Robobulls Development Goals

This document details issues the project faces that are as-yet unresolved along with all known information on those problems thus far.

1) Consolidating Build Environment

High priority! The most current development is located in the ssl-league-fixed branch. However, several altered team configuration files used in robobulls vs robobulls simulation tests are stored in the team-blue-config and team-yellow-config branches (an improper use of git branching). Additionally, all currently active branches (team blue/yellow, ssl-league fixed) are both ahead and behind of master in commits, and both the team config branches and ssl-league-fixed are branching away from one another.

In the interest of smooth collaboration, it is extremely important to consolidate the branches into a more workable environment ASAP. The team configuration files (in team-blue-config, the files altered by these commits are detailed in 16b7364 and 4dd3b63; in team-yellow-config, these commits are 03b7293 and 4a2f9df) should be written into an option into the Robobulls GUI, if possible, or stored & read otherwise without requiring independent branches. Once done, these two branches can be depreciated. The commits in each branch beyond the configuration file changes will need merging into master.

Once those changes & merges are in, ssl-league-fixed can be merged into master and depreciated, leaving master as the sole branch. From there, the team can checkout a dev branch and work together in the common master/dev branch style or whatever method you prefer.

2) Yisicomm Packet Improvements

The YisiRobComm::sendVelsLarge function in the yisirobcomm.cpp file details how Robobulls constructs the communication packets to be sent to the physical bots. Our implementation currently sends packets to bots one by one, although the Yisibot transmitter is capable of updating three bots simultaneously (see the Yisibot manual for more information). As the camera and packets are limited to 60 FPS, improving packet construction to update three bots per packet should markedly improve the robot's reaction speed.

3) Kick Type & Power Modulation

The bots should use the appropriate power when making goals/passes and choose between normal/chip kicks, but are currently hard-coded to use max power kicks of only one type - see the shootMode and shootPowerLevel variables in yisirobcomm.cpp. In other words, whenever the bots want to kick the ball, pass it, chip it, etc, they will simply slam the ball at full force, in probably the wrong kick type.

The ball should never exceed the 6.5 m/s ball speed limit (measured in 3-dimensional space) limit as defined by the SSL ruleset. Note: In the robot model in model/robot.h, there exists a getChip() function, which may be useful. There does not yet appear to be any logic for calculating kick power, however.

4) Integrate new Refbox communication protocol

Since approximately 2012-2013, SSL-Refbox has implemented a new communication protocol which sends much more game information than the legacy protocol. Currently, Robobulls is using the legacy implementation. Robobulls should update to the new protocol and improve our AI to make use of the additional information to make better, more strategic decisions during the game.

5) Protocol Buffers (protobuf)

The new Refbox communication protocol as well as grSim uses Google's Protocol Buffers to communicate over the network. At the time of this writing (7/25/18), grSim (and probably the new refbox packets) expects protoc version 3.0.0, whereas Robobulls requires version 2.6.1 to compile, causing issues when attempting to install the software out-of-lab. To better integrate with the other programs and improve ease of installation, Robobulls should move to the same protobuf version that the other programs are utilizing.

6) Installation Script Improvement

The installation guide details how to install the Robobulls dashboard in and out of lab; however, installation outside of the lab is an arduous, largely manual process. The existing installation script should be adapted to better handle those circumstances.

7) Robots frequently collide during pathing

Robubulls uses FPPA pathing (fast path planning algorithm), the details of which can be

found here, and the implementation of which is found in movement/pathing/fppa_pathplanning files. Special note to commit 56bba26, where a 'personal bubble' was added around each bot, in which obstacles are ignored, as well as commit 08b6f67, which runs that algorithm as much as possible. Together, these changes seem to introduce behavior that results in bots immediately disregarding any obstacle that ever comes too close.

See Also: The ToDo.html list found in the HTML documentation.

Additional Resources

Robobulls Readme
Robobulls Installation Guide
Robobulls Operation Guide
Robobulls Development Goals
SSL-Robocup Ruleset
SSL-Refbox Documentation
SSL-Vision Official Documentation
SSL-Vision Robobulls Operation Guide
Yisibot Manual
Robobulls GUI Honors Thesis (Overview of Robobulls)

Also, see the Robobulls pre-2016 Doxygen HTML documentation, found by opening the Documentation.html file located in documents of the main project folder.

Contact Info

After reading all of the above documentation, if you need additional clarification on an issue detailed above and the current lab admins are not available or are unfamiliar with your problem, you can contact one of the previous team members for more information.

May-Aug 2018 Team Members:
nherbert2@mail.usf.edu

- Familiar with Yisirobot movement code, out-of-lab installation. Authored readme, devgoals, installation guide, operation guide. (8/03/18)

kellerc@mail.usf.edu