Survey response 28

Software

Team Name
Blenders FC

Is your software fully or partially OpenSource. If so, where can it be found:
Our software is fully OpenSource and can be found in https://github.com/BlendersFC/BlendBots_FC.git

Do you have a kinematic or dynamic model of your robot(s)? If so, how did you create it (e.g. measure physical robot, export from CAD model)?
Yes, we have a kinematic model that was developed with official measurements of the physical robot and represented as links where the final effector is the foot in order for this to be always parallel to the floor.

Are you using Inverse Kinematics? If so what solution (analytic, (pseudo)inverse jabcobian, etc...) are you using?
Yes, we are using Jacobean according to Kajita in Introduction to Humanoids book.

Are you simulating your robot? If so what are you using simulation for?
Yes, we made the kinematic calculations through MATLAB and also simulated in this software to generate a walking algorithm and key poses. We have also setup the environment to test our algorithms and robot behavior in Gazebo.

What approach are you using to generate the robot walking motion?
The initiation of walking involves aligning the robot's head with a designated target, such as a ball. Once the head achieves precise centering, tilt and pan values are derived. Utilizing trigonometric calculations based on these values, the 3D position of the ball is accurately determined. This information is then instrumental in defining the robot's intended walking trajectory. Through the application of the Kajita mathematical model. This model treats the entire robot as a constrained pendulum, with its center of mass strategically positioned at the hips. By adhering to this framework, the robot follows a specified trajectory while ensuring the center of mass remains within its support polygon.

What approach are you using to generate motions for standing up?
Inverse kinematics is used to define a starting and a final position, these are defined in a function that is called when the IMU surpasses a threshold and in between points are interpolated to create a full standing up motion.

What approach are you using to generate kicking motions?
To generate kick motions we are using the kinematic model by processing key poses in the movement. This allows us to maintain stability and control how far we can kick. Because of an internal controller in the robots' servos, poses can be interpolated without having to provide an exact position for each sample time.

Do you use any other motions than the previously mentioned? If so, what approaches are you using to generate them?
In the motion area, we also worked on the robot's walk, where the programming included in the robot is used as a base, but we modified certain parameters to make the walk more stable, according to the type of terrain that we had (artificial grass). We also focused on the way the robot gets up whether it falls forward or backward, and to implement this we based ourselves on Kajita's algorithm in the book "Introduction to humanoid robotics".

Which datasets are you using in your research? If you are using your own datasets, are they public?
A dataset for Haar Cascade is used (own dataset), in which photos and videos of the ball and the soccer field were taken by the members in our team's laboratory to train the model. This model has been in constant improvement to detect the fifa white soccer ball required.

What approaches are you using in your robot's visual perception?
The robots' visual perception is approached with three different strategies: ball detection, goal detection and line detection. Ball detection was originally detected by HSV segmentation for the Mexican Robotics Tournament; however, because of the challenges we faced with the illumination and surrounding elements, it was later improved with a machine learning based algorithm by cv2 named Haar Cascade. This model was trained with positive and negative images taken by the team in a controlled environment. On the other hand, line detection was implemented with the Hough Transform and Canny cv2 functions, helping the robot localize itself. Finally, in the goal detection algorithm, initially, we identify the green grass in the scene, treating it as a key reference for the goal posts' lower section. Once the green regions are isolated, we apply grayscale histograms exclusively within these areas. Subsequently, we look for borders or contours within the interest areas with the highest white pixel concentration, obtaining the coordinates of the center of the goal. Both of these points, ball and goal, are finally communicated to the movement algorithm as ROS topics.

Are you planning with objects in Cartesian or image space? If you are using Cartesian space, how do you transform between the image space and cartesian space?
At the moment, objects are detected with image space, centering the object of interest in the camera view and simple trigonometric functions are used to transform into cartesian space. However, the visual SLAM method is planned to be implemented as a way to detect the exact distance between the robot and the objects of interest or the other robots too.

How is your robot localizing?
The anticipated program for the robot uses an Adaptive Monte Carlo Localization (AMCL) algorithm to generate particles and estimate the robot’s position on the map. Currently, the team faces an instrumentation restriction due to the simple monocular cameras on the robots, but the use of a 2D SLAM in the system is intended to be able to generate the desired map and distances with the equipped cameras. By using a visual SLAM method, we can test the functionality of the ORB-SLAM2 and OpenVSLAM systems for their efficiency, and be able to use the particle filter algorithm to facilitate localization.

Is your robot planning a path for navigation? Is it avoiding obstacles? How is the plan executed by the robot (e.g. dynamic window approach)?
The algorithm estimates the distance using the camera's tilt. Once the distance is determined, the robot plans a path in three dimensions. The footsteps planned in advance are computed using the Jacobean matrix (using the walking module embedded) , ensuring effective navigation before reaching a predefined threshold, which is a proximity suitable for kicking. However, the already existing algorithm will be improved to avoid obstacles, like other robots and make a more collaborative match within the robots of the team. Also, by using our goal detection algorithm, which already has a line, green and white mask implemented, the only step left will be subtracting the ball and any other obstacle will be avoided by changing the robots path.

How is the behavior of your robot's structured (e.g. Behavior Trees)? What additional approaches are you using?
It is structured as a state machine. The robot initiates the process by actively searching for the ball until it successfully locates it. Once the ball is detected, the robot centers it within its vision, head movement. Subsequently, the robot starts walking towards the ball, continuously checking the distance. When the distance becomes sufficiently short, the robot proceeds to kick the ball.

Do you have some form of active vision (i.e. moving the robots camera based on information known about the world)?
The head control operates in two distinct states depending on the presence of a ball within the camera frame. On one hand, in the event of ball detection, a precision-oriented head control mechanism is engaged, aligning the center of the ball precisely with the camera frame's center through calibrated tilt and pan movements, thereby minimizing positional error. On the other hand, when no ball is detected, an algorithm for ball search is initiated. This algorithm employs trigonometric functions, causing controlled oscillations of the head within a predetermined range to explore the environment. Upon detecting a ball, the system transitions to the first state. If, after a 15-second interval, no ball is found, the robot autonomously rotates to facilitate a new round of environmental exploration.

Do you apply some form of filtering on the detected objects (e. g. Kalman filter for ball position)?
We are currently not implementing a Kalman filter for ball position in our field players, but our goalkeeper is equipped with a Kalman filter, which is used to predict ball trajectory in order to move its body to the best position to block a goal.

Is your team performing team communication? Are you using the standard RoboCup Humanoid League protocol? If not, why (e.g. it is missing something you need)?
Yes, our robots’ hotspots are turned off and all four are connected to a router, which a workstation is also connected to run a master and launch each robot individually. Though not participated yet in RoboCup, our robots are able to listen to the referee.

Please list contributions your team has made to RoboCup
Despite not participating in RoboCup yet, we’re advancing our robotic skills through other approaches. Our focus is on continuous research and development to increase our tech expertise and contribute significantly to the field and the Humanoid League’s goal.

Please list the scientific publications your team has made since the last application to RoboCup (or if not applicable in the last 2 years).
Our team hasn’t produced any scientific publications. We are actively looking to provide contributions to the scientific community in the near future.

Please list the approaches, hardware designs, or code your team is using which were developed by other teams.
Our team is not using designs nor code from other teams, but we do access the ROBOTIS OP3 open source demos and modules to aid our walking algorithm.

What operating system is running on your robot and which middleware are you using (for example Ubuntu 22.04 and ROS2 Galactic)?
Workstation: Ubuntu 20.04 and ROS Noetic 1.16 Robot: Ubuntu 16.04 and ROS Kinetic Kame

Is there anything else you would like to share that did not fit to the previous questions?
The team is currently working on several alternatives to improve and make our algorithm more robust. We have developed a model with YOLO and a previously trained model with footballs; however, computational capacity has been a challenge to run in a workstation or inside the robot. Looking at solutions, we may look into changing the main controller with one with greater capability.