Pfizer/DRL AI Challenge

1. General

The purpose of this document is to outline the framework, the methods, and the goals of the Pfizer AI challenge. The document will cover the necessary tools, suggested flow, as well as the scoring mechanism to determine the winners.

2. Framework

The challenge presented to the participants is to create a mechanism of autonomously tracking a drone in a video taken within the DRL racing simulator.
Contestants will be provided with a practice video of a flight on a game map, practice input timestamps, and practice ground truth data in CSV format on which to train/refine their code.
The contestants will submit the results of their development in the form of a zip file and will be scored on the best performance of their code applied to a “scoring” video that captures a different flight on the same game map.

3. Methods and Tools

Contestants must develop code using Python 3.12.
Code must be able to run on Ubuntu 22.04 64bit.
The code will need to be able to accept a video filename as input, and a csv file that includes a series of timestamps in millisecond precision. The code will output, for each timestamp, the timestamp itself and the x and y coordinates of the drone on the video in a CSV format - one line per timestamp. For the purpose of this challenge (0,0) coordinate will be considered the upper left corner of the video.

Example command line:

```
python tracker.py flight1.mp4 input.csv
```

Example input.csv:

```
Ts
000001094
000001111
000001127
000001143
000001160
```

Example output.csv generated by the call to the command line above:

```
Ts, x, y
000001094,150,220
000001111,152,223
000001127,157,230
000001143,163,245
000001160,159,260
```

As an additional resource for practicing/training, the contestants will be provided with a ground-truth telemetry that will match the expected output of the provided training inputs.

Some tools to consider (depending on the chosen approach):

- Virtual environment - you are required to submit your work with requirements.txt file that allows us to recreate your python environment
- OpenCV - swiss army knife toolkit for computer vision
- TensorFlow - extensive machine learning toolbox

4. Flow

Recommended development flow is as follows:

- Split the video into several segments, make it easier for you to test your code on parts you haven't seen/trained on
- Make sure that you are able to open the video from your code and process it frame by frame (try outputting a frame to see that it appears as expected)
- Make sure to include error checking on the inputs and present clear error messages when failure occurs
- Practice extracting frames based on timestamps
- Modularize your core processing algorithm such that you can call it as a black box

- Once you have a working algorithm that performs well on your initial segment - test it on the video segments you have kept aside for testing. Try to make sure the test segments are somewhat different from the segment you used for practice/training - this will give you a good sense about what to expect during scoring.

5. Submission

- When you are ready to submit your work - make sure the dependencies are frozen and are all inclusive inside requirements.txt file
- Create a clean environment from your frozen dependencies and test your code one last time to make sure it works when scoring is done.
- Make sure to include a readme.txt file explaining installation and execution and including your personal details - first and last name, email address at the top of the file
- Compress your python code (tracker.py), requirements.txt, and readme.txt using a zip archive and submit the resulting zip file.
- There is no need to include the practice video, practice timestamps, practice telemetry, or your output.csv

6. Scoring

- We will run contestants' code on our "scoring" video with a set of scoring timestamps
- The scoring will be calculated similar to the following: $score = \sum_{i=0}^{n} \sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2}$
  Where **n** is the number of timestamps in the input file we will use for scoring, (x_i, y_i) being our "scoring" ground truth coordinates, $(\hat{x}_i, \hat{y}_i)$ being the contestant's output coordinates for respective timestamps
- In other words - winner will be the person who will output drone location that is closest to the ground truth telemetry ($score$ value will be lowest among the contestants)