— 00_sommario.md —

Manuale Tecnico - Radar Target Simulator

Introduzione al Manuale

Questo documento costituisce il manuale tecnico completo per l'applicazione **Radar Target Simulator**. È destinato a sviluppatori, ingegneri di test e utenti avanzati che necessitano di una comprensione approfondita dell'architettura, delle funzionalità e delle decisioni di progettazione dell'applicazione.

Il manuale è organizzato in capitoli modulari per facilitare la consultazione e la manutenzione. Ogni capitolo è un file Markdown indipendente, collegato da questo sommario.

Indice dei Contenuti

— 01_introduzione.md —

Capitolo 1: Introduzione

1.1. Scopo dell'Applicazione

Il **Radar Target Simulator** è un'applicazione software progettata per l'ambiente di sviluppo e test di sistemi radar. Il suo scopo primario è generare e trasmettere dati di traccia sintetici (target simulati) verso un'unità di elaborazione radar (chiamata "server" o "device under test") per verificarne il comportamento e le prestazioni di tracciamento.

L'applicazione consente di:
* **Creare scenari complessi:** Definire uno o più target, ciascuno con una traiettoria tridimensionale programmabile composta da segmenti di volo (waypoint) e manovre dinamiche.
* **Simulare il movimento:** Eseguire una simulazione in tempo reale (o accelerato) che aggiorna lo stato cinematico di tutti i target secondo le traiettorie definite.
* **Trasmettere i dati:** Inviare gli stati dei target al sistema radar tramite diversi protocolli di comunicazione (SFP, TFTP, Seriale).
* **Ricevere e visualizzare dati reali:** Acquisire i dati di traccia elaborati dal sistema radar e visualizzarli in tempo reale su un Plan Position Indicator (PPI).
* **Analizzare le prestazioni:** Confrontare i dati "ground truth" della simulazione con i dati ricevuti dal radar per calcolare metriche di errore e valutare l'accuratezza del tracciamento.

L'applicazione è uno strumento fondamentale per il test "hardware-in-the-loop", permettendo di validare algoritmi di tracciamento radar in un ambiente controllato e ripetibile, senza la necessità di asset fisici reali.

1.2. Architettura di Alto Livello

L’applicazione è costruita su un’architettura modulare che separa l’interfaccia utente dalla logica di simulazione e comunicazione. I componenti principali interagiscono come mostrato nel diagramma seguente.

(Placeholder per l’immagine)

Descrizione dell’Immagine: `architettura_alto_livello.png`
Qui ti aspetti di vedere un diagramma a blocchi. Disegna quattro riquadri principali:
1. *Un riquadro grande a sinistra etichettato “**Interfaccia Utente (GUI - MainView)**”.*
2. *Un riquadro a destra etichettato “**Motore di Simulazione (SimulationEngine - Thread)**”.*
3. *Un riquadro in basso a destra etichettato “**Gestore Comunicazione (CommunicatorManager)**”. Questo blocco si collega a sua volta a blocchi più piccoli per “SFP”, “TFTP”, “Seriale”.*
4. *Un riquadro centrale, connesso a tutti gli altri, etichettato “**Hub Dati di Stato (SimulationStateHub)**”.*
Disegna frecce per mostrare le interazioni:
* *GUI -> SimulationEngine (Avvia/Ferma Simulazione, Carica Scenario).*
* *SimulationEngine -> SimulationStateHub (Scrive “Dati Simulati”).*
* *SimulationEngine -> CommunicatorManager (Invia comandi `tgtset`).*
* *CommunicatorManager -> SimulationStateHub (Scrive “Dati Reali” ricevuti dal radar).*
* *SimulationStateHub -> GUI (Fornisce dati per aggiornare PPI, tabelle, etc.).*

I componenti chiave sono:

- **Interfaccia Utente (GUI - `MainView`):** Basata su Tkinter, è il punto di controllo centrale per l’utente. Gira sul thread principale e orchestra tutte le operazioni. È responsabile della visualizzazione dei dati (PPI, log, tabelle) e della cattura degli input utente.

- **Motore di Simulazione (`SimulationEngine`):** Un thread separato che gestisce il ciclo di vita della simulazione. A intervalli regolari, calcola il nuovo stato di ogni target attivo e invia gli aggiornamenti al gestore della comunicazione. Funziona in background per non bloccare l’interfaccia utente.

- **Hub Dati di Stato (`SimulationStateHub`):** È il “cuore” thread-safe dell’applicazione. Funge da database in memoria che raccoglie e centralizza tutti i dati di stato, sia quelli generati dal simulatore (`simulated`) sia quelli ricevuti dal radar (`real`). Tutti gli altri componenti leggono e scrivono da questo hub, garantendo la coerenza dei dati tra i diversi thread.

- **Gestore della Comunicazione (`CommunicatorManager` e interfacce):** Un livello di astrazione che gestisce i diversi protocolli di comunicazione. Riceve comandi di alto livello (es. “invia questo scenario”) e li traduce in comandi specifici del protocollo selezionato (es. una serie di comandi seriali o un singolo payload JSON su SFP).

Questa architettura disaccoppiata permette di modificare o aggiungere un protocollo di comunicazione senza impattare il motore di simulazione, o di cambiare l'interfaccia grafica senza alterare la logica di calcolo.

1.3. A chi è rivolto questo manuale

Questo documento è stato redatto per un pubblico tecnico:

- **Sviluppatori Software:** Che necessitano di comprendere il flusso dei dati, le responsabilità di ogni modulo e le convenzioni di codice per manutenere o estendere l'applicazione.
- **Ingegneri di Test e Validazione:** Che usano l'applicazione per testare sistemi radar e hanno bisogno di comprendere in dettaglio come vengono generati i dati, come configurare scenari complessi e come interpretare i risultati dell'analisi.
- **Utenti Avanzati:** Che desiderano sfruttare al massimo le funzionalità dell'applicazione, inclusi gli strumenti di debug e la configurazione avanzata.

Questo manuale **non** è inteso come una guida rapida per l'utente finale, per la quale verrà prodotta una documentazione separata e più sintetica.

— 02_concetti_fondamentali.md —

Capitolo 2: Concetti Fondamentali

Questo capitolo descrive i concetti teorici e le convenzioni di base che governano il funzionamento del Radar Target Simulator. Una comprensione solida di questi principi è essenziale per utilizzare l'applicazione in modo efficace e per interpretarne correttamente i risultati.

2.1. Scenario, Target e Traiettoria

L'applicazione organizza i dati secondo una struttura gerarchica semplice ma potente:

- **Scenario:** È il contenitore di livello più alto. Rappresenta una singola sessione di test e contiene una collezione di uno o più target. Ogni scenario ha un nome univoco che permette di salvarlo e ricaricarlo.
- **Target:** Rappresenta un singolo oggetto (es. un aereo, un missile) che il simulatore deve muovere. Ogni target è definito da:
    - Un **ID** numerico univoco.
    - Una serie di **attributi di stato** (attivo, tracciabile).
    - Una **traiettoria** che ne descrive il movimento nel tempo.
- **Traiettoria:** È una sequenza ordinata di `Waypoint` che definisce il percorso di un target. La traiettoria non è un semplice elenco di coordinate, ma una serie di istruzioni di manovra.

- **Waypoint (o Punto di Manovra):** È il blocco costruttivo fondamentale di una traiettoria. Ogni waypoint descrive un segmento di volo o una manovra specifica. L'applicazione supporta diversi tipi di manovra (`ManeuverType`), tra cui:
    - **Fly to Point:** Usato tipicamente come punto di partenza, definisce una posizione iniziale in coordinate polari (range, azimut, altitudine) e uno stato cinematico iniziale (velocità, heading).
    - **Fly for Duration:** Istruisce il target a mantenere una velocità, un heading e un'altitudine costanti per una durata di tempo specificata.
    - **Dynamic Maneuver:** Permette di simulare manovre complesse applicando accelerazioni (longitudinali, laterali e verticali) per una data durata.

Quando una simulazione viene avviata, il `SimulationEngine` calcola in anticipo il percorso completo (un elenco di punti `(t, x, y, z)`) per ogni target, interpolando o campionando le manovre definite dai waypoint. Questo percorso pre-calcolato diventa la "ground truth" per la durata della simulazione.

2.2. Il Sistema di Coordinate

La gestione corretta dei sistemi di coordinate è l'aspetto tecnicamente più critico dell'applicazione. È fondamentale per garantire che il movimento simulato sia realistico e indipendente dai movimenti dell'ownship (la nostra piattaforma).

2.2.1. Il Problema: Un Sistema di Riferimento Relativo

Un approccio ingenuo potrebbe calcolare la posizione dei target sempre in modo relativo alla posizione *corrente* dell'ownship. Questo porterebbe a un comportamento irrealistico: se l'ownship virasse a destra, l'intera scena simulata ruoterebbe solidalmente con esso, come se i target fossero "incollati" a un sistema di riferimento centrato sull'aereo. Questo è fisicamente scorretto, poiché le traiettorie dei target nel mondo reale sono indipendenti dai movimenti dell'osservatore.

2.2.2. La Soluzione Adottata: Il "Simulation Frame"

Per risolvere questo problema, abbiamo introdotto il concetto di un **sistema di riferimento fisso per la simulazione**, chiamato "Simulation Frame".

1. **Snapshot a T=0:** Nel momento esatto in cui l'utente avvia la simulazione, il `SimulationController` cattura lo stato corrente dell'ownship (posizione `(x, y)` e `heading` assoluti nel mondo) fornito dal sistema reale.
2. **Origine Fissa:** Questo snapshot viene salvato nello `SimulationStateHub` come **"origine della simulazione"**. Questa origine diventa un punto di riferimento immutabile per tutta la durata della sessione.
3. **Calcolo delle Traiettorie:** Il `SimulationEngine` calcola e aggiorna le posizioni di tutti i target simulati *all'interno di questo Simulation Frame*. Per convenzione, l'origine `(0,0)` del Simulation Frame corrisponde alla posizione dell'ownship a T=0, e il suo asse Y è allineato con l'heading dell'ownship a T=0.

4. **Trasformazione per la Visualizzazione:** Per disegnare i target sulla PPI, il `ppi_adapter` esegue una trasformazione in tempo reale:
    a. Prende la posizione di un target nel Simulation Frame.
    b. La **ruota** e la **trasla** utilizzando i dati dell'origine della simulazione per convertirla in una posizione assoluta nel "mondo".
    c. Calcola la sua posizione finale **relativa** alla posizione *corrente* dell'ownship per la corretta visualizzazione sulla PPI.

Questo approccio garantisce che le traiettorie simulate si evolvano in un sistema di coordinate stabile e inerziale, mentre l'ownship è libero di muoversi nel mondo come dettato dai dati di navigazione reali.

2.2.3. Dati Simulati vs. Dati Reali

Lo `SimulationStateHub` mantiene due storie separate per ogni target ID:
* **Dati Simulati:** La "ground truth" generata dal `SimulationEngine`, espressa nel Simulation Frame.
* **Dati Reali:** Le posizioni dei target ricevute dal sistema radar, espresse in coordinate assolute del mondo.

Questa separazione è ciò che permette all'applicazione di confrontare i due flussi di dati e calcolare le metriche di errore di tracciamento.

2.3. Modalità di Visualizzazione PPI

Il PPI Display supporta due modalità di visualizzazione standard per orientare l'operatore.

(Placeholder per l'immagine)

Descrizione dell'Immagine: `modalita_ppi.png`
Qui ti aspetti di vedere due cerchi PPI affiancati che mostrano la stessa situazione tattica. Per esempio, l'ownship ha un heading di +45° (Nord-Est) e un target si trova a Nord.
* **PPI a Sinistra (North-Up):**
* La griglia ha lo 0° in alto.
* L'icona dell'ownship al centro è ruotata di 45° verso destra.
* Il settore di scansione è anch'esso ruotato di 45° a destra.
* Il target è disegnato in alto, sulla linea dei 0°.
* **PPI a Destra (Heading-Up):**
* L'icona dell'ownship e il settore di scansione sono fissi e puntati in alto.
* La griglia è ruotata di -45° (in senso orario). La tacca dei 0° (Nord) si trova in basso a destra.
* Il target è disegnato in alto a sinistra, a un angolo di -45° rispetto alla linea di fede superiore.

2.3.1. North-Up (Nord in Alto)

Questa è la modalità "mappa". L'orientamento del display è fisso, con la linea dei 0° che punta sempre verso il Nord geografico.

- **Griglia PPI:** Fissa, con 0° in alto.
- **Elementi dell'Ownship (Icona, Settore Radar):** Ruotano per riflettere l'heading corrente dell'aereo. Questo permette all'operatore di vedere la propria orientazione rispetto al mondo esterno.
- **Mondo Esterno (Target, Tracce):** Disegnato a coordinate assolute sulla mappa fissa.

2.3.2. Heading-Up (Prua in Alto)

Questa è la modalità "vista dal cockpit". L'orientamento del display è dinamico e allineato con la prua dell'aereo. La linea dei 0° del display rappresenta sempre la direzione in cui l'aereo sta andando.

- **Griglia PPI:** Ruota in senso opposto alle virate dell'aereo per mantenere l'heading allineato in alto.
- **Elementi dell'Ownship (Icona, Settore Radar):** Fissi, puntati verso l'alto.
- **Mondo Esterno (Target, Tracce):** Ruota solidalmente con la griglia.

La corretta implementazione di queste due modalità è fondamentale per fornire all'operatore una consapevolezza situazionale accurata.

— 03_interfaccia_principale.md —

Capitolo 3: Guida all'Interfaccia Principale

L'interfaccia utente (GUI) del Radar Target Simulator è progettata per fornire un controllo centralizzato su tutte le fasi della simulazione, dalla creazione degli scenari all'analisi dei risultati. Questa sezione descrive in dettaglio ogni componente della finestra principale.

3.1. Panoramica Generale

La finestra principale è suddivisa in diverse aree funzionali, ciascuna con un ruolo specifico.

(Placeholder per l'immagine)

Descrizione dell'Immagine: `panoramica_gui.png`
Qui ti aspetti di vedere uno screenshot a schermo intero dell'applicazione. Usa un editor di immagini per aggiungere delle etichette numerate che puntano alle seguenti sezioni:
1. *Etichetta "1" che punta al **PPI Display**.*
2. *Etichetta "2" che punta al **Pannello di Controllo Scenari** (in alto a destra).*
3. *Etichetta "3" che punta al **Pannello di Lavoro a Schede** (sotto il controllo scenari).*
4. *Etichetta "4" che punta al **Pannello di Connessione** (sopra il PPI).*
5. *Etichetta "5" che punta al **Pannello dei Log** (in basso).*
6. *Etichetta "6" che punta alla **Barra di Stato** (in fondo).*

3.2. Il PPI Display

Il Plan Position Indicator (PPI) è il componente visivo centrale dell'applicazione.

- **Scopo:** Fornisce una rappresentazione grafica in tempo reale dello scenario tattico, mostrando la posizione dell'ownship, dei target simulati e dei target reali ricevuti dal radar.
- **Caratteristiche:**
    - **Griglia Polare:** Mostra anelli di range (in miglia nautiche) e linee di azimut (in gradi).
    - **Controlli Radar:** Permettono di cambiare il range massimo visualizzato e di attivare/disattivare l'animazione della linea di scansione dell'antenna.
    - **Modalità di Visualizzazione:** Offre le modalità `North-Up` e `Heading-Up` (vedi Capitolo 2.3 per i dettagli).
    - **Opzioni di Visualizzazione:** Consente di mostrare/nascondere selettivamente i punti e le tracce (trail) dei target simulati e reali.
    - **Legenda:** Identifica i simboli grafici per l'ownship, i target simulati e i target reali.

3.3. Pannello di Controllo Scenari

Questa sezione, situata in alto a destra, è dedicata alla gestione dei file di scenario.

- **Selettore Scenario:** Un menu a tendina per caricare rapidamente uno scenario esistente. La selezione di uno scenario lo carica immediatamente nell'area di editing.
- **Pulsanti di Gestione:**
    - **New…:** Apre una finestra di dialogo per creare un nuovo scenario vuoto, chiedendone il nome.
    - **Save:** Salva le modifiche correnti sovrascrivendo il file dello scenario attualmente selezionato.
    - **Save As…:** Salva le modifiche correnti in un nuovo file di scenario, chiedendone il nome.
    - **Delete:** Rimuove permanentemente il file dello scenario attualmente selezionato.

3.4. Pannello di Lavoro Principale (a schede)

Questo pannello a schede organizza le diverse fasi del lavoro.

3.4.1. Scheda "Editing scenario"

È l'area di lavoro predefinita per la creazione e modifica delle traiettorie. Contiene la **Lista dei Target** (`TargetListFrame`), una tabella che mostra i parametri iniziali di ogni target dello scenario corrente. Da qui è possibile:
* **Add:** Aggiungere un nuovo target allo scenario, aprendo l'Editor di Traiettoria.
* **Remove:** Rimuovere il target selezionato.
* **Edit Trajectory…:** Aprire l'Editor di Traiettoria per modificare il target selezionato.

3.4.2. Scheda “Simulation”

Questa scheda contiene i controlli per l’esecuzione della simulazione in tempo reale.

(Placeholder per l’immagine)

Descrizione dell’Immagine: `pannello_simulazione.png`
Qui ti aspetti uno screenshot che mostra in dettaglio la scheda “Simulation”, evidenziando:
1. *I pulsanti “Start Live” / “Stop Live”.*
2. *Lo slider di progresso della simulazione.*
3. *La sezione “Ownship State”.*
4. *La tabella “Active Targets”.*

- **Live Simulation Engine:**
  - **Start Live / Stop Live:** Pulsanti per avviare e fermare la simulazione.
  - **Speed / Update(s):** Controlli per accelerare la simulazione e definire la frequenza di invio degli aggiornamenti al radar.
  - **Reset Sim:** Riporta la simulazione allo stato iniziale (T=0) senza fermarla.
  - **Reset Radar:** Invia un comando di reset al radar per cancellare tutte le tracce esistenti.
  - **Slider di Progresso:** Mostra l’avanzamento della simulazione e permette di “cercare” (seek) un punto specifico nel tempo trascinando il cursore (solo a simulazione in pausa).
- **Ownship State:** Un pannello di sola lettura che visualizza i dati di navigazione più recenti dell’ownship (posizione, altitudine, heading, velocità) ricevuti dal sistema reale.
- **Active Targets:** Una tabella che, durante la simulazione, mostra i dati geografici calcolati in tempo reale per i target *simulati*.

3.5. Pannello di Connessione

Posizionato sopra il PPI, questo pannello gestisce la connessione con il sistema radar.

- **Indicatori di Stato:** Mostra il tipo di connessione configurata (SFP, TFTP, Seriale) e i parametri principali (es. indirizzo IP e porte).
- **Settings…:** Apre la finestra di configurazione dettagliata della connessione (vedi Capitolo 5).
- **Connect / Disconnect:** Stabilisce o interrompe la comunicazione con il device under test.

3.6. Log e Barra di Stato

Pannello dei Log

Situato nella parte inferiore della finestra, visualizza i messaggi di log generati dall’applicazione. È uno strumento essenziale per il debug e per monitorare le operazioni in background. I messaggi sono colorati in base al loro livello di severità (es. INFO, WARNING, ERROR).

Barra di Stato

La barra in fondo alla finestra fornisce informazioni di stato rapide:
* **Indicatori di Connessione (Target/LRU):** Due LED grafici (rosso per disconnesso, verde per connesso) mostrano lo stato della comunicazione.
* **Messaggio di Stato:** Un'area di testo che mostra lo stato corrente ("Ready", "Simulation running", etc.) o messaggi temporanei.
* **Rate / Latency:** Visualizza le metriche di comunicazione in tempo reale, come i pacchetti al secondo (pkt/s) e la latenza stimata.
* **Uso Risorse:** Mostra l'utilizzo corrente di CPU e memoria da parte dell'applicazione.

— 04_gestione_scenari_e_target.md —

Capitolo 4: Gestione di Scenari e Target

Questo capitolo illustra il flusso di lavoro pratico per creare, modificare e gestire gli scenari di simulazione e i target che li compongono. Il cuore di questa attività si svolge attraverso l'Editor di Traiettoria e l'Editor dei Waypoint.

4.1. Creare, Salvare e Caricare uno Scenario

La gestione degli scenari avviene tramite il **Pannello di Controllo Scenari** (descritto nel Capitolo 3.3). Il flusso tipico è:

1. **Creazione:** Si preme il pulsante `New...` e si assegna un nome al nuovo scenario. L'applicazione crea uno scenario vuoto e lo seleziona.
2. **Popolamento:** Si aggiungono uno o più target utilizzando il pannello "Editing scenario" (vedi sezione successiva).
3. **Salvataggio:** Le modifiche vengono salvate periodicamente tramite il pulsante `Save`. Utilizzando `Save As...` è possibile creare una copia dello scenario corrente con un nuovo nome.
4. **Caricamento:** Utilizzando il menu a tendina, è possibile passare da uno scenario all'altro. L'applicazione caricherà la configurazione del target dal file `.json` corrispondente.

Tutti gli scenari sono salvati come file `JSON` individuali all'interno di una cartella `scenarios/` situata nella directory principale dell'applicazione.

4.2. L'Editor di Traiettoria (`TrajectoryEditorWindow`)

Questa finestra di dialogo è il principale strumento per definire il comportamento di un singolo target. Si apre quando si aggiunge un nuovo target o se ne modifica uno esistente.

(Placeholder per l'immagine)

Descrizione dell'Immagine: `editor_traiettoria.png`
Qui ti aspetti uno screenshot della `TrajectoryEditorWindow`. Evidenzia le due aree principali:

1. *Un riquadro a sinistra etichettato "**Lista Waypoint e Controlli**", che mostra la tabella con le manovre e i pulsanti "Add", "Edit", "Remove".*
2. *Un riquadro a destra etichettato "**Anteprima Traiettoria e Controlli di Playback**", che mostra il PPI con la traccia disegnata e i pulsanti Play/Pause/Stop.*

La finestra è divisa in due sezioni:

- **A sinistra: Lista dei Waypoint**
  - Questa tabella elenca in sequenza tutte le manovre (waypoint) che compongono la traiettoria del target.
  - I pulsanti `Add`, `Edit`, `Remove` permettono di gestire la lista.
  - L'opzione `Use Spline` consente di interpolare i punti della traiettoria utilizzando una curva Catmull-Rom per un movimento più fluido (richiede almeno 4 waypoint).
- **A destra: Anteprima della Traiettoria**
  - Un PPI dedicato mostra un'anteprima statica del percorso completo del target.
  - I controlli di playback (`Play`, `Pause`, `Stop`, `Speed`) permettono di avviare una simulazione visiva della traiettoria per verificarne il comportamento dinamico prima di salvarla. Lo slider sottostante consente di navigare rapidamente lungo la timeline della simulazione.

4.3. L'Editor dei Waypoint e le Manovre (`WaypointEditorWindow`)

Questa è la finestra modale in cui si definiscono i parametri di una singola manovra. Il contenuto della finestra cambia dinamicamente in base al "Tipo di Manovra" selezionato.

(Placeholder per l'immagine)

Descrizione dell'Immagine: `editor_waypoint_dynamic.png`
Qui ti aspetti uno screenshot della `WaypointEditorWindow` con la "Dynamic Maneuver" selezionata. Evidenzia le sezioni chiave:
1. *Il menu a tendina "Maneuver Type".*
2. *La sezione "Turn Definition" con le opzioni "By G-Force" e "By Turn Rate".*
3. *I campi per le accelerazioni longitudinali e verticali.*

4.3.1. Fly to Point

Questa manovra è **obbligatoria come primo waypoint** di ogni traiettoria, poiché definisce lo stato iniziale del target.
* **Parametri Principali:**
* `Target Range`, `Target Azimuth`, `Target Altitude`: Definiscono la posizione iniziale del target in coordinate polari relative all'origine della simulazione.
* `Initial Velocity`, `Initial Heading`: Definiscono il vettore velocità iniziale del target.
* `Duration to Point`: Se usato come waypoint successivo al primo, questo parametro

definisce il tempo che il target impiegherà per raggiungere le coordinate specificate. La velocità risultante viene calcolata e mostrata automaticamente.

4.3.2. Fly for Duration

Questa manovra istruisce il target a mantenere uno stato cinematico costante.
* **Parametri Principali:**
* `Duration`: Per quanti secondi mantenere lo stato.
* `Constant Velocity`, `Constant Heading`, `Constant Altitude`: I valori cinematici da mantenere.

4.3.3. Dynamic Maneuver

Questa è la manovra più complessa e potente, che permette di simulare virate, accelerazioni e cabrate/picchiate. Il movimento viene calcolato integrando le accelerazioni nel tempo.
* **Parametri Principali:**
* `Duration`: La durata della manovra.
* `Maneuver Speed`: La velocità di riferimento per il calcolo della virata.
* **Turn Definition:** È possibile definire la virata in due modi equivalenti:
1. **By G-Force:** Specificando l'accelerazione laterale in 'g'. Questo è il modo più comune in ambito aeronautico.
2. **By Turn Rate:** Specificando la velocità angolare in gradi al secondo (°/s).
* **Soluzione Tecnica:** I due campi sono sincronizzati. Modificandone uno, l'altro viene ricalcolato automaticamente in base alla velocità di manovra, usando la formula `a = v * ω`, dove `a` è l'accelerazione, `v` è la velocità e `ω` è la velocità angolare.
* **Longitudinal Accel (g):** Accelerazione lungo l'asse di volo del target (per accelerare o decelerare).
* **Vertical Accel (g):** Accelerazione lungo l'asse verticale del target (per cabrate o picchiate).

La combinazione di questi waypoint permette di costruire traiettorie estremamente varie e realistiche, adatte a coprire un'ampia gamma di scenari di test.

— 05_configurazione_e_connessione.md —

Capitolo 5: Configurazione della Comunicazione

Una volta definito uno scenario, il passo successivo è configurare la comunicazione per trasmettere i dati dei target simulati al sistema radar (device under test). L'applicazione supporta diversi protocolli, gestiti tramite un'unica finestra di configurazione.

5.1. La Finestra `ConnectionSettingsWindow`

Questa finestra di dialogo, accessibile dal menu `Settings -> Connection...` o dal pulsante `Settings...` nel Pannello di Connessione, centralizza la configurazione per le due interfacce logiche principali: **Target Connection** e **LRU Connection**.

(Placeholder per l'immagine)

Descrizione dell'Immagine: `finestra_connessioni.png`
Qui ti aspetti uno screenshot della finestra `ConnectionSettingsWindow`. Evidenzia le seguenti aree:
1. *Un riquadro attorno alla sezione "Target Connection".*
2. *Un riquadro attorno alla sezione "LRU Connection" (anche se non ancora utilizzata, va mostrata).*
3. *Una freccia che punta al menu a tendina "Type" per selezionare il protocollo (SFP, TFTP, Seriale).*
4. *Una freccia che punta alle schede sottostanti, che cambiano in base al tipo selezionato.*
5. *Una freccia che punta al pulsante "Test Connection".*

La finestra permette di configurare in modo indipendente la comunicazione per l'invio dei dati dei target (`Target Connection`) e per un'eventuale comunicazione secondaria con un'altra LRU (`LRU Connection`). Per ogni connessione, è possibile scegliere il tipo di protocollo e impostarne i parametri specifici.

5.2. Dettaglio dei Protocolli

5.2.1. Configurazione SFP (Simple Fragmentation Protocol)

SFP è un protocollo basato su UDP progettato per la trasmissione affidabile di dati. È il metodo di comunicazione più avanzato e raccomandato per l'applicazione.

- **Server IP:** L'indirizzo IP del sistema radar che riceverà i dati.
- **Server Port:** La porta UDP su cui il sistema radar è in ascolto per i pacchetti SFP.
- **Local Port:** La porta UDP su cui l'applicazione si metterà in ascolto per ricevere i pacchetti di ritorno (es. ACK SFP o dati reali dal radar). **È fondamentale che questa porta non sia in uso da altre applicazioni sulla macchina.**
- **Use JSON Protocol:** Se selezionato, il simulatore invierà gli aggiornamenti dei target utilizzando un payload JSON compatto, invece del formato testuale legacy. Questo è il metodo preferito per i sistemi radar che lo supportano, in quanto più efficiente e strutturato.
- **Prediction Offset (ms):** Un valore in millisecondi che introduce una compensazione manuale per la latenza. Il motore di simulazione anticiperà il calcolo della posizione del target di questo intervallo di tempo prima di inviare i dati, per compensare la latenza di rete e di elaborazione del server.

5.2.2. Configurazione TFTP (Trivial File Transfer Protocol)

In questa modalità, l'applicazione genera uno script di comandi testuali e lo carica su un server TFTP in esecuzione sul sistema radar.

- **Server IP:** L'indirizzo IP del server TFTP.
- **Server Port:** La porta del servizio TFTP (di solito la 69).

5.2.3. Configurazione Seriale

Questa modalità utilizza una connessione seriale (es. RS-232/RS-422) per inviare i comandi testuali.

- **COM Port:** Il nome della porta seriale da utilizzare (es. `COM1` su Windows, `/dev/ttyS0` su Linux). Il menu a tendina viene popolato automaticamente con le porte rilevate sul sistema.
- **Baud Rate:** La velocità di trasmissione. Deve corrispondere a quella impostata sul dispositivo ricevente.
- **Altri Parametri (Parity, etc.):** Sebbene non esposti nell'interfaccia principale `ConnectionSettingsWindow`, possono essere configurati nel file `settings.json` se necessario.

5.3. Testare la Connessione

Il pulsante `Test Connection` esegue un rapido controllo per verificare la raggiungibilità e la configurazione di base del servizio selezionato, senza stabilire una connessione permanente.

- **SFP:** Tenta di aprire e associare un socket sulla porta locale specificata. Un successo indica che la porta è libera e utilizzabile.
- **TFTP:** Invia una richiesta di lettura (RRQ) per un file inesistente. Un successo si ha se il server risponde (anche con un errore "file not found"), confermando che il server TFTP è in esecuzione e raggiungibile. Un fallimento indica un timeout o un problema di rete.
- **Seriale:** Tenta di aprire la porta COM con i parametri specificati e la richiude immediatamente. Un successo indica che la porta esiste e non è bloccata da un'altra applicazione.

L'uso di questa funzione è fortemente raccomandato dopo aver modificato i parametri di connessione per diagnosticare rapidamente eventuali problemi di configurazione.

— 06_esecuzione_simulazione.md —

Capitolo 6: Esecuzione della Simulazione e Analisi

Questo capitolo descrive il flusso operativo completo per eseguire una sessione di test, dal caricamento di uno scenario all'analisi dei dati raccolti.

6.1. Flusso Operativo Tipico

Una sessione di test completa con il Radar Target Simulator segue tipicamente questi passaggi:

1. **Configurazione e Connessione:**
    - Verificare e, se necessario, modificare le impostazioni di connessione tramite la finestra `ConnectionSettingsWindow` (Capitolo 5).

    - Premere il pulsante `Connect` per stabilire la comunicazione con il sistema radar. L'indicatore LED nella barra di stato diventerà verde.
2. **Caricamento dello Scenario:**
    - Selezionare lo scenario di test desiderato dal menu a tendina nel "Pannello di Controllo Scenari". I target dello scenario verranno caricati e visualizzati nella tabella "Target List" e in anteprima sul PPI.
3. **Avvio della Simulazione:**
    - Passare alla scheda "Simulation".
    - Premere il pulsante `Start Live`. Questa azione innesca il processo di avvio descritto nella sezione successiva.
4. **Monitoraggio in Tempo Reale:**
    - Osservare il PPI: i target simulati (verdi) inizieranno a muoversi lungo le loro traiettorie. Se il sistema radar sta tracciando correttamente, appariranno i target reali (rossi) sovrapposti o vicini a quelli simulati.
    - Monitorare i pannelli "Ownship State" e "Active Targets" per dati cinematici numerici.
    - Controllare la barra di stato per metriche di comunicazione (rate, latenza) e il pannello dei log per eventuali messaggi di errore o informativi.
5. **Fine della Simulazione:**
    - La simulazione termina automaticamente quando tutti i target hanno completato la loro traiettoria.
    - È possibile interromperla manualmente in qualsiasi momento premendo il pulsante `Stop Live`.
6. **Analisi dei Dati:**
    - Al termine della simulazione (sia naturale che manuale), i dati della sessione (ground truth simulata, dati reali ricevuti e metadati) vengono salvati automaticamente in un file di archivio `.json`.
    - Navigare alla scheda "Analysis", aggiornare la lista e aprire la sessione appena conclusa per analizzare le prestazioni di tracciamento.

6.2. Il Processo di Avvio della Simulazione

Premere `Start Live` non avvia semplicemente il movimento dei target. Innesca una sequenza di operazioni orchestrata dal `SimulationController` per garantire uno stato iniziale pulito e sincronizzato:

1. **Reset del Radar (in background):** Viene inviato un comando di reset al sistema radar per cancellare tutte le tracce preesistenti. Questo assicura che il test parta da una situazione nota. L'applicazione attende una conferma che il radar sia "pulito" prima di procedere.
2. **Snapshot dell'Origine (T=0):** Viene catturato lo stato corrente dell'ownship (posizione e heading) e salvato come "origine della simulazione" fissa (vedi Capitolo 2.2.2).

3. **Invio dello Scenario Iniziale:** L'intera configurazione dello scenario (posizioni iniziali di tutti i target) viene inviata al sistema radar in un'unica operazione atomica (es. tramite un singolo payload JSON o una sequenza `tgtinit`).
4. **Avvio del `SimulationEngine`:** Solo dopo che i passaggi precedenti sono stati completati con successo, viene avviato il thread del `SimulationEngine`. Da questo momento in poi, il motore inizierà a calcolare gli stati successivi dei target e a inviare comandi di aggiornamento (`tgtset` o payload JSON) a intervalli regolari.

Questo processo garantisce che il sistema radar e il simulatore siano allineati allo stesso stato iniziale prima che il movimento dinamico abbia inizio.

6.3. Analisi Post-Simulazione

Al termine di ogni sessione, l'applicazione archivia i dati raccolti. Questi dati possono essere analizzati per una valutazione quantitativa delle performance.

6.3.1. La Scheda "Analysis"

Questa scheda elenca tutti i file di archivio delle simulazioni passate, mostrando data, nome dello scenario e durata. Da qui è possibile:
* `Refresh List`: Ricaricare l'elenco dei file dalla cartella di archivio.
* `Open Archive Folder`: Aprire la cartella contenente i file `.json` per un'ispezione manuale.
* `Analyze Selected`: Aprire la finestra di analisi per la sessione selezionata.

6.3.2. La Finestra di Analisi (`AnalysisWindow`)

Questa finestra fornisce una sintesi statistica e grafica dell'errore di tracciamento per un target specifico.

(Placeholder per l'immagine)

Descrizione dell'Immagine: `finestra_analisi.png`
Qui ti aspetti uno screenshot della `AnalysisWindow` con dati di esempio. Evidenzia:
1. *La tabella "Error Statistics", mostrando le righe per "Mean", "Std Dev" e "RMSE".*
2. *Il grafico "Error Over Time", mostrando le tre curve per gli errori X, Y, e Z.*
3. *Il selettore "Select Target ID" per scegliere quale target analizzare.*
4. *Le etichette "Avg. Latency" e "Prediction Offset" che mostrano i valori di temporizzazione.*

- **Selezione del Target:** È possibile analizzare un target alla volta selezionandolo dal menu a tendina.
- **Statistiche di Errore:** La tabella mostra le metriche chiave calcolate come `Errore = Posizione_Reale - Posizione_Simulata_Interpolata`.
    - **Mean (Errore Medio):** Indica una deriva o un bias sistematico. Un valore medio vicino a zero è desiderabile.
    - **Std Dev (Deviazione Standard):** Misura la variabilità o "jitter" dell'errore attorno alla sua media. Un valore basso indica un tracciamento stabile.

    - **RMSE (Root Mean Square Error):** Una misura complessiva della magnitudine dell'errore. È una delle metriche più importanti per valutare l'accuratezza generale.
- **Grafico dell'Errore nel Tempo:** Il grafico mostra l'andamento dell'errore istantaneo lungo gli assi X, Y e Z per tutta la durata del tracciamento. È utile per identificare in quali fasi della traiettoria (es. durante una virata) l'errore aumenta.
- **Metadati di Sincronizzazione:** La finestra mostra anche la latenza media stimata durante la sessione e l'eventuale `Prediction Offset` manuale che era stato applicato, fornendo un contesto cruciale per l'interpretazione dell'errore medio.

— 07_strumenti_di_debug.md —

Capitolo 7: Strumenti di Debug

Il Radar Target Simulator include strumenti avanzati progettati specificamente per il debug e la diagnostica a basso livello. Questi strumenti sono accessibili dal menu `Debug` della finestra principale.

7.1. SFP Packet Inspector (`SfpDebugWindow`)

Questa è la finestra di debug più potente dell'applicazione, dedicata all'ispezione e all'interazione con il protocollo SFP (Simple Fragmentation Protocol). Permette di monitorare il traffico di rete, inviare comandi manuali e ispezionare il contenuto dei payload ricevuti.

(Placeholder per l'immagine)

Descrizione dell'Immagine: `finestra_debug_sfp_ris.png`
Qui ti aspetti uno screenshot della finestra `SfpDebugWindow`. Deve avere la scheda "RIS" selezionata per mostrare le tabelle più importanti.
Evidenzia:
1. *La sezione superiore con i controlli di connessione e il "Simple Target Sender".*
2. *La scheda "Raw" con un dump esadecimale di un pacchetto.*
3. *La scheda "RIS" con le due tabelle: "Scenario" a sinistra e "Targets" a destra.*
4. *La scheda "History" con l'elenco dei pacchetti ricevuti.*

7.1.1. Panoramica della Finestra

La finestra è suddivisa in tre aree principali:

1. **Controlli Superiori:**
    - **Connessione:** Permette di stabilire o interrompere una connessione SFP indipendente da quella della finestra principale, utile per il debug isolato.
    - **Simple Target Sender:** Un'interfaccia rapida per inviare un singolo comando `tgtset` o un payload JSON per un target con parametri definiti al volo. È

estremamente utile per testare la risposta del radar a uno stimolo specifico senza caricare un intero scenario. I pulsanti nelle schede “CMD” e “JSON” permettono di inviare comandi rapidi come il reset.
    - **Script to send:** Un campo per inviare comandi testuali arbitrari.
2. **Pannello a Schede (Notebook):** Il cuore della finestra, permette di ispezionare i dati ricevuti sotto diverse forme.
3. **Barra di Stato (non mostrata):** Fornisce informazioni aggiuntive.

7.1.2. Interpretazione delle Schede

- **Scheda `Raw`:** Mostra un dump esadecimale completo dell’ultimo pacchetto SFP ricevuto. Visualizza l’header SFP decodificato campo per campo (inclusi flag, flow ID, TID) e il corpo del payload in formato esadecimale e ASCII. È lo strumento fondamentale per il debug a livello di protocollo.
- **Scheda `Log`:** Un’area di log dedicata agli eventi specifici della finestra di debug.
- **Scheda `RIS`: La scheda più importante per l’analisi dei dati reali.** Quando l’applicazione riceve un payload di stato dal sistema radar (tipicamente con Flow ID ‘R’), questa scheda lo decodifica e lo visualizza in due tabelle:
    - **Tabella `Scenario`:** Mostra i dati globali della piattaforma/ownship inviati dal radar (posizione, velocità, heading, altitudine, modalità operativa, etc.).
    - **Tabella `Targets`:** Elenca tutti i target (tipicamente fino a 32) e il loro stato come riportato dal radar. Per ogni target, visualizza flag, heading e posizione `(x, y, z)` in coordinate assolute.
- **Scheda `History`:** Tiene traccia degli ultimi pacchetti SFP ricevuti, mostrando timestamp, Flow ID, TID e dimensione. Selezionando una riga, il contenuto completo di quel pacchetto viene visualizzato nella scheda `Raw`, permettendo di analizzare pacchetti passati.
- **Schede `MFD`/`SAR`/`BIN`/`JSON`:** Queste schede sono “payload-aware”. Quando viene ricevuto un payload con il Flow ID corrispondente, il suo contenuto viene visualizzato in un formato appropriato:
    - `MFD`/`SAR`: Tentano di renderizzare il payload come un’immagine.
    - `BIN`: Mostrano un dump esadecimale.
    - `JSON`: Formattano e visualizzano il payload come un albero JSON.

7.2. Pannello Livelli di Log (`LoggerPanel`)

Accessibile da `Debug -> Logger Levels...`, questa finestra permette di controllare dinamicamente, a runtime, il livello di verbosità dei log per ogni singolo modulo dell’applicazione.

(Placeholder per l’immagine)

Descrizione dell'Immagine: `pannello_logger.png`
Qui ti aspetti uno screenshot della finestra `LoggerPanel`.
Evidenzia:
1. *La lista dei logger disponibili a sinistra.*
2. *Il campo "Filter" in alto a sinistra.*
3. *Il menu a tendina "Level" a destra per selezionare il livello (DEBUG, INFO, etc.).*
4. *Il pulsante "Apply".*

- **Scopo:** È uno strumento indispensabile per il debug. Se si sospetta un problema in un modulo specifico (es. `sfp_transport`), si può aumentare il suo livello di log a `DEBUG` per ottenere informazioni estremamente dettagliate, senza essere sommersi dai log degli altri moduli.
- **Funzionamento:**
    1. La lista a sinistra mostra tutti i logger registrati nell'applicazione.
    2. Si seleziona il logger di interesse.
    3. Si sceglie il livello di log desiderato dal menu a tendina (`DEBUG` per la massima verbosità, `INFO` per il normale funzionamento, `WARNING` o `ERROR` per visualizzare solo i problemi).
    4. Si preme `Apply`. La modifica ha effetto immediato.
- **Persistenza:** Le modifiche ai livelli di log vengono salvate in un file `logger_prefs.json`, in modo da essere mantenute tra un'esecuzione e l'altra dell'applicazione.

— 08_riferimenti_tecnici.md —

File: `08_riferimenti_tecnici.md`

Capitolo 8: Riferimenti Tecnici

Questo capitolo finale fornisce dettagli tecnici sui formati dei file utilizzati dall'applicazione e un glossario dei termini chiave.

8.1. Formato dei File di Dati

L'applicazione utilizza il formato JSON (JavaScript Object Notation) per la persistenza di tutte le sue configurazioni e dati. Questo formato è stato scelto per la sua leggibilità umana e per la facilità di parsing in Python.

8.1.1. File di Impostazioni (`settings.json`)

Questo file contiene tutte le impostazioni globali dell'applicazione, esclusi gli scenari. È suddiviso in sezioni:

- **`general`**: Contiene le impostazioni generali della GUI (es. geometria della finestra) e la configurazione della connessione.
    - **`connection`**: Un oggetto che contiene le sotto-sezioni `target` e `lru`, ciascuna delle quali definisce il tipo di protocollo (`sfp`, `tftp`, `serial`) e i relativi parametri.
- **`debug`**: Contiene flag e impostazioni per le funzionalità di debug (es. `save_tftp_scripts`).

Esempio di Struttura:

```
{
    "general": {
        "scan_limit": 60,
        "max_range": 100,
        "geometry": "1200x900+100+100",
        "last_selected_scenario": "Scen_Esempio",
        "connection": {
            "target": {
                "type": "sfp",
                "sfp": {
                    "ip": "127.0.0.1",
                    "port": 60003,
                    "local_port": 60002,
                    "use_json_protocol": true,
                    "prediction_offset_ms": 20.0
                },
                "tftp": { ... },
                "serial": { ... }
            },
            "lru": { ... }
        }
    },
    "debug": {
        "enable_io_trace": false
    }
}
```

8.1.2. File degli Scenari (`scenarios.json`)

Questo file contiene le definizioni di tutti gli scenari creati dall'utente. È un dizionario JSON in cui ogni chiave è il nome di uno scenario.

- **Ogni scenario** contiene a sua volta una lista di `targets`.
- **Ogni target** contiene il suo `target_id` e una lista di `trajectory` (i waypoint).
- **Ogni waypoint** è un oggetto che descrive una manovra con i suoi parametri specifici.

Esempio di Struttura:`json { "Scen_Virata_Semplice": { "name": "Scen_Virata_Semplice", "targets": [ { "target_id": 0, "active":`

```
true,                  "traceable": true,                  "use_spline": false,
"trajectory":
[                      {                          "maneuver_type": "Fly to
Point",                           "target_range_nm": 50.0,
"target_azimuth_deg": 45.0,                           "target_altitude_ft":
20000.0,                           "target_velocity_fps": 843.9,
"target_heading_deg": 270.0                       },                      {
"maneuver_type": "Dynamic Maneuver",                          "duration_s":
20.0,                          "lateral_acceleration_g": 2.0,
"turn_direction": "Right"                       }                  ]
}         ]     } }
```


8.1.3. File di Archivio Simulazione (`YYYYMMDD_HHMMSS_NomeScenario.json`)

Questi file, salvati nella cartella `archive_simulations/`, contengono un resoconto completo di una singola sessione di simulazione.

- **`metadata`**: Informazioni sulla sessione (timestamp, durata, latenza stimata, etc.).
- **`scenario_definition`**: Una copia dello scenario così com'era al momento dell'avvio.
- **`ownship_trajectory`**: Una serie di snapshot dello stato dell'ownship durante la simulazione.
- **`simulation_results`**: Il cuore dei dati. Un dizionario per `target_id` che contiene due liste:
  - **`simulated`**: La "ground truth" di tutti gli stati `(timestamp, x, y, z)` generati dal `SimulationEngine`.
  - **`real`**: Tutti gli stati `(timestamp, x, y, z)` ricevuti dal sistema radar per quel target.

8.2. Glossario

- **Azimuth:** L'angolo orizzontale di un target rispetto al Nord. Nell'applicazione, la convenzione è: 0° = Nord, angoli positivi in senso antiorario (CCW, verso Sinistra/Ovest), angoli negativi in senso orario (CW, verso Destra/Est).
- **Heading (Prua):** La direzione orizzontale in cui l'aereo (ownship o target) sta volando. Segue la stessa convenzione dell'azimuth.
- **North-Up:** Modalità di visualizzazione PPI in cui la griglia è fissa con il Nord (0°) in alto.
- **Heading-Up:** Modalità di visualizzazione PPI in cui la prua dell'ownship è sempre fissa verso l'alto e la griglia del mondo ruota.
- **Ownship:** La propria piattaforma (il nostro aereo) su cui il radar è installato.
- **PPI (Plan Position Indicator):** La classica visualizzazione radar circolare che mostra range e azimut dei target.
- **Simulation Frame:** Il sistema di coordinate cartesiane fisso, definito dallo stato dell'ownship a T=0, all'interno del quale si evolvono i target simulati.
- **SFP (Simple Fragmentation Protocol):** Protocollo di comunicazione su UDP usato per lo scambio di dati con il sistema radar.

- **Target:** Un oggetto (simulato o reale) tracciato dal sistema.
- **Trajectory:** La sequenza di manovre (waypoint) che definisce il percorso di un target simulato.
- **Waypoint:** Un singolo segmento di una traiettoria che definisce una manovra specifica (es. volo rettilineo, virata a ‘g’ costante).

```
```