SSL ETDP notes

Immortals:

use a gyro to do angular prediction and help with visual latency.

also use it for delta margin shooting (a la CM Dragons)

they're able to run robots up to 10 seconds without vision data

use an outer PID loop on omega

changed their wireless comm to a new RF module because their bandwidth sucked. They use UDP, bidirectional. That means they can run several AI computers. They also are calculating interference that happens from data loss, but not doing anything with it other than changing the channel automatically to try to remove it.

"screw performance of our code on the AI computer, we've got i7's, just throw more hardware at it"

Memory is the new bottleneck

They focus on data oriented design

They say object oriented sucks because it degrades the performance of memory due to unneeded caching. Pay attention to what's going to get cached.

Want to use GPU computing to do field evaluation and calculating the safety of every point on the field and doing path planning with that

they get 0.2 ms delay for AI and vision filtering on a laptop with their code

SB RRT is a new path planning algo they use. It biases the tree to safer areas of the field.

Use ANN to remove some obstacles

KIKS:

nothing

MRL:

have a huge lab, government level resources

AI architecture is based on STP platform by James Bruce

dynamic assignment of some pre defined roles to robots

dynamic role assignment is hard

how to pick strategies:

random

higher scoring algo but still some randomness so lower performing ones get picked sometimes

weighted

use (or want to use) GPU computation ie OpenAcc, OpenCL, Directe Compute, Cuda

calculate a score between 0 and 1 for each opponent robot to prioritize who to attack. Goalie is always 0.

For each team, they try to figure out which positions on the field are most important. They just plot in 3D the goals vs location. [note, just looking at goals isn't enough. sucessful passes and stuff like that are also important. Look at how sabermetrics decides what is valuable and try to apply it]

They characterize both dynamic and static ball positions

obsessed with beating and/or being better than parsian haha

Regioning: look for empty spaces on the field. To find them, they put virtual point lights on the field and look for "shadows" to find objects and open chunks of field.

have online debugging [which sucks compared to ours]

"we can really quickly change everything" [no kidding, that includes during the match...]

New electronics have a big ARM7 and an FPGA. Real time stuff is done in FPGA. So ARM does RF module stuff, solenoid driver, dribbler, and debugger. FPGA does motor control and handles current protection by reading in the current sensor.

tried to optoisolate a bunch of stuff, but discovered that it sucks for latency and power efficiency. And they're huge. So they handle noise by just rounding.

New motor driver is stolen from Tigers. switched to NFETs, since PFETs suck. Not sure if they are using H bridges they built themselves or if they're using monolithic chips for that.

Wireless: using nRF2401 on 2.4 GHz. Also stick a PA on there (BBA-519-A) to get up to 50 mW

max speed of 12 m/s. Theoretically restricted to 8 m/s, but definitely cheated. Stayed with their old system. [we could probably save a lot of weight and space if we make ours only capable of 8 m/s]

they used a genetic algorithm to optimize all the parameters to find the best kicker. Publishing soon.

finally got on board with new wheels that everyone else uses.

moved their ball sensors [we need to do that too]

carbon fiber attenuates wireless a lot

CNC'd the entire shell out of a single piece of polyethylene

insanely fast movement can defeat their robot ranking algo

Parsian:

use Cuda to do the point light and shadow trick

Robodragons:

use a similar software architecture to us

do pretty simple ball following

Skuba:

when passing, you need to optimize the angle that you impart to the ball so that the receiver can catch it. "If you kick fast enough, everything can be considered a static system"

Passing twice is a good idea now because there are more robots

doing more passes gets harder because there are so many angles

[the diagrams in skuba's talk are good]

they do an evaluation to maximize the angles to shoot to each robot. That's all defined in their slides.

then they use a decision function to decide if they should pass once or twice.

their slides have a bunch of psuedocode in them about how they figure out where the ball is

Zjunlict:

decision system has an external file configuration, then does a bayes filter, a role match and FSM, and low level control

their playbooks are external text files

they use a bayes filter to decide if they want to attack, sit still, or defend

they use the munkres algo for rol matching