Software

Imaging

To enhance the resilience of our vision system against the challenges posed by varying sunlight conditions, we have initiated efforts to optimize imaging results through the adjustment of exposure across multiple levels. This strategy aims to mitigate the issues of overexposure and underexposure that often arise despite the implementation of embedded auto-exposure functionality within the camera's driver. However, this automated adjustment can inadvertently lead to fluctuations in exposure levels, complicating the detection of critical elements such as the ball, goals, and the field itself. Consequently, we have opted to manually set specific exposure levels, fine-tuning them based on the immediate outcomes observed in the imaging process.

Our approach involves analyzing the image histogram—a graphical representation of the distribution of pixel brightness. If the histogram demonstrates a significant skew towards the left (indicating underexposure) or the right (suggesting overexposure), we then adjust the camera's exposure to either the next or previous level, respectively. This method allows us to more precisely control the imaging quality in real-time, adapting to the dynamic lighting conditions encountered during play.

Given that lighting conditions tend to remain relatively stable for periods ranging from seconds to minutes, continuously calculating the histogram within every cycle of the program would be excessively time-consuming and inefficient. To optimize the balance between computational efficiency and the effectiveness of our imaging adjustments, we prioritize the evaluation of the cognition results. This preliminary assessment determines whether a histogram calculation is necessary. By implementing this selective approach, we aim to conserve processing resources while still achieving optimal exposure settings, ensuring that our vision system remains both accurate and responsive under a wide array of lighting conditions.

Cognition

Cognition is a necessary part of the robots to perceive the surroundings. Our detection for the field and the lines are based on traditional computer vision techniques like hough transform, canny or simply using thresholds to get areas. Deep learning method is used for the ball, goalposts and robots. Until last year, we are using YOLOv1-tiny, which is fast enough to run on our previous hardware TX1. As our hardware is updated to TX2, we change our algorithm to a newer version YOLOv3-tiny and change the implementation to TensorRT. Compared to YOLOv1-tiny, YOLOv3-tiny runs a little slower, but has higher accuracy,especially for small objects, like a ball 10 meters away. For higher inference performance, we choose TensorRT to optimize our network model. NVIDIA TensorRT is a platform for high-performance deep learning inference. It can apply optimization to trained models and selects platform specific kernels to maximize performance on NVIDIA GPUs.

Besides, we undertake to recognize the corner using its rich line features. When the robots are around the field corner, where they can see almost nothing except the corner lines. Such additional reliable landmarks

will improve the localization. The location is given by the voting result of the white lines. With stable image, this method can locate the corner quickly and accurately.

Navigation

Our navigation strategy employs Adaptive Monte Carlo Localization (AMCL), utilizing a diverse array of landmarks such as circles, lines, goals, and corners. We are actively working to incorporate additional features within the playing field to enhance the markers available for AMCL, aiming to enrich the environmental cues for more accurate localization.

One challenge we've encountered is the instability in the robot's pose, particularly when walking. This instability introduces significant noise in orientation measurements, making it difficult for the particle filter to converge accurately on the robot's position. To mitigate this, we rely on an Inertial Measurement Unit (IMU) to provide reliable direction information. Consequently, our algorithm focuses solely on estimating the robot's (x, y) position on the field, leveraging the IMU for orientation data.

The process of updating particles is driven by odometry data, which is then refined based on the landmarks detected by our cognitive systems. More specifically, after obtaining the recognition results of the corner points, we project them onto the plane of the soccer field, thereby determining the position of the corner points within the robot's coordinate system. Then, this position is used as landmark information for the particle filter to update the position of the particles. This year, thanks to improvements in our odometry's reliability, we've been able to decrease the noise parameters within our localization system. This adjustment has made our localization results significantly more stable and effective, demonstrating the positive impact of our enhancements on the system's overall accuracy.

Behavior

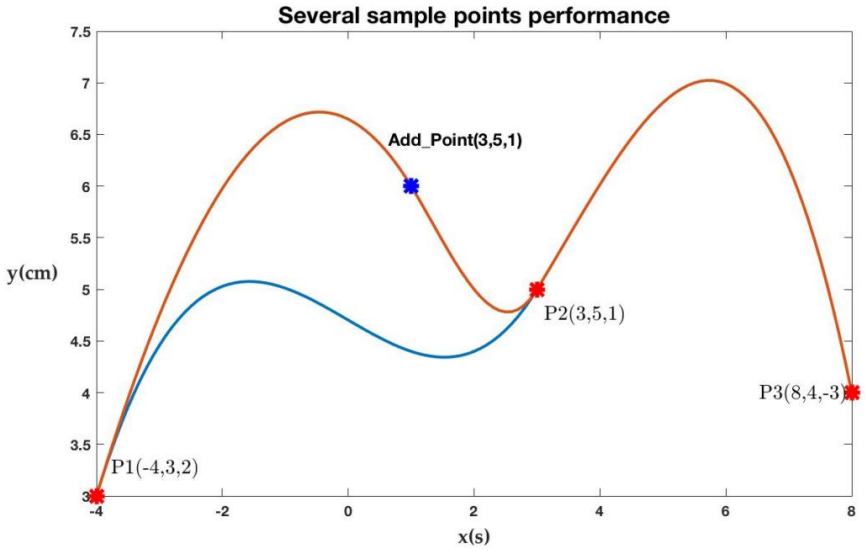
Currently, our behaviour module is based on a mixture of behaviour tree and finite state machine, implemented in Python. It reads information from the localization module and game controller module, and plans action command in each main loop. As the localization is not satisfying enough for complicate moving strategies, we choose decentralized decision making. Thus, our robot scan use relative positions to make a decision which can avoid the impact of their wrong self-localization.
Last year, our attacking robots all using a same strategy and no matter where other teammates are. As a result, they often collide when two robots walking toward the ball and finally both of them lost the ball. Thus, we design cooperation skills to avoid collision this year.The most important rule is that one robot will control the ball(which means go to kick the ball) until it kicks or falls to avoid continuous status changing caused by uncertain information. When one robot is controlling the ball(we call it kicker), the other attacker will assist it(we call it assistant). More specifically, it will stand and wait at a suitable point. When the kicker kicks or falls, the assistant will soon go ahead to control the ball and become the kicker.Also, there is some exception for example, when the assistant is much closer to the ball than the kicker. they will switch their roles. Such dynamic role changing greatly improves offense efficiency.As mentioned above, we are using relative positions for decision making. We used to decide the kicking direction based on the self-localization result and real goal position. However, as we can accurately see the goal, a better way is deciding the target according to the goal position seen by the robot. It ensures that the robot is kicking to a goal rather than a fixed point.

Motion

The motion module of our robot is a sophisticated system designed to ensure precise movement and stability. It is divided into four integral child modules, each specializing in a different aspect of motion control. These modules work in concert to enable the robot to interact with its environment efficiently and effectively.
1. IO Module: The Input/Output (IO) module plays a crucial role in managing external devices, including servos, the Inertial Measurement Unit (IMU), and foot pressure sensors. It acts as a bridge between the robot's central processing unit and its physical components. The IO module facilitates communication by sending instructions to actuators, such as directing servo movements, and by collecting data from sensors, such as orientation from the IMU or pressure data from the feet. This real-time data exchange is essential for dynamic balance and precise control of the robot's movements.
2. Kinematics Module: The Kinematics module is at the heart of the robot's movement, enabling it to assume specific poses or gestures. This module utilizes inverse kinematics to calculate the necessary joint angles for achieving desired positions, employing geometric methods to find analytic solutions for leg movements. When the robot stands, the module calculates the angles for both legs to maintain balance. Forward kinematics is also a part of this module, used to estimate the robot's center of mass. This estimation is crucial for monitoring the robot's state and aids significantly in the design of control algorithms to maintain stability and fluid motion.
3. State Manager Module: The State Manager is pivotal in managing the robot's task states, allowing it to adapt to unforeseen circumstances such as collisions or falls. This module continuously assesses the robot's

situation and decides on the best course of action to recover or adjust its behavior. It serves as a critical link between the behavior module, which dictates the robot's intended actions, and the motion module, ensuring that movement is always aligned with the robot's current state and objectives.

4. Trajectory Generation Module: This module is dedicated to planning the robot's movement paths and generating smooth trajectories for its joints to follow. It employs a graphical interface that allows for the customization of movement patterns using piecewise cubic spline curves. This interface is instrumental in designing complex movements that are both efficient and aesthetically pleasing. Furthermore, the module is exploring the use of intelligent control algorithms to refine these trajectories, enhancing the robot's ability to move in more nuanced and adaptive ways.



Together, these modules form a comprehensive system that enables our robot to perform a wide range of movements with high precision and reliability. By continuously refining these modules, we aim to achieve even greater levels of performance and versatility in our robotic systems.