Coder: Weston Yohe
Manager: James Blacklock
August 20, 2022

Medium Fidelity Radar Simulator

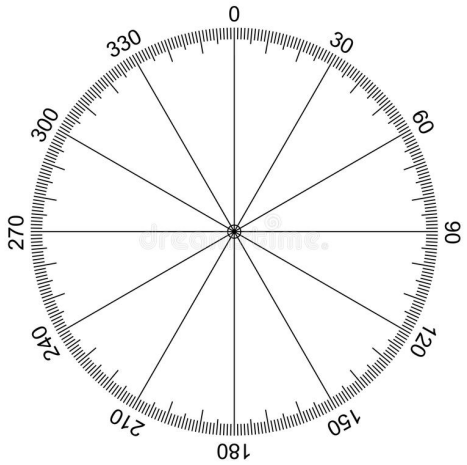
Introduction

This project was assigned/created as a solo summer internship project with the objective of creating a medium fidelity radar simulator which simulates a radar system consisting of multiple radar faces which possess a varied amount of unique search sectors per radar face. The ultimate objective of this program is to give user's an accurate, but quick, reference to search, detect and track data from a user-defined radar system in a user-defined simulation environment. While the program is configurable and functional, it is still a work in progress. Further work must be done to provide higher fidelity. Currently, the program inputs specifically formatted text files consisting of the user-defined parameters of the simulation environment and radar system. Upon correct and successful data input, the program will simulate the system with basic search, detection and tracking functionality. This document provides a high level overview of simulation functionality and a description of running the simulation.

Technical Overview

Coordinate System:

As mentioned in the introduction, the emphasis of this simulator is to simulate a radar system that consists of multiple radar faces. Thus, a universal coordinate system must be established. Currently, the program uses a simple “flat-earth” coordinate model where the radar system’s location is arbitrary and irrelevant, but acts as the centering point for the simulation environment (the radar system acts as the graphical origin). Additionally, a fixed spherical coordinate system is assumed with the flat-earth model. In terms of elevation, the “ground” acts as the 0 degree elevation position, while 90 degrees in elevation is perpendicular, away from the ground. For the azimuth plane, a 2D picture of the assumed coordinate system is shown below.



Example of flat-earth spherical coordinate system in azimuth plane

Simulation Time & Reference Frames/Intervals:

Once the user characterizes the radar system and simulation environment, the simulation can begin for the user-inputted amount of time. Once the simulation begins, all radar faces independently and simultaneously start searching for targets until the overall simulation time has been reached. This independent and simultaneous searching of radar faces is achieved by using reference frames/intervals, which the duration is specified by the user. At the beginning of a frame, the simulation environment (targets) is “saved”. Then, one radar face is chosen to search for potential targets. This searching on said radar face lasts during the entire frame duration. The time inside the frame is incremented based on the dwell time (time spent scanning one specific position) of the current search sector of the radar face. Once the frame time/limit has been reached, the entire environment (targets) is restored/reseted to the environment at the beginning of said frame. Then the same process continues for another radar face, until all radar faces have been simulated for that specific frame. After all the radar face’s in the system have been simulated to the reference frame, the whole simulation and environment is updated by the frame duration and the next reference frame/interval can begin. For example, say a simulated radar system has three radar face’s and the simulation is currently at 15 minutes 30 seconds and is about to start a five second frame (five seconds for simplicity). The environment is saved to its current state at 15 mins and 30 seconds. Next, one of the radar face’s is chosen to search. This radar face will search for five seconds or until the simulation time reaches 15 minutes and 35 seconds. Once the current frame reaches 5 seconds or 15 minutes 35 seconds, the simulation environment is repositioned/restored to 15 minutes and 30 seconds and the next radar face will search the same frame for five seconds. And with a radar system of three faces, this process will happen again another time. Once all three faces have been simulated for the frame of 15 minutes 30 seconds to 15 minutes 35 seconds, the entire simulation environment will then be updated to 15 minutes 35 seconds. Then the entire process repeats again, but now for the frame of 15 minutes 35 seconds to 15 minutes 40 seconds. This process continues until the overall simulation duration (configured by user) has been reached.

Search, Detect & Track

Currently the simulation supports basic search, detection and tracking functionality. As described above, each radar face independently searches for targets throughout the entire duration of the simulation. Each radar face will have a specific position with an FOV (field-of-view) extent, characterized by the user. Within a radar face’s FOV extent, there will be one or multiple search sectors, each with their own searching extents in terms of azimuth, elevation and range. Initial starting scanning positions are randomized for every search sector, but all radar faces and search sectors follow the same searching pattern. This searching pattern for search sectors is scanning the azimuth plane clockwise incrementing each position by the radar face’s searching half-power beamwidth. When the current searching position for the search sector reaches its azimuth search extent, the next scanning position will be repositioned to the beginning (most counterclockwise position) of the azimuth search extent and the elevation scanning position is incremented by the radar face’s searching half-power beamwidth. Then the searching pattern resumes to scanning the entire azimuth search extent, but now at

the new elevation position. This pattern repeats until the elevation search extent limit has been reached. At this point, the current search sector will reset to the beginning of the azimuth and elevation search extents (lowest counterclockwise position). An example will be given for better understanding.

For example, a search sector has an azimuth search extent of 15 degrees to 30 degrees and an elevation extent of 45 degrees to 60 degrees. The first search beam will be initialized to a random position within this search extent, say 20 degrees azimuth and 50 degrees elevation, (20,50)degrees, for this example. After the first search beam scans at (20,50)degree position, the next scanning position will be incremented in the azimuth plane by the half-power beamwidth. If the half-power beamwidth is 5 degrees, then the next scanning position is (25,50)degrees then (30,50)degrees. At (30,50)degrees scanning position, the azimuth search extent limit has been reached, so the next scanning position will be positioned to 20 degrees azimuth and the elevation will increment by half-power beamwidth. For this example the scanning position will now be (15,55)degrees. The pattern continues by incrementing the azimuth plane until the azimuth search extent limit has been met again, which then the elevation position will increment. Once the scanning position reaches both azimuth and elevation search limits, (30,60)degrees, the next scanning position will be to the start of both extents, which is (15,45)degrees for this example.

If a radar face has multiple search sectors, all search sectors themselves still follow the pattern described above, but the radar face itself has a pattern of selecting search sectors to send searching beams. This pattern is simply sending one search beam through a search sector then incrementing to the next search sector to send another search beam. For example if a radar face has 3 search sectors. The first search beam will be sent through the current scanning position of search sector one. The next search beam will be sent through the current scanning position of search sector two. And the next search beam will be sent through the current scanning position of search sector three. Since this example only has 3 search sectors, the next search beam will be sent through sector one’s new scanning position (due to the search sector pattern described above) and the pattern continues.

There are two forms of detection in this program, searching detection and tracking detection. Tracking detection will be covered in the tracking functionality details. A search detection is a detection made from a search beam from one of the radar’s search sectors. There is a criteria that must be met in order to make a search detection. This criteria consists of a target being inside the search beam's current half-power beamwidth scanning position, if the calculated SNR (signal-to-noise ratio) value exceeds the user-inputted threshold, and if the target is within the range extent limits specified by the user. If all these criteria are met, then a detection has been made and a confirmation beam will be immediately sent to confirm if the detection is indeed a target. If the confirmation beam sent also passes all the criteria above, then a tracking initialization/profile is made to start tracking the target.

Once a target has been detected and confirmed by the confirmation beam, a tracking profile will be created. This tracking profile is a tracking algorithm used to estimate a target’s next position from its current detected position. Currently, the tracking algorithm used is an alpha-beta-gamma (a-b-y) filter. When a tracking profile is created, a rough initial estimation is made for the target’s next position. At the start of each reference frame/interval (described in the “Simulation Time” section) the tracker will send a tracking beam to the estimated position. If the

target is detected by the track beam in the estimated position, the tracking profile continues and another estimation is made on the target's next position. If the target is not detected by the track beam in the estimated position, the tracking profile is deactivated. The more times a target is detected by a track beam from the tracking profile, the more refined the algorithm becomes at tracking said target.

While searching detections are made independently by individual search sectors within a radar face, tracking detections work in a different fashion. The radar tracker utilizes the whole radar system to track a target. This may be more apparent when describing the criteria needed for a tracking detection. For a tracking detection the criteria that must be met is, the target is currently within the track beam's current half-power beamwidth scanning position, the tracking SNR value exceeds the user-inputted threshold, and the target must be within the FOV extent of any radar face. The last criteria is what makes track detections different from search detections. As long as the target is still within the FOV extent of any radar face within the radar system, tracking can continue without losing track.

Using the Simulation

Configuring and Starting Simulation

```cpp
#include "configManager.h"
#include <time.h>

using namespace std;

int main() {
    //Simulation setup
   srand(time(0));      //Creates unique seed which is used to randomize starting scanning positions
   configManager test1; //Creating configuration manager object which handles .txt files containing input/output data

    //Below is inputting all .txt files containing input parameters needed to run simulation
    test1.inputRadarFile("inputFiles/RadarInfo.txt"); //File containing radar.h parameter data
    test1.inputFaceFile("inputFiles/FaceInfo.txt"); //File containing radarFace.h parameter data
    test1.inputSectorFile("inputFiles/SectorInfo.txt"); //File containing searchSector.h parameter data
    test1.inputTargetFile("inputFiles/TargetInfo.txt"); //File containing target.h parameter data

    radar simRadar; //Creating radar object to be simulated
    test1.initializeInputData(simRadar); //Assigns parameter data from .txt files above to simuated radar object
    //simRadar.printSimInfo(simRadar);   //Optional function, prints to console all parameter/characteristic data gathered from .txt files

    //Start of simulation
    simRadar.startSimulation(); //Function starts radar simulation for created radar object

    //Post simulation
    //test1.searchDataOutput(simRadar,"outputFiles/searchData.txt"); //Optional function, prints simulation search data to .txt file
    test1.closeInputFiles(); //Closes all files (from above) used to input data for simulation

    return 0;
}
```

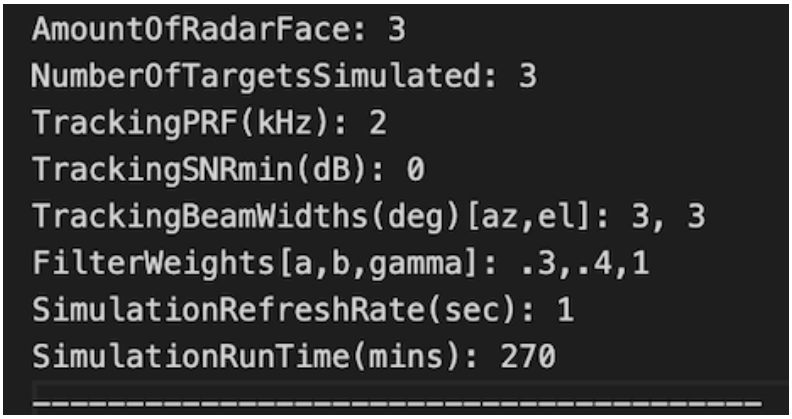
Example main.cpp

As seen from the picture above of the example main.cpp, configManger.h and time.h classes are included, along with the srand() function. While the time.h class and srand() function are not essential, srand() utilizes time.h to create a unique seed which is used for randomizing the starting scanning position for search sectors. ConfigManager.h must be included. Currently the program requires the input of four different text files in main.cpp for configuring the simulation. Each text file must be “inputted” (functions shown in example main.cpp) before initializing all the data contained in the inputted files. For inputting the files, the files can have any name as long as the appropriate rootpath/name is inputted with the corresponding function input. There is a specific format that the textfiles must follow. This format will be described later in this section. After inputting all the necessary textfiles, the initialization function must be called. This function collects the data stored in the input files and configures the data to the simulation. Upon successful data input and initialization, the simulation can begin with the startSimulation() function. A description and example of each input text file and its formatting is shown below.

Inputted Textfiles

Text files follow a similar format. NameOfParameter, (units of parameter), [sequence of data]. The ":" is a delimiter which signifies the beginning of data collection for parameters. Only numbers and commas should be used after the ":". The "-" is a delimiter value which signifies the end of all parameter input for a designated object. Following the "-" should be the beginning input for the next object.

1. Radar: defines the simulation environment (number of targets), along with physical characteristics of the radar, such as number radar faces and tracker characteristics.



Example radar input text file

Description,
AmountOfRadarFace: amount of radar faces in radar system
NumberOfTargetsSimulated: amount of targets to be simulated in environnement
TrackingPRF: tracking pulse repetition frequency (beams per second)
TrackingSNRmin: minimum signal-to-noise ratio needed for tracking detection
TrackingBeamWidths: half-power beamwidths (3db) of track beam
FilterWeights: Alpha-Beta-Gamma tracking weights
SimulationRefreshRate: Length of refresh interval/frame (described in "Simulation Timing")
SimulationRunTime: How long the simulation runs for in simulated time

2. Radar Face: defines the internal characteristics of a radar face on the radar and how many search sectors.

```
Boresight[azimuth,elevation](deg,deg): 0,45
FovAzExtent[clockwise](deg,deg): 300,60
FovElExtent[downToUp](deg,deg): 0,90
3dbBeamwidth[az,el](deg,deg): 5,5
DetectableSNRmin: 0
WaveFrequency[Ghz]: 90
Bandwidth(kHz): 30
EffectiveAttenaArea(m^2): 5
PeakPower(kW): 150
NoiseFigure[dB]: 2.5
TotalSystemLoss[dB]: 9
AmountSearchSectors: 3
-------------------------------------------------------
```

Example radar face input text file

Description,
Boresight: middle position of radar face
FovAzExtent: azimuth field-of-view extent
FovElExtent: elevation field-of-view extent
3dbBeamwidth: half-power beamwidth of search beams
DetectableSNRmin: minimum detectable signal-to-noise ratio for search detection
WaveFrequency: frequency of search beam/wave
Bandwidth: bandwidth of search beam/wave
EffectiveAttenaArea: effective antenna area (accounts for gain)
PeakPower: peak power supplied to radar face
NoiseFigure: noise figure or internal loss of radar system
TotalSystemLoss: overall losses due to processing, environmental effects, etc.
AmountSearchSectors: amount of search sectors used for radar face

If there are multiple radar faces (the specific amount must be inputted in the radar text file), the above parameters must be copied and pasted below the “-”. This effectively creates a new set of data to be inputted for the next radar face. If there are more sections than what is specified in the radar input text file, the simulation will result in compiling errors.

3. Search Sector: defines search timing and characteristics of a search sector

```
1.0AzExtent[clockwise](deg,deg): 300,340
ElExtent[downToUp](deg,deg): 0,90
RangeExtent[min,max](Km): 0,200
refreshRate(sec): 10
-------------------------------------------------------
```

Example search sector input text file

AzExtent: azimuth searching field-of-view extent
ElExtent: elevation searching field-of-view extent
RangeExtent: distance which detection is unambiguous
refreshRate: time needed to search entire search sector volumetrically

If there are multiple search sectors for radar face (the specific amount must be inputted in the radar face text file), the above parameters must be copied and pasted below the "-". This effectively creates a new set of data to be inputted for the next search sector. If there are more sections than what is specified in the radar input text file, the simulation will result in compiling errors. Due to the possibility of each radar face having multiple search sectors, all search sectors must be defined for the first radar face before moving to the next radar's face search sectors. This is shown in the above example with a 1.0 before azExtent. For example, a system has two radar faces, one with two search sectors and the other with three. The first radar face that is inputted in the radar face input text file is the face consisting of two search sectors. Thus, in the search sector input text file, these two search sectors must be defined before the next radar's face search sectors, which consists of three in this example.

4. Target: defines a simulated target's position, trajectory and radar-cross-section (RCS).

```
Target 1 coordinates[x,y,z](km relative to boresight):51.22,13.725,53
Target 1 velocity[x,y,z](m/s): 100,0,0
Target 1 acceleration[x,y,z](m/s^2): 0,0,0
Target 1 RCS[DBsm]: 1
-----------------------------------------------------------------
```

Example target input text file

Coordinates: cartesian coordinates of targets position relative to radar system
Velocity: velocity values of simulated target
Acceleration: acceleration values of simulated target
RCS: radar-cross-section of simulated target

If there are multiple targets (the specific amount must be inputted in the radar text file), the above parameters must be copied and pasted below the "-". This effectively creates a new set of data to be inputted for the next target. If there are more sections than what is specified in the radar input text file, the simulation will result in compiling errors.

<u>Expanding Program Functionality</u>

While this program is running and functional, much work must be done to achieve the ultimate goal given for this simulator. Under the current framework input text files and configManager.h are directly intertwined and most of the current radar functionality logic is located in radar.h. If a new parameter is added to a text file or if an existing parameter is deleted from a text file or if an entirely new text file is made for new user input, the corresponding logic must be implemented/modified within configManager.h. ConfigManager.h contains all the logic

used to gather the data contained in text files and assigns the data to using class setter and getter functions. If radar functionality needs to be modified, it will most likely be done in radar.h. Currently, radar.h contains the logic needed for the current search, detect and track functions (tracking algorithm is contained in tracker.h). More specifically, radar.h contains the startSimulation() function which acts as the main simulation loop that drives the simulation. If modifications are made to any other part of the program, commented source code is available on this GitHub page to further describe the low level functionality of existing classes, functions and variables. Additionally, an example radar system and simulation environment is used with the source code, which can be used as an additional reference. If there are any questions about the program, please reach out!