Walking

Walking algorithm is the basis of robot movement and competition. Its goal is to realize the robot's stable starting, walking, stopping, kicking ball and standing up. And it is also expected to maintain the robot's stability when disturbed. Therefore, we are committed to enhancing the anti-interference ability of gait and the robustness of the algorithm as much as possible.

In terms of specific implementation, the basic principle of the algorithm is three-dimensional linear inverted pendulum model and the ZMP theory. And our implementation method refers to the quintic walk algorithm of team Bitbots, which uses a large number of parameters to generate the center of mass and the origin of coordinates of feet, to obtain the current position and the target position of the next frame, and uses the interpolation algorithm of quintic polynomial to calculate the trajectory. Then the inverse dynamics method is used to get the target position of each motor. The program code is written into Jetson TX2, which sends instructions to STM32 for controlling the servo motors to realize the movement of the robot.

Following the above strategies, the gait performance of our robot has been well evaluated. Through tuning the parameters frame by frame, our robot can basically achieve stable walking. However, the open-loop algorithm has some problems that may cause the robot to fall into an unstable state affected by the fluctuation of grass. Thus, a new feedback control method was proposed to make small disturbance compensation by trunk shaking, so that the ZMP point would still be close to the current support point when disturbed. This new compensation method is planned to be integrated into the current algorithm in the coming semester.

Vision

The vision part is an important information source to guide the robot to make decisions and actions. The vision algorithm needs to recognize the object and calculate the distance. It uses the image information obtained by the monocular camera to recognize the football, the sideline of the field and the goal frame line, and returns the vectors representing the position and distance of the target object for the calculation of the decision part. In order to enhance the performance of the robot, we need to improve the algorithm accuracy and arithmetic speed as much as possible.

The current vision algorithm is based on OpenCV library of computer vision, using deep learning method and training through artificial neural network. Our neural network system adopts nonlinear decision-making strategy, which is constructed by two layers of sigmoid elements, that is, one layer of hidden layer and another layer of the output layer. The size of the hidden layer is determined according to the actual calculation ability and the technical details of the processor. The size of the output layer is no less than 3, and the relative position of the ball is determined according to the output vector.

Regarding input coding, we adopt gray-scale maps of 32p resolution which has been preprocessed. To increase the operation speed, the grayscale of each pixel is obtained by random sampling of the corresponding region of the source image. This strategy makes it easy to convert input into vector processing. In terms of output coding,

we combine direction and distance in the form of vector to indicate the position of the ball. To accelerate the training process, the minimum value of the standard output of the training sample is slightly greater than 0, and the maximum value is slightly less than 1.

The neural network structure adopts the standard three-layer acyclic sigmoid network. In the process of training, the neural network modifies the weight vector while cross-verifying the size of the hidden layer. That is to say, lack of hidden layers will lead to poor fitting result and waste of training samples. By contrast, too many hidden layers will lead to over-fitting and high training costs. We will add impulse term to enhance accuracy and speed up training in the process of applying the backpropagation algorithm.

After debugging in the simulation environment and testing on the real robot, our current vision algorithm can basically identify the ball, the side line and the goal, providing accurate information for the decision algorithm, and guiding the robot to react properly. Now we are planning to use the ZMD mini, a binocular stereo camera, to substitute our current monocular camera to obtain better image information including the depth of the photo. Then we may develop a new vision algorithm based on the knowledge of 3D vision.

Localization

Self-localization plays a vital role in robot software system, since the robot needs to estimate its position and orientation on the football court. Specifically speaking, robot localization is a state estimation problem, where robot needs to calculate position and orientation based on the data provided by firstly, sensor data, namely the data from the camera and IMU, secondly, control information, where the torque information of the angle motors are used as input, thirdly, a map, which is a global reference frame with landmarks in essence. The two major elements of the localization process, i.e. scan matching and probabilistic model are elaborated in details as follows.

Scan based localization fetches information from camera, which provides the photos of the environment. Since our robot applies a monocular camera, a 2D picture of the surrounding environment is sent to the computation model. The gap between 2D photo and 3D environment indicates that at least two pictures as well as a shift of robot camera pose is needed to implement localization. To figure out the relative transformation between two locations the Iterative Closest Point (ICP) matching algorithm is applied.

This algorithm first extracts the landmark points in each picture, for every point it picks from the other corresponding picture that is the closest. Then we use these correspondences to estimate the transformation. A desired alignment would be accessed by repeating the above two steps. Therefore, the transformation between two figures is specified.

Actually, if we possess the accurate initial pose of the robot on the soccer court, with the aid of ICP algorithm we could have implemented the localization process. However, due to the existence of noise, we need to apply the probabilistic model to solve this problem. To eliminate the noise, we use the particle filter algorithm.

The prediction, or control update, incorporates the states of particles with data from the odometer and IMU, and then some Gaussian noise is added. In the measurement update, we first incorporate the data from the camera and the odometer, so we can distinguish similar landmarks and know their directions relative to the robot, and then we can update the states with this information. After that, we resample the particles. In this step, we're trying to keep as 'many low-probability particles' as possible. In the fourth step, we draw a final estimation from the particles, which can be used to make decisions in behavior algorithms. The state space is divided into 10x10x10 cells and we find the 2x2x2 cube which has the most particles. The weighted average of particles in this cube is the final estimation.

Behavior

Inspired by the hierarchical state machine (HSM) programmed in XABSL, we introduce HSM into our algorithms utilized to generate behavior. Our HSM is implemented with Lua embedded in C++. The framework of the algorithm is composed of several super-states and each super-state is constructed by a group of subordinate states. In our case, those super-states include kicking the ball, listening to the controller, standing up, and defense. Correspondingly, those subordinate states, which are all basic actions able to be executed directly by the robot, involve searching the ball, approaching the ball, shooting at the goal, etc.

Apart from our current policies, we are also interested in multi-agent reinforcement learning for behavior, which we have already carried out some relative research in simulation. Hopefully the new method would be deployed on the next version of out robot in the near future.

Contributions to the RoboCup community

To achieve robust robot locomotion more efficiently, bipedal walking algorithms based on deep reinforcement learning methods are highly desired. To this end, we proposed to design an adaptive reward function for imitation learning from the references and used Adaptive Mimic algorithm to train the agent to achieve a fast and stable walking pattern from an infeasible reference. This work was published on https://arxiv.org/abs/2112.03735.