**Match Simulation System Review**

**System Overview**

MSD 2020 built a match simulation system in Simulink for the purpose of generating simulated game states to be used in verification / validation of their implemented AutoRef code. The system is comprised of the following sub-modules:

* A **companion MATLAB script** for variable initialization (player X0, V0, θ0; ball X0, V0, θ0; camera X0, orientation; simulation settings)
* A **Simulink model** which integrates object positions over time from initial positions and velocities and outputs object trajectories over the entire simulation time.
* A **virtual model file** .WRL which renders the simulated match based on the computed object trajectories.

The simulation system generates video and / or image files which are then used by the MSD 2020 AutoRef system software to analyze the game and detect violations.

**System Interpretation**

**Simulink Model:**

The model executes the basic function of simulation sufficiently, but there is a lot of room for improvement. In the “Final game simulation” folder’s Simulink model, object trajectories are hard-coded into individual scripts which dictate each object’s trajectory based on conditional statements over time. In other words, Player 1 starts with an initialized position and velocity generated by the **companion MATLAB script**, and then Player 1’s trajectory script applies linear operations to compute position for each time step. Here’s an example for one player’s y coordinate computation:

function y = fcn(u)

%u = out.simout.time;

if u <= 5

y = -5.5;

else u > 5 && u <= 20

y = 0.2 \* u - 6.5;

end

end

This means that for each desired gameplay simulation, each object’s trajectory computation needs to be hard-coded. This is obviously an extremely manual process but given the time constraints of the project it was likely the only feasible option for the simulation thrust.

**Suggested Improvements**

1. Implement multiple camera viewpoints in the simulation environment and develop a workflow to perform multi-view image fusion for improved monitoring perspective. This could allow us to predict an optimal camera configuration (number and position) while minimizing the number of cameras required. It would be a good technical challenge and the feasibility seems reasonable.
2. Develop (or obtain) real match video data instead of using the simulation environment for AutoRef system testing, and adapt the previously developed AutoRef software for use on real game footage. The technical challenge would be good, however there may be difficulty in obtaining sufficient real footage for enough game scenarios. This is a question that we could probably answer quickly in an interview with Tech United.
3. Create a hybrid model-based and real image-based simulation system where real camera images of the field, players, and ball are segmented and applied as objects in the simulation environment. This could allow us to use the flexibility of the simulation environment and apply it to real images for testing. The technical challenge here may be the lowest compared to other options, and maybe this is something we could implement in addition to another improvement option without much effort.