Walking

The main process of our robot's walking control is an integration of LIP-Based MPC and QP-Based WBC.

Firstly, the algorithm estimates the current state of the robot based on sensor data, such as position, velocity, and attitude. Using these data and the motion command as an input, the model analyses the corresponding action to control.

Using the LIP model, the algorithm predicts the future motion of the robot and calculates optimized basic control inputs to maintain balance. Additionally, the QP-Based algorithm considers task requirements and various constraints, optimizes the entire body movement through quadratic programming, ensuring that the robot maintains overall stability while executing tasks.

Through the combination of these two models, the final control input is sent to the robot's actuator, driving the robot to perform optimized actions.

Vision

The software structure now is based on the ROS framework, which greatly increases the efficiency of multitasking coordination and simplifies the process of application of a new algorithm.

There are four modules of the algorithm of our robot, vision is one part of them. Vision Module is used to recognize feature objects in the football court and get their position relative to the robot. We use ZED 2i Stereo Camera, a binocular camera, and our vision algorithm is based on computer vision (OpenCV).

YOLOv3 is used in our algorithm to recognize the ball, goal and sidelines. In our last few competitions, the code works well after training with dataset from a monocular camera. Since we have upgraded our vision sensor to a binocular camera, adjustment and validation is necessary. Meanwhile, the distance and direction of objects can be easily calculated.

Localization

Self-localization is a state estimation problem. The robot needs to estimate its position and orientation from the data of its sensors, mostly the camera and IMU. We choose the widely used particle filter algorithm to solve this problem.

The structure of this algorithm is made of Map input, Initialization, Prediction, Updating the weight of Particles and Re-sampling. The process of map input is done in advance by giving standard playing field map. The algorithm of the initialization of particles is also important. Gauss noise is added to generate many particles. We design different algorithms for different situations, such as initialization at the beginning of the match or initialization after picking up. The stage of prediction incorporates the states of particles with data from the IMU, mostly orientation. During the updating, we first incorporate the data from the camera and IMU, then increase the weight of particles whose observation target most close to real feature objects. At resampling, to prevent kidnapped particles, we chose half particles based on updated weight, and generate the other half by adding noise on the chosen particles. The particle with the highest weight among these particles represents the position of the robot.

We have changed our camera from a monocular camera to binocular camera. It provides depth information, and will increase the accuracy of localization algorithm. We plan to accomplish this process in the coming spring semester.

Behavior

Building upon the previous hierarchical state machine (HSM) framework, we have transitioned to employing behavior trees (BT) to enhance the clarity and structure of our decision-making processes. Within our algorithms, behavior trees are integrated to govern the behaviors of our soccer robot. The algorithm framework exhibits a hierarchical structure, with the behavior tree comprising multiple nodes, each representing distinct behaviors the robot can demonstrate during a soccer match. Notably, we have established super-nodes including 'defending,' 'attacking,' 'passing,' and 'dribbling,' with each nesting sub-nodes representing specific actions such as 'tracking the ball,' 'positioning for a shot,' 'passing to a teammate,' and 'dribbling past opponents.'

By leveraging the behavior tree, we can systematically organize and prioritize the robot's actions based on the prevailing game dynamics and our strategic objectives. By defining the tree's structure and outlining conditions for transitioning between nodes, we can develop a flexible and adaptable logic system for our soccer robot. The hierarchical nature of the behavior tree facilitates seamless addition, modification, or removal of specific behaviors, preserving the overall structure of the tree. This modular and scalable aspect proves advantageous when refining and advancing the robot's behavior over time.

The behavior tree approach facilitates a structured and user-friendly methodology for designing our soccer robot's logic. It enables the amalgamation of simple actions to define complex behaviors and offers customization and evolution flexibility. With the behavior tree, our soccer robot can make astute decisions and display dynamic behavior on the field. Although we maintain an interest in multi-agent reinforcement learning for behavior, the deployment timeline for this method on our robots is currently undetermined.

Furthermore, in addition to the aforementioned actions, our robots engage in simultaneous communication via the Robot Operating System (ROS), exchanging field information such as the ball and player locations.