Global Systems Description

Humanoid League (KidSize)

UTRA

Jonathan Spraggett, Amy Loh, Jason Wang, Nam Nguyen, Derek Lam

(Canada)

1. Software

1.1 Robot Operating System (ROS)

The software architecture is based on ROS (source code). Our architecture is largely based on the Humanoid League Messages by Hamburg bit-bots, however, we have several design changes. Instead of creating multiple ROI regions for different regions of interest (balls, field lines, etc.), we have a node called field ROI that first detects the grass and filters out all the objects that are not of interest. Then, geometry is used to locate the points on the field in 3D.

1.2 Computer Vision

Information is obtained from the images acquired by the camera through a series of filtering stages in the computer vision system. The filters are as follows:

– Field Detection - A color filter supported by a color space estimator (which obtains the color of the grass) to get the area of the field. The results can be seen in figure 2.

– Field Line Detection - From the field area message we detect straight lines and find the individual intersections of all those field lines using Hough Lines. The results can be seen in figure 2.

– Pole Detection - The pole is detected using the Hough Line Transform. A high threshold value is set to ignore the vertical net lines and hence to only detect the vertical pole lines.

– Ball Detection - A FCNN trained with Pytorch locates the ball and sends its coordinates. The results can be seen in figure 1.

– Robot/ Obstacle Detection - A FCNN trained with Pytorch locates obstacles and sends its coordinates. The results can be seen in figure 1.

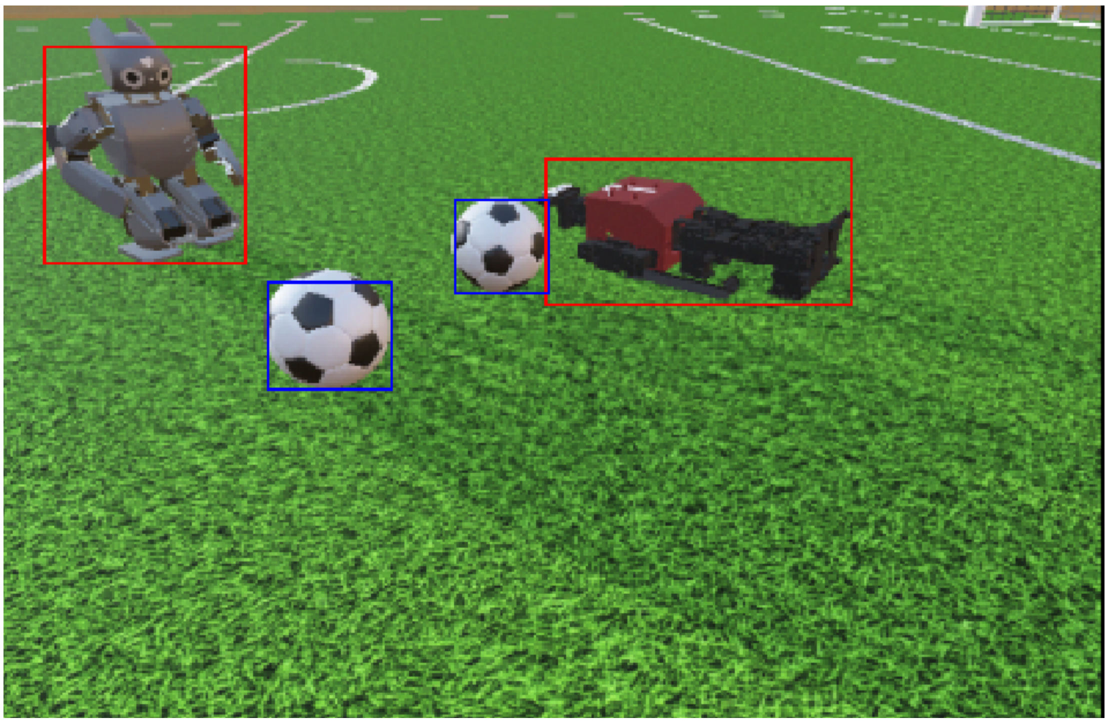

Fig. 1. FCNN Output. The blue box is the ball and the red is a robot

1.3 Localization

An adaptive Monte-Carlo localization approach is being used to get the location of the robot on the field. This approach requires laser scanning and odometry data to work. From the computer vision system coordinates from the field line detection are converted from point clouds into laser scans. Odometry data is estimated from the walking engine based on the footpath and fused with IMU data in an EKF. The overall process of the computer vision system and localization is shown in Fig 2.

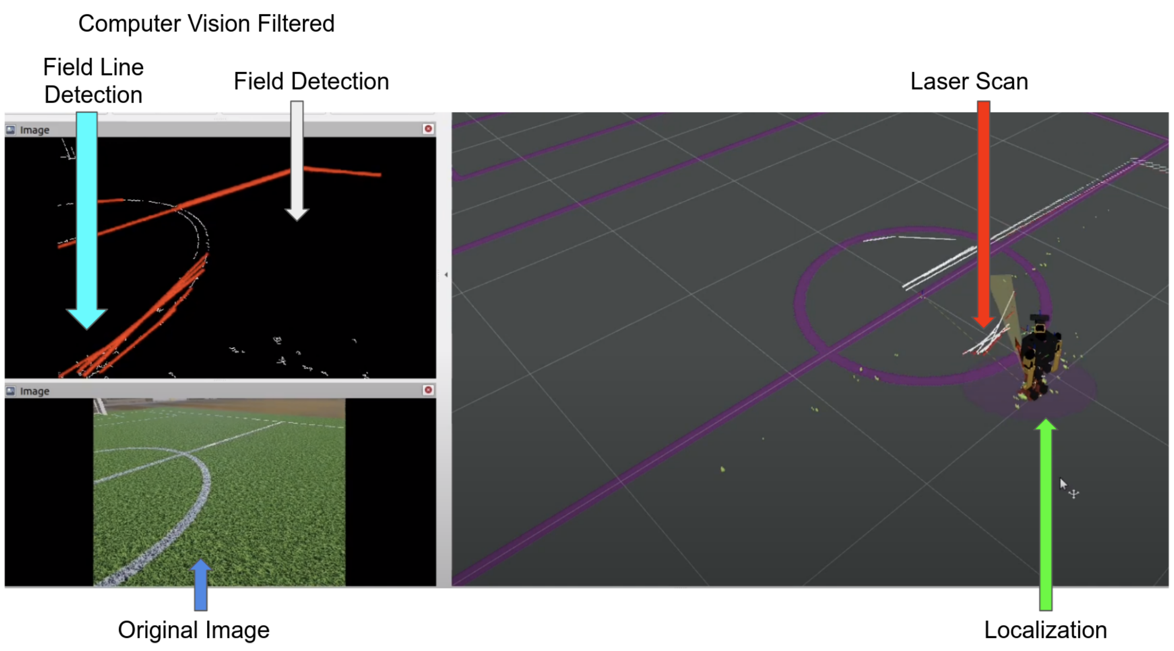

Fig. 2. Localization process

1.4 Mechanical Control

Leg trajectories are determined using inverse kinematics. First, a fixed trajectory for the robot's body was generated. Using this trajectory for the body, two footstep trajectories were determined alongside the robot body to move the robot from point A to point B. The locations of the footsteps were then calculated. Inverse kinematics was calculated to move the robot whilst maintaining balance. A multi-state PID controller using the yaw and pitch angles of the robot as feedback was used to stabilize the movement

1.5 Robot Strategy

A Hierarchical state machine was used to determine the basic strategy of the robot based on the inputs. Various team-based strategies were developed, and the strategy framework allows different strategies to be easily swapped during the game. A testing framework was also written to allow the testing of various strategies without depending on other software components.

1.6 Team Communication

A serverless UDP system was built for communicating between robots. Information passed between the robots includes states and localization information. This is yet to be implemented to improve the localization of the robot.

1.7 Navigation

RRT (Recursively random trees) was used to make path decisions based on obstacles. The trajectory was generated in advance and will be regenerated in the case of a fall or being blocked by an obstacle System Architecture A multi-agent docker setup was created to be able to simulate multiple robots using ROS and Webots

2. Hardware

2.1 Overview of Electrical and Mechanical systems

UTRAs robot is inspired by the Darwin-OP open-source humanoid research robot [1]. It stands 50 cm tall and weighs 2.3 kg. There are two control units on the robot. One is a Jetson TX2 module from NVIDIA that handles high-level and computationally-intensive algorithms such as computer vision, localization, and artificial intelligence (AI). The other control unit is an STMicroelectronics ARM-based 32-bit microcontroller, which handles the real-time control algorithms for controlling the actuators and acquiring motion data. A communication link between the two units uses publisher-subscriber messaging to transmit high-level commands (such as moving to certain coordinates on the field) and relay state information. This approach of using two controllers in an asynchronous manner disentangles the control systems from AI and computer vision, thereby reducing the probability of computational bottlenecks and improving reliability.

2.2 Mechanical design and manufacturing

2.2.1 Overview

The robot is actuated by 6 Dynamixel AX-12A and 12 Dynamixel MX28-12A servos. There are six Dynamixel MX28-12A servos in each leg and two Dynamixel AX-12A in each arm and neck. All of the electronics are embedded inside the box-shaped body. A camera is fixed to the top of the body. The robot's structure is 3D printed. This helps reduce manufacturing effort and increases precision over handmaking metal components. The robot is depicted in Fig. 3.

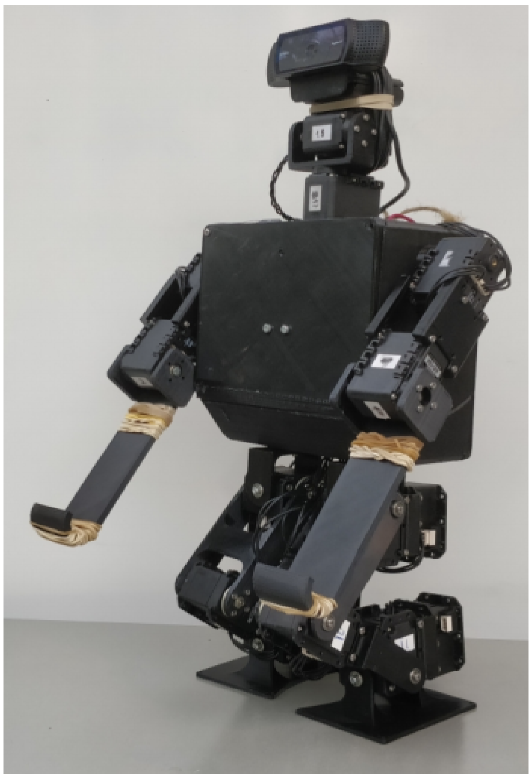

Fig. 3. Photograph of the robot.

2.2.2 Torso

The robot's torso has a simple box design with a removable front and backplate for easy access to electronics. This design was chosen because the electrical components were still being determined. This simple design would help the robot to be more flexible with electrical component choices and upgradeable in the future. During design, we had a choice of putting servos on the outside of the body or inside. Putting the motors on the outside would reduce the size of the torso thus printing time would be reduced. But that in turn would make mounting motors hard because the screws would have to be inserted from the inside of the torso. For ease of assembly, motors were placed on the inside. The inside of the torso is depicted in Fig. 4.

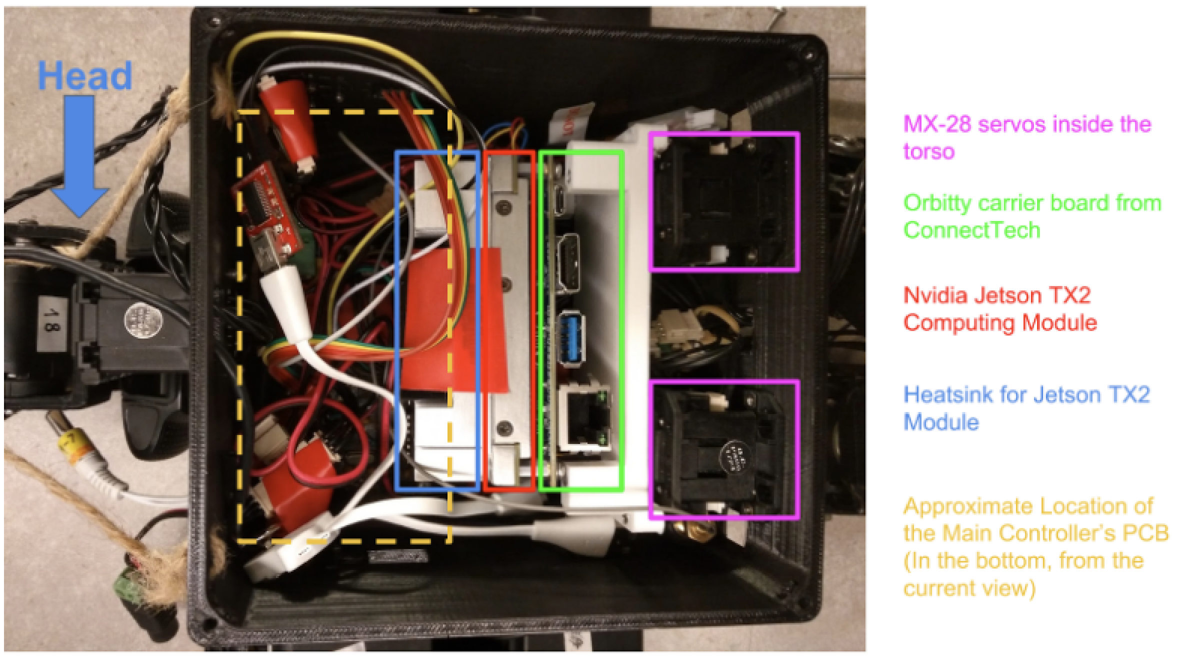
Head
MX-28 servos inside the torso
Orbitty carrier board from ConnectTech
Nvidia Jetson TX2 Computing Module
Heatsink for Jetson TX2 Module
Approximate Location of the Main Controller's PCB (In the bottom, from the current view)

Fig. 4. Electronic components inside the torso.

2.3 Electronic design and manufacturing

2.3.1 Jetson TX2 Embedded Computer

The Single Board Computer (SBC) hosts an NVIDIA System on a Chip that runs Ubuntu 16.04 on its quad-core ARM Cortex-A57 and is also capable of high-performance parallelized computing through the 256 Pascal GPU cores. The TX2 module is used with the Orbitty Carrier board from Connect Tech Inc. to make its I/O accessible. The TX2 is responsible for running the high-level control and strategy software as well as computer vision algorithms; thus, the carrier board is connected to a Logitech C920 HD camera for image acquisition. The TX2 module also dedicates a USB port for communication with an ST microcontroller, which handles the real-time control of the peripherals on the robot.

2.3.2 MicrocontrollerComputer

A STM32H743ZITx microcontroller (hereon STM32) on a Nucleo-H743ZI development board is used to execute the low-level software that handles communication with the inertial measurement unit (IMU) and servo motors in real-time. The buffer circuitry lets the microcontroller read and write to the Dynamixel smart servos that utilize a half-duplex TTL UART interface for communication. There are 5 such buffer circuits on the PCB which allows 5 independent daisy chains of servos that the microcontroller can use to communicate with 5 servos at the same time. This helps decrease communication latency which is crucial for active control during walking. All the software, depicted in a simplified manner in Fig. 5, was written from scratch with the exception of FreeRTOS and the hardware abstraction layer provided by STMicroelectronics

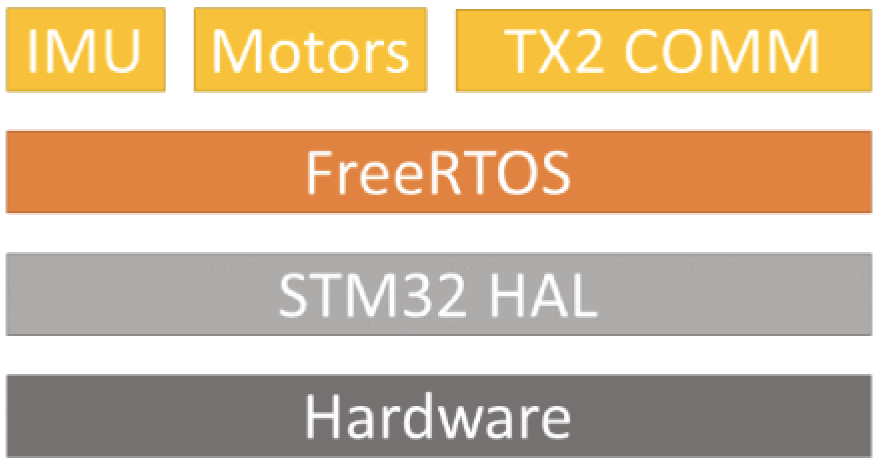

Fig. 5. Software organization. The microcontroller executes 3 processes on top of FreeRTOS.

2.3.3 Actuation

12 Dynamixel MX-28s form the legs (4 daisy chains of 3 motors), and 6 Dynamixel AX-12As form the arms and neck (1 daisy chain). In software, each daisy chain has its own queue into which commands to update the goal position and read the current position can be sent, as well as its own thread to service the commands. When the microcontroller receives a packet of goal angles from the Jetson, it distributes the write commands among the queues and the threads initiate asynchronous transfers via DMA to update the goal positions of the smart servos. The architecture of the embedded software can be found in figure 6.

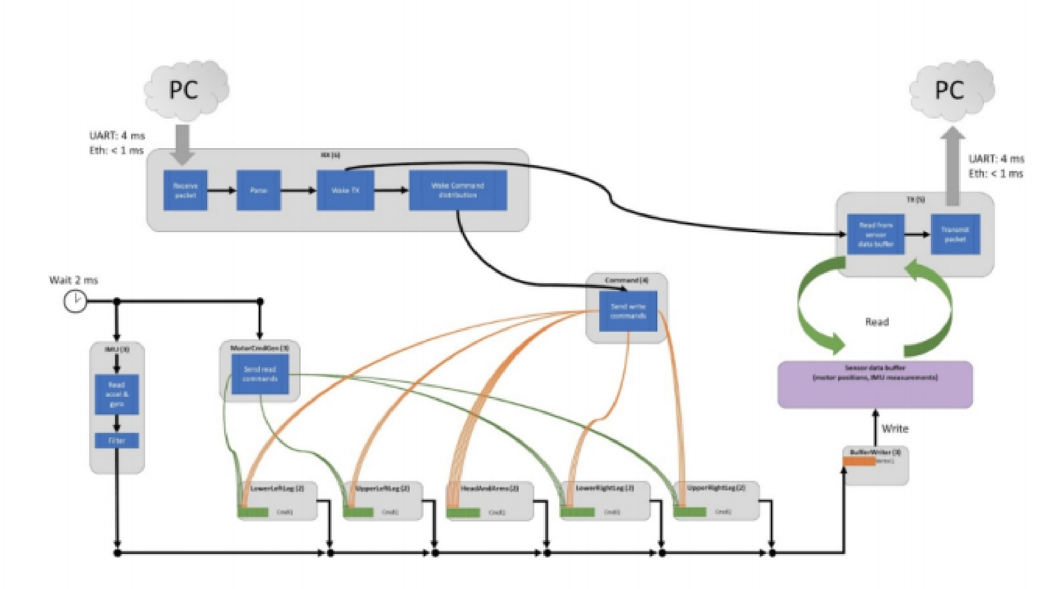

Fig. 6. Embedded software

2.3.4 Sensing

The parts of the software related to retrieving sensor data — current positions for the legs and inertial measurements (MPU6050) — are time-triggered. Fresh sensor data is collected every 2 ms.

2.3.5 Communication

UART and a virtual serial port are used to transfer data between the Jetson and microcontroller. The packets sent and received are of identical structure and take 4 ms to transmit at the current symbol rate of 230,400. Because tests with Ethernet on the F7 demonstrated less than 500 μs mean transmission time under busy CPU conditions and Global EDF scheduling on the Jetson side, this communication system will likely be pursued in future designs.

2.3.6 Power System

The battery inside the robot is a 3-cell 2200mAh LiPo variant that powers all electrical components of the robot: Microcontroller PCB (which provides power for the servo motors) and Nvidia's Jetson TX2 computer module accompanied by the Orbitty carrier board from ConnectTech Inc. It is estimated that the battery can barely last a match of 10 mins.

3. Performance Evaluation (Result)

Based on our performance in RoboCup 2021 virtual competition, we can conclude that our software systems worked as intended, and is able to direct the robot to play a full game and score goals. As demonstrated in our demos below, our current systems are capable of directing the robot to track the ball, walk towards the ball, localization, score a goal, goalie logic, perform getups and kicks, as well as various other functionalities that enable us to win games.

Intelligent behaviour from Bez in Webots

UTRA RoboCup 2020 Humanoid League (KidSize class) Submission

4. Acknowledgements

We acknowledge the University of Toronto for providing the necessary space and funding for the project.

5. References

1. Trossden Robotics, Darwin-OP Humanoid Research Robot - Deluxe Edition. Available online at http://www.trossenrobotics.com/p/darwin-OP-Deluxe-humanoidrobot.aspx