Robotic Drone Referee Project- 2016 (MSD Batch 2015-17):

Scope: Ball out of play.

Used MATLAB

Proposed Idea: Multiple

Multi-sensor/drone agent solution

Ultra Wideband System: Accuracy increases when the number of UWBS tags is increased and used for drone positioning.

Refereeing out of Pitch:-

Enabled when conditions are met:

1. Drone position is known
2. Drone is inside the pitch
3. The ball has been detected
4. The line has been detected, filtered, and matched with the outer model line references.

1 and 2 are "Localisation", 3 is "Ball-detection", and 4 is "Line-detection and field line predictor".

Field-line predictor:-

- Altitude of drone (from drone IMU)
- Drone position (from localization block)
- Camera vision angel (predefined)
- Yaw angle of the drone (sensor fusion or IMU block)

Hough transform to detect the lines

Ball-detection:-

Used color detection. Removing colors that do not have a predefined color threshold. Then a circular shape matching.

- "Colour Thresholder" app from MATLAB
- Hough Transform for circular shape detection

Line-detection:-

Hough transform is used. Steps:-

1. Undistort the images.
    a. "Camera calibration" app from MATLAB
    b. Black and White boxes are used to calibrate (chess board)
2. A colour mask to get only the lines on the frame
    a. Used similar technique as in "ball-detection"
3. Line detection
    a. Hough Transform
    b. MATLAB

World Model:-

- Trilateration using UWBS
- Used three UWBS to know the position of the drone. One beacon on the drone too
- Accuracy is 10cm

- Used TREK100 kit from Decawave.

Integration Strategy:-

- Worked with Simulink and MATLAB
- Difficulty with integrating multiple subsystems in Simulink.
- Difficulty in getting drone feed to MATLAB -> R-pi with a camera used to stream to Matlab
- Magnetometer freezing, resulted in problems with yaw angle estimation
- Had to use two computers, one to run the refereeing demo and the other to run the drone controller. Because the latter needs a Wifi connection with the drone and the former functions are connected to R-pi.

Tests and Discussion:-

The conclusions of the test (demo) are the following:

- The localization method based on trilateration is very robust and accurate but it should be tested in bigger fields.
- The color-based detection works well with no players or other objects in the field. Further testing including robots and taking into account occlusion should be considered.
- The out-of-pitch refereeing is very sensitive to the psi/yaw angle of the drone and the accuracy of the references provided by the Field Line estimator.
- Improvement in distinguishing between close parallel lines should be continued.
- Increasing localization accuracy and refining the Field Line estimator should be researched.

Comments from Anshid:

- Lane detection and the world view model of the techunited can be used if the cameras are stationary.
- Faced difficulties during integration. So, plan with integration in mind.
- The movement of the drones needs to be precise, the sudden up and down movements can cause irregularities in the measurements.
- Integration is important. We have to think about this before starting the project. How are we going to integrate the subsystems into one system?

Autonomous Referee System 2017 (MSD Batch 2016-18):

Scope:-

- Detect ball out of play
- Detect collision

System Architecture:-

- Used Paradigm used by KU Leuven in collaboration with TU Eindhoven. (Link)
- Used a layered approach. From top to bottom, each subsequent layer the level of detail increases.
- Image processing algorithms, sensor fusion algorithms for world model, and communication protocol for communicating between multiple hardware.
- Simulink was used
- Skill-matcher
    - The skill selector needs to determine which piece of hardware can perform this skill.
    - Hardware configuration files for every hardware.
- Supervisor:
    - Responsible for monitoring the system tasks and for coordinating the subsystems concerning these tasks.

Implementation:-

Used image processing toolbox in Matlab

1. **Line Detection:** Hough transform. Updated the MSD2015 algorithms. Separated line detection code from all detection codes
2. **Detect balls:** Colour filter to identify balls, assumed the ball colour is red, orange or yellow. (upper corner of CbCr-plane). Blob detection to identify the ball. Confidence is calculated based on the blob size and roundness.
3. **Detect Objects:** Object or player detection is similar to Ball detection. Instead of colour filtering in CbCr plane, it's in Y-axis. Assumptions is that the players are coated in black fabric. Bigger blobs than the ball detection.

Ball out of Pitch: Used the algorithm developed by MSD2015. Added a particle filter to predict the position of the ball.

Collision Detection: Used the world model and raw images. Track position and velocity to predict whether they are colliding and use space between two blobs. Implemented the latter one.

Location of the Drone: Used altimeter in the drone. The camera yields the planar position w.r.t the field.

Locating the Ball and the Player:

Assumptions or the pricniples followed:-

- The center of the image is assumed to be focal center of the camera and this is coincident with the center of the image.
- The camera is always parallel to the ground plane, neglecting the tilting of the drone on drone's roll (ϕ) and pitch (θ) axes.
- The camera is aligned and fixed in a way that, the narrow edge of the image is parallel to the y-axis of the drone as shown in the figure below.

- The distance from the center of the gravity of the drone (which is its origin) to the camera lies along the x-axis of the drone and known.
- The height of the camera with respect to the drone base is zero.
- The alignment between camera, drone and the field (shown in a figure.)

The height of the camera and the FOV information of the camera are used to calculate the pixel-to-millimeter conversion.

Path Planning: For the drones.

World Model: Constantly updating the model using sensor fusion. Also, acts as a storage unit, saving the last known positions of multiple objects.

Ball position filter and sensor fusion: Used Monte Carlo Localisation. The reason is that it can handle multiple object tracking. The filter perfomrs three tasks:

1. Predict ball position based on previous measurement
2. Adequately deal with sudden change in direction
3. Filter out measurement noise

Task 1 and 2 are conflicting and 1 and 3 are closely related. 1 is using a strong filter and 2 is using a weak filter.

System Identification and dynamic modelling of the Drone: Known inputs are provided and the response is measured. The relations between input and output will give system parameters. Used "System Identification Toolbox" in Matlab.

Hardware:

1. **Drone:** AR Parrot Drone Elite Edition. Controlled by a mobile phone (android or ios).
2. **Camera:** the camera of the drone was not properly able to transmit data to Matlab.
3. **Top Camera:** wide angle camera fixed above the playing field to measure the location and orientation of the drone. Used "GigeCam" toolbox in Matlab.
4. **Wifi-Webcam:** Since the camera of the drone was unable to broadcast to Matlab, a wifi based camera is used. Calibrated using Checker-board.

Drone Motion Control: High Level Controller calculates the speed in global coordinate system and sends it to a Low Level Controller. HLC designed and LLC is already implemented in the drone.

Conclusions: Suggested to use TechUnited turtle or Omni-bot with a Kinect as a line referee.

Comments from Anshid:

1. Hardware and software communication was not analysed initially and had to change the plan in the middle due to this.
2. A camera on top of the arena is used to calculate the position of the drone. Can this camera system be used instead of drone?

Drone Referee 2018 (MSD Batch 2017-19):

Scope:

- Free throw, ball is out of the bounds of the pitch
- Robot players are touching each other. Foul detection (Collision Detection)k

System Architecture:

- Follows "CAFCR: A Multi-view Method for Embedded Systems Architecting. Balancing Genericity and Specificity"

Hardware:

- **Drones:** developed by Peter Rooijaker (master student from TU/e) as his master thesis project. The drone consists of a flight controller called Pixhawk PX4, motors, a fish eye camera, lidar, Raspberry Pi as an onboard processor for processing vision, and an indoor GPS beacon. Telemetry was used to communicate between the drone and the off-board PC
- **Indoor GPS System:** The localization of the drone was done by installing an Indoor GPS system by placing ultrasound beacons around the field. One of these beacons was placed on the drone which gave the drone coordinates.
- **Dev-PC:** Used a Linux machine that can connect to the TechUnited network to access the world.
- **Referee Laptop:** A Laptop where a remote referee is monitoring the game and has an HMI to give his decision based on the recommendation and video feed.

World Model:

- **Games States:** Stores the state of the game.
- **Drone coordinates:** stores the coordinates of the drone obtained from the indoor GPS and lidar.
- **Ball and Robot Coordinates:** Ball coordinates were obtained using fish eye camera and R-pi. Robot coordinates from TechUnited World Model.

Flight and Control:

- Lidar for Heigh
- Marvelmind Localisation for X and Y coordinates. GPS indoor sensor.
- Autonomous Flight: Actions taken by Pixhawk controller. Receives Mavelink messages which are generated in Simulink.
- Path Planning: Assumption is that the drone stays in the same altitude during the game and ball is always stays on the ground.

Image Processing:

- MATLAB
    - FFmpeg
    - Gigecam
    - Hebicam
- C++/python/Java
    - OpenCV

C++/OpenCV is used to reduce the detection time. ARUCO markers are used to get the "calibration matrix" and "distortion coefficients" (OpenCV: Fisheye camera model).

Collision Detection:

Two methods used to track the robot position

- **ArUco Markers:** Augmented Reality library based on OpenCV. Two drawbacks of the system: the strategy used is based on trajectory planning with ball visibility at all time, this means collisions happened out of the view are not detected. The second drawback is the camera and players are moving all the time, at a high relative speed, and the camera is unable to detect AR tags or detect them with high latency.
- **Techunited world model:** Used the world model broadcasted by the turtles. Position between two robots and the relative velocity between them are used to detect collision. Sign of a velocity tag is used to identify the guilty player.

GUI: Created two GUIs

- Remote Referee interface (1st GUI)
    - Collects information from the Simulink model and processes it for a human referee
    - Control the Simulink model remotely
- Audience interface (2nd GUI)
    - Controlled by 1st GUI to change the LED colors depends on the referee's decision

Conclusions:

- Intel NUC is replaced by R-pi and a telemetry connection
- Latency of drone calculation in detection -> started using TechUnited turtle data
- The telemetry interface has latency issues, a lower update rate than Pixhawk connected directly to the on-board NUC. UDP showed higher update rate frequency and reliability but connections are completely secure.
- High latency in the video streaming of the fish-eye camera using R-pi -> solved by using IP camera but is not implemented
- Simulation in path planning.
- Supervisory control: CIF is a better tool compared to Matlab state flow. CIF tool implemented in Simulink using S-function C-code.
- The propeller airflow of the drone causes the ball to mode, so use a more compact drone to tackle this.

Codes and resources are updated in repositories and shared at the end.

Comments from Anshid:

1. Collision guilty player detection is only based the relative speed between the two players and not depend on the playing position.
2. Drone path planning is simple 2D, ignoring the vertical movement of the drone. Also, ball is assumed will not move in vertical direction.
3. Latency in transmitting the data.
4. Detection speed is slow in Matlab

Drone Referee 2019 (MSD Batch 2018-20):

Scope:

Their main focus was on drone control rather than implementing any rules.

Drone: Avular Curiosity by Avular.

The drone can be programmed into three computing modules: Low-Level module, high-level module, and compute module. High-Level and Low-Level Modules use a 180MHz Cortex-M4 processor and the Compute Module is a Raspberry Pi Compute Module 3. Low-Level module for calculations related to flight stabilization and monitoring the current state of the drone. High-level model for high-level calculations such as path planning or position control of the drone. Compute module in combination with High-Level and Low-Level modules for intrinsic computations.

Communication is through X-bee and MAVLink protocol.

Human Machine Interface (HMI):

Programmed in C# which communicates with the world model running in MATLAB. The functions are:

- To display a video stream from the camera
- Provide a replay of the last 30 seconds as and when needed
- Send take-off, clear-to-land and land commands to the drone
- Display recommendations such as goal kick, corner and throw in
- Display the state of the game

Ball Localisation: Used the existing protocol for the teams. The teams communicate with the referee base station. The data include ball position, ball velocity, robot id and robot position.

Drone Localisation: Implemented using Avular Ultra Wide Band System.

Comments from Anshid:

1. They worked mainly on controlling the drone and feeding video feed to the referee.
2. For detection of ball and robot they are using the data transmitted by the turtle bots to the referee base station.

Drone Referee 2019 (MSD Batch 2019-21):

Scope of the Project	The goal of the project is to assist the main referee by capturing important game situations by using drones. The drone was used to provide images to a remote referee, who decided based on images and repetitions. The remote referee can then inform the drone or another onsite referee system about decisions (which in turn are communicated to the teams).
System Architecture	Requirements, Contexts and Problems
Solution Space	Computer vision, Deep Neural Networks Navigation: Path planning in 1D and 2D Communication: Ultrawideband, Radio
Operation Space	ROS, Crazyfile Open CV, Python Libraries Virtual Simulation Model tests, hardware tests, software debugging
	Visualizer and simulator
Rules	Their main focus was also on drone control instead of implementing any rule. Rule (Assumption) ball remains on the ground (ball in play) $Z_{ball} = 0$

System Decomposition:

1- Carzy File 2.0 Drone Platform : Low Level Drone Control
   - Kalman Filter (Estimates the position of drone based on Sensor Measurements)
   - Motion controller (Generates outputs into motor drivers0
   - Sensor inputs (Drone Pose coordinates)

2- Action Planner : High Level Drone Control
   - Generates kinematic parameters provided to drone and into visualizer.
   - Path Planner
   - Tracking control

3- Perception System
   - Streams and process Images for an HMI
   - Image processing (Transformation, HSV Color mapping)
   - Object detection (MATLAB toolboxes and OpenCV)

4- HMI
   - Field of view
   - Streams live footage to viewer.

5- Implementation
   - Manual Flight of Drone (Windows to setup the software part of manual flight)
   - Autonomous Flight (Ubuntu 16.04)
   - Hardware required (Bitzraze Crazyradio PA USB dongle)
   - A remote control
   - Autonomous Flight (Loop positioning system, Ubuntu 16.04, Python Scripts)

Following modifications were made :

a. Ball position extraction from visualizer data.

b. Development of a path planning algorithm that takes the visualizer data and converts it to drone position data.
c. Drone position data extraction and sending it to Visualizer for visual representation.

6- Localization

- The drone is equipped with an IMU sensor. Different options were considered before implementation
    1- Camera and markers
    2- Ultra wideband positioning
    3- Optical motion detection
    4- Combination of 2 and 3

7- Path Planning

- Algorithm 1: Drones moving parallel to the sidelines.
    - Ball remains on the ground
    - 2 drones are used outside the field moving parallel to the sidelines with yaw movements.
    - Ball should always be at the center of the camera views.

- Algorithm 2: Drones moving in a circle.
    - Objective: To find the shortest path for the drone while keeping the ball in the field of view.

    - The altitude, z_{drone} of the drone is same throughout the duration of the game.
    - The ball is always on the ground, i.e., $z_{ball} = 0$.
    - The drone encounters no obstacles at altitude, z_{drone}

8- Visualization

- Only Streaming video
- FPV camera (Hardware) Wolfwhoop WT05FPV
- Skydroid 5.7 dual receiver
- A bright orange ball was used for high contrast.
- Filters applied to the image (erosion and dilation, blob detection and taking centroid)

9- Integration

- Extract the ball location from previous recorded game to (.csv file) and translate that into drone setpoint.
- Integration of Drone , visualizer and path planning sub-system.
- Visualizer (MATLAB) ,Drone (Python)
- Drone writes a (.csv file) that contains the drone actual location and visualizer reads the file as visualizer camera position.
- Static vision system was used to get the ball position. This static camera feeds the ball location into path planning algorithm that will generate the drone set point.

10- Conclusion

- The ball position relative to the field has been determined using computer vision.
- Difficulty in setting up drone

- Invest more in drone hardware .
- More efforts should be focused on the computer vision.
- The project scope should be precise.
- The selection and integration of hardware issues of drone, owing to safety issues, the limitation was to use a small FPV camera because of latency issues with other options.

Drone Referee 2022 (MSD Batch 2022-24):

Project Scope:

To design an algorithm to check set piece such as corner kicks

- Using stadium cameras to achieve the project goal.
- Scope of the project: The objective is to design the algorithm for checking set pieces including situations like corner kicks.

- **Rules:**
    - Corner kicks for validation of the algorithm
- **Method and Procedure**
    - Using one of the Security cameras, a corner kick procedure was used to validate the developed algorithm.
- Object detection algorithm to detect the ball and players.
- Birds eye transformation will be applied to calculate the distance between players.
- By using the distances, the corner kick procedure was checked and a signal depending on the player's position sent to the referee.
- YOLO: Deep learning architecture used for object detection.
- **Rules for Corner Kick:**
    - Referee gives the signal
    - Robot of the attacking team takes the kick position
    - Attacking team players at least 2m away ball radius
    - Defending team players at least 3m away ball radius
- Image collection of the ball and players
- Bird’s eye view transformation
- Video recording through one of the security cameras. 300 frames are stored to be used as a data set.
- Training of the machine learning model and call for predictions.
- After predictions, implement corner kick procedure in algorithm.