DRDC September 2024 Update

Initial Simulation Results and Next Steps

Amos Hebb
a.hebb@mail.utoronto.ca

University of Toronto

2024-09-10

Summary

After defending thesis, focused on simulation development.

In this presentation we will:

- Discuss the system model.
- Discuss the simulation environment.
- Review current state of simulation.
- Discuss next steps.

System Model

- Queue management situation.
- We consider a **Environment** to be a *Theatre* and a *Platform*.
- An **Agent** must select a *task* from the *queue*.
- **Theatre** is one or more *targets*.
  - **Target** is an object to detect, kinematics, arrival rate, *etc.*
- **Platform** is one or more *sensors* and a *processor*.
  - **Sensor** Initially, we an S-Band Multi-Function Radar that can:
    - **search:** detect previously undetected targets, or
    - **track:** update known targets
  - **Processor** processes sensor data and updates the *queue*.
- **Queue** are a list of *tasks* the *Agent* may choose to execute.

Surveillance

- Surveillance region is quantized into a set of N fixed non-overlapping cells
- The radar uses a number of beams $\boldsymbol{A} = \{1, ..., A\}$, between which it must divide its time.
  - time allocated to a beam to perform a task denoted as τ_a
  - τ_a is fixed, for the S-band radar, at $\tau_a = 0.01s$
  - Simulation Arena: $([0, 200]\text{km}, [0, 200]\text{km}, [0, 20]\text{km})$
  - **Change** Region: Azimuth $[0°, 90°]$, Elevation $[0°, 30°]$
  - 15 uniform azimuth cells, 5 unifrom elevation cells.
  - **Change** No range resolution.
- **Change** We assume a scan duration ΔT of 1 second.
  - Initially we assumed $\Delta T = 1$ second
  - Surveillance tasks have a norminal rate of once per ΔT
  - Cost-free search coefficient $\alpha \in (0, 1]$.
  - Define $\Delta T := \frac{n(\boldsymbol{A}) \times \tau_a}{\alpha} = \frac{75 \times 0.01s}{0.5} = 1.5s$

Tracking

- Target motion per Singer trajectory model, we have four targets.
  1. Linear: $\sigma^2 = 0$
  2. Whiplash: $\sigma^2 \sim U\left[20\frac{m}{s^2}, 35\frac{m}{s^2}\right], \Theta \sim U[10s, 20s]$
  3. Zigzager: $\sigma^2 \sim U\left[0\frac{m}{s^2}, 5\frac{m}{s^2}\right], \Theta \sim U[1s, 4s]$
  4. Sweeper: $\sigma^2 \sim U\left[5\frac{m}{s^2}, 20\frac{m}{s^2}\right], \Theta \sim U[30s, 50s]$
- Targets are generated at random locations and velocities.
- In earlier work I've found uniform generated targets make evaluation difficult.
  - Developed a scenario specification format.

Tracking Tasks

- Revisit time t_r
  - $t_r = 0.4\left(\frac{R_t \sigma_\theta \sqrt{\Theta}}{\Sigma}\right)^{0.4} \times \frac{U^{2.4}}{1+\frac{U^2}{2}}$
  - R_t is target range.
  - Θ, Σ are Singer model parameters.
  - We use $U = 0.3$
  - Once a task has terminated, set $t_{\text{start}} := t_r$
- Dwell time
  - $\tau_c = \tau_c^n \times \left(\frac{R_T}{R_0}\right)^4 \times \text{SN}_0$
  - $t_c^n = 0.01s, R_T$ from target, $R_0 = 184\text{km}$

Gym

- **Gymnasium** (Gym) is a toolkit for developing and comparing reinforcement learning algorithms.
- *Game loop*: Time advances until a function releases a sensor, then the agent selects the next function to execute.
- Environment, theatre and platform.
  - Theatre is designed to support pre-defined scenarios.
  - Platform is designed to support multiple sensors.
- Observations: The queue.
  - Current idle sensor index as one-hot vector.
  - Single search task, t_{start}, t_{dwell}, $[t_{\text{elapsed}} \ldots]$
  - Each possible track task, t_{start}, t_{dwell}, priority
- Actions: Task index. (Executing a 200ms scheduling frame requires memory)
- Step: Execute the selected task, calculate reward, update the queue, update the theatre.

JAX

- JAX is a Python library for numerical computing
- Differentiable rigid body physics engine (BRAX) create and propagate many targets in 3D.
  - Most of the motion and geometry is included in BRAX.
- Run the simulation in parallel on GPU, much faster.

Demo

Demo the ESTesque simulation.

`demo_est.py`:

- Create a theatre and platform.
  - Theatre has 10 targets, all visible, 5 tracked.
  - Platform has 1 sensor.
- Define the scheduler.

`est_scheduler.py`:

- On each step:
  - If buffer has > 0 tasks, execute the first task.
  - Else: While the sum of dwell times is less than 200ms:
    - If all $t_{\text{start}} \geq \tau_a$: a = 0, add a search to buffer.
    - Else: $a = \min_{\{a_i \in \boldsymbol{A}\}}(t_{\text{start}})$, add task earliest start time to buffer.

Next Steps

- Implement a multi-sensor platform.
  - Observaiton already supports multiple sensors.
  - Platform definition will need to be updated to support generalized coordinates.
- Implement a multi-sensor EST.
  - Similar, creates two buffers and re-populates both when either is empty.