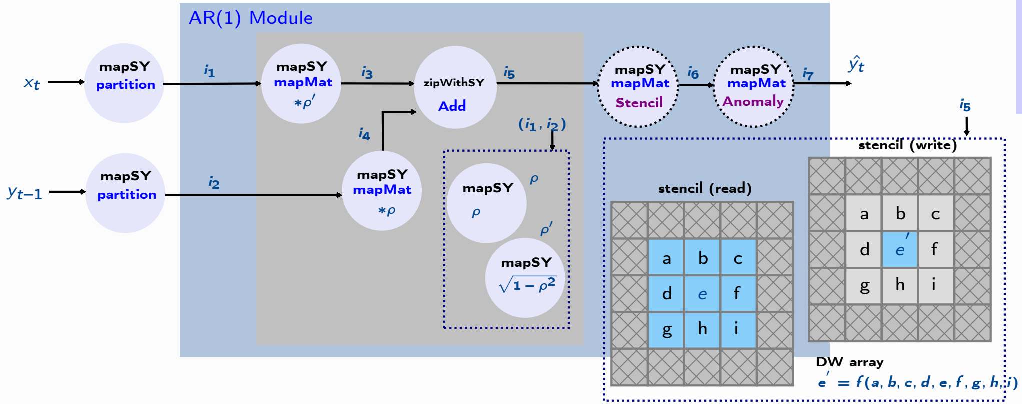
Statistical sample data: 0.5×0.5 km (4%)



Demo 2: 2D-AR(1) processing on ForSyde/Haskell

Change Detection with AR Model



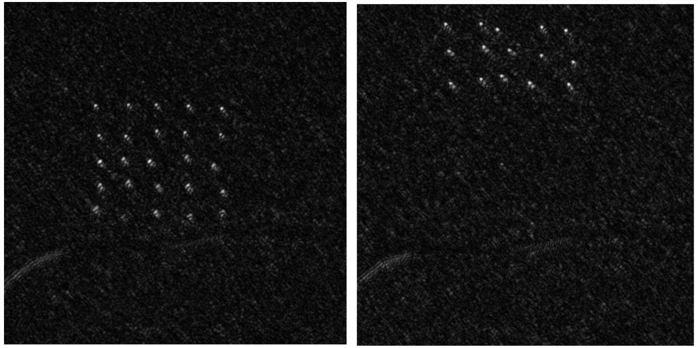


▶ Files

- changeDetection.hs
- classifier.py
- Itesti.dat (500×500) (i=Exp.No.)
- Irefin.dat (500×500) (n=lags)
- .mat (known targets position)

Demo 2: 2D-AR(1) processing on ForSyde/Haskell

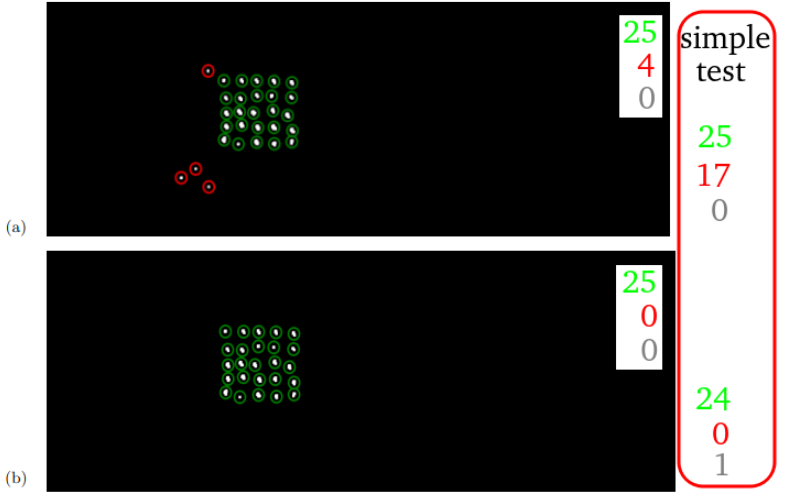
Surveillance (t) and Reference $(t-1)$



Objectives:

- Extend to n-lags with parallelism
- HW/Detection optimization

Classification Results:



(a) threshold $= 0.5$, (b) threshold $= 0.6$

KTH

VETENSKAP
OCH KONST