Doppler Map

1. Doppler Map (Harta Range-Doppler)
Ce este:
Harta Range-Doppler este o reprezentare bidimensională care arată cum răspunde radarul în funcție de distanța (range) și viteza relativă (Doppler) a obiectelor detectate. Aceasta este folosită pentru a localiza și măsura viteza țintelor mobile.

Cum se generează:
Se aplică un FFT pe axa de distanță (range) – pentru fiecare sweep radar.

Apoi, se aplică un FFT pe axa de timp lent (slow time) – adică între sweep-uri – pentru a detecta schimbări de fază cauzate de mișcare (viteza).

Rezultatul este o matrice complexă în care fiecare celulă indică puterea semnalului la o combinație distanță-viteză.

Ce conține:
Axa X: viteza relativă (Doppler shift)

Axa Y: distanța de la radar

Valoarea pixelului: intensitatea semnalului reflectat

Cum e afectată:
Interferența poate introduce semnale fantomă în zone greșite ale hărții (ex. viteză falsă).

Zgomotul poate reduce contrastul, făcând țintele mai greu de detectat.

Obiectele nemișcate (ziduri, sol) apar pe linia de viteză zero (Doppler = 0).

Ce sunt cercurile:
Cercurile din Doppler map sunt artefacte vizuale cauzate de ferestrele de ponderare (ex. Hann) și de interferențe constructive/destructive în semnal.

Acestea nu reprezintă obiecte reale, ci efecte de sidelobes sau aliasing spectral (efectul FFT asupra semnalului de la o țintă puternică).

Angle Map

2. Angle Map (Harta Range-Angle)

Ce este:

Harta Range-Angle reprezintă distribuția semnalelor radar în funcție de distanță și unghi de sosire (AOA – Angle of Arrival). Este folosită pentru a determina direcția în care se află o țintă față de radar.

Cum se generează:

Se aplică FFT pe axa unghiulară (virtual array) – adică pe datele colectate de la mai multe antene (array).

Această transformare convertește semnalul în funcție de direcția din care a venit.

Pentru a obține unghiul, sistemul trebuie să fie MIMO sau să folosească o matrice de antene (virtual array), iar semnalele să fie coerente.

Ce conține:

Axa X: unghiul de sosire (în grade sau radiani)

Axa Y: distanța

Valoarea pixelului: puterea semnalului reflectat de obiecte din acea direcție

Cum e afectată:

Interferența poate masca unghiul real al țintei, dând impresia că un obiect se află într-o altă direcție.

Zgomotul poate reduce rezoluția unghiulară, făcând obiectele apropiate greu de separat.

O antenă cu prea puține elemente sau spațiere necorespunzătoare degradează performanța.

Ce sunt cercurile:

Cercurile din harta unghiulară (range-angle) sunt lobi secundari sau artefacte de FFT și apar când o țintă puternică „scapă” energie în alte direcții din cauza rezoluției unghiulare limitate.

Ele pot da impresia unor direcții multiple, dar doar una este corectă (vârful lobilor principali).

Hartă	Axa X	Axa Y	Ce vezi?	Ce sunt cercurile?
Doppler Map	Viteză (Doppler)	Distanță	Ținte mobile	Lobi laterali / interferență FFT
Angle Map	Unghi	Distanță	Direcția din care vine semnalul	Lobi secundari ai antenei virtuale

```
%%

% Inițializează pozițiile țintelor în cadrul de referință al vehiculului ego
tgtPoses = targetPoses(egoCar);

% Distanța și unghiul vehiculului țintă relativ la radarul victimă
[tarRange, tarAngle] = rangeangle(tgtPoses(1).Position');

% Viteza radială a vehiculului țintă relativ la radarul victimă (semnul negativ
% este din cauza axelor de referință diferite între radialspeed și drivingScenarioDesigner)
tarVelocity = -radialspeed(tgtPoses(1).Position', tgtPoses(1).Velocity');

% Distanța și unghiul vehiculului de interferență relativ la radarul victimă
[intRange, intAngle] = rangeangle(tgtPoses(2).Position');

% Viteza radială a vehiculului de interferență relativ la radarul victimă
intVelocity = -radialspeed(tgtPoses(2).Position', tgtPoses(2).Velocity');

%%

%Transceiver Signal Processing



%% Obtain Interference-free and Interfered Data Cubes
% Definește eșantioanele de timp rapid și timp lent
Nft = round(fmcwwav1.SweepTime*fmcwwav1.SampleRate);  % Numărul de eșantioane
% de timp rapid pentru radarul victimă
Nsweep = 192;                                % Numărul de eșantioane
% de timp lent pentru radarul victimă
Nift = round(fmcwwav2.SweepTime*fmcwwav2.SampleRate); % Numărul de eșantioane
% de timp rapid pentru radarul de interferență
Nisweep = ceil(Nft*Nsweep/Nift);                % Numărul de eșantioane
% de timp lent pentru radarul de interferență

% Generează codul TDM-MIMO pentru elementele antenei de interferență
wi = helperTDMMIMOEncoder(Nit, Nisweep);

% Asamblează trenul de pulsiuni de interferență la timpul de scenariul radarului
% de interferență
rxIntTrain = zeros(Nift*Nisweep, Nvr);

% Inițializează timpul de scenariul
time = scenario.SimulationTime;

% Inițializează profilele actorilor
actProf = actorProfiles(scenario);
```

```
%
    Obține trenul de pulsiuni de interferență la timpul de scenariul radarului
% de interferență
for l = 1:Nisweep
    % Generează toate drumurile către radarul victimă
    ipaths = helperGenerateIntPaths(tgtPoses, actProf, lambdaRdr2);

    % Obține semnalul primit doar cu interferență
    rxInt = iradar(ipaths, time, wi(:, l));
    rxIntTrain((l-1)*Nift+1:l*Nift, :) = rxInt;

    % Obține timpul curent al scenariului
    time = time + fmcwwav2.SweepTime;

    % Mută țintele înainte în timp pentru următorul sweep
    tgtPoses(1).Position = [tgtPoses(1).Position] + [tgtPoses(1).Velocity] * fmcwwav2.SweepTime;
    tgtPoses(2).Position = [tgtPoses(2).Position] + [tgtPoses(2).Velocity] * fmcwwav2.SweepTime;
end

% Trunchiază trenul de pulsiuni de interferență la lungimea trenului de pulsiuni
% de țintă
rxIntTrain = rxIntTrain(1:Nft*Nsweep, :);

% Asamblează trenul de pulsiuni de interferență la timpul de scenariul radarului
% victimă
rxIntvTrain = permute(reshape(rxIntTrain, Nft, Nsweep, Nvr), [1, 3, 2]);



%%

% Dechirpează semnalul țintei la radarul victimă pentru a obține cubul de date
% fără interferență,
% care este folosit pentru a trasa răspunsul țintei ca referință ideală de performanță.
% De asemenea, dechirpează semnalul combinat de țintă și interferență pentru a
% obține cubul de date cu interferență,
% care este folosit pentru a trasa efectele interferenței asupra detectării țintei.

% Generează codul TDM-MIMO pentru elementele de antenă ale radarului victimă
wv = helperTDMMIMOEncoder(Nvt, Nsweep);

% Asamblează cubul de date fără interferență
XcubeTgt = zeros(Nft, Nvr, Nsweep);

% Asamblează cubul de date cu interferență
Xcube = zeros(Nft, Nvr, Nsweep);
```

```
%
    Inițializează timpul de scenariul
time = scenario.SimulationTime;

% Inițializează pozițiile țintelor în cadrul de referință al vehiculului ego
tgtPoses = targetPoses(egoCar);

% Inițializează profilele actorilor
actProf = actorProfiles(scenario);

% Obține cubul de date la timpul de scenariul radarului victimă
for l = 1:Nsweep
    % Generează drumurile țintelor către radarul victimă
    vpaths = helperGenerateTgtPaths(tgtPoses, actProf, lambda);

    % Obține semnalul primit fără interferență
    rxVictim = vradar(vpaths, time, wv(:, l));

    % Dechirpează semnalul țintei fără interferență
    rxVsig = dechirp(rxVictim, sig);

    % Salvează sweep-ul în cubul de date
    XcubeTgt(:,:,l) = rxVsig;

    % Obține semnalul de interferență primit
    rxInt = rxIntvTrain(:,:,l);

    % Dechirpează semnalul cu interferență
    rx = dechirp(rxInt + rxVictim, sig);
    Xcube(:,:,l) = rx;

    % Obține timpul curent al scenariului
    time = time + fmcwwav1.SweepTime;

    % Mută țintele înainte în timp pentru următorul sweep
    tgtPoses(1).Position = [tgtPoses(1).Position] + [tgtPoses(1).Velocity] * fmcwwav1.SweepTime;
    tgtPoses(2).Position = [tgtPoses(2).Position] + [tgtPoses(2).Velocity] * fmcwwav1.SweepTime;
end


%%
% Calculate number of range samples
Nrange = 2^nextpow2(Nft);
```

```
% Define range response
rngresp = phased.RangeResponse('RangeMethod','FFT', ...
    'SweepSlope',fmcwwav1.SweepBandwidth/fmcwwav1.SweepTime, ...
    'RangeFFTLengthSource','Property','RangeFFTLength',Nrange, ...
    'RangeWindow','Hann','SampleRate',fmcwwav1.SampleRate);

% Calculate the range response of interference-free data cube
XrngTgt = rngresp(XcubeTgt);
%%
% Decode TDM-MIMO waveform
[XdecTgt,Nsweep] = helperTDMMIMODecoder(XrngTgt,wv);

% Number of Doppler samples
Ndoppler = 2^nextpow2(Nsweep);

% Size of virtual array
Nv = Nvr*Nvt;

% Doppler FFT with Hann window
XrngdopTgt = helperVelocityResp(XdecTgt,Nrange,Nv,Ndoppler);




%% dopler map

% Pulse repetition interval for TDM-MIMO radar
tpri = fmcwwav1.SweepTime*Nvt;

% Maximum unambiguous velocity
vmaxunambg = lambda/4/tpri;

% Plot range-Doppler map
helperRangeVelocityMapPlot(XrngdopTgt,rngresp.SweepSlope,...
    fmcwwav1.SampleRate,tpri,rangeMax,vmaxunambg,Nrange,Ndoppler,lambda);

%% angle map

% Number of angle samples
Nangle = 2^nextpow2(Nv);

% Angle FFT with Hann window
XrngangdopTgt = helperAngleResp(XrngdopTgt,Nrange,Nangle,Ndoppler);

% Target Doppler bin
tarDopplerBin = ceil(tarVelocity/lambda*2*tpri*Ndoppler+Ndoppler/2);

% Plot range-angle map
helperRangeAngleMapPlot(XrngangdopTgt,rngresp.SweepSlope,...
    fmcwwav1.SampleRate,rangeMax,vrxEleSpacing,Nrange,Nangle,tarDopplerBin,lambda);
```