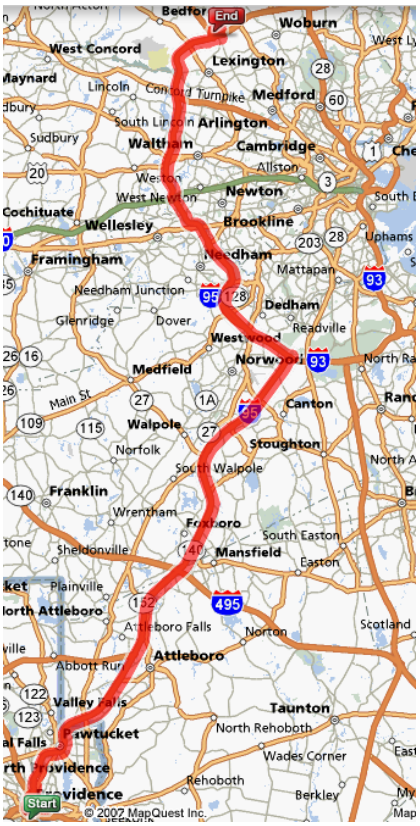
1 Dates

- Assignment 3 waypoint milestone due: Friday, October 16, 2009
- Assignment 3 project demonstrations in class: Friday, October 23, 2009
- Assignment 3 handin due: Friday, October 23, 2009 (10 pm)

2 Introduction

With our introduction to PSG in assignments 1 and 2 behind us, it is now time to dig into making our robots play soccer. The next four projects will follow different approaches to autonomous robot control for 1-on-1 soccer through deliberation, state estimation, reaction, and learning. In the current assignment, you will use path planning algorithms to control your robot soccer player using overhead sensing. In class, we will cover several different planning algorithms that you can choose from for your implementation. Your planner implementation within your control client will be providing updated game state and the static dimensions of the field. You will evaluate your client through basic skills challenges such as navigating to specific locations, and competition against other groups.



3 Path Planning

For this project, your client will deliberatively plan and navigate paths for the robot. At its core, this client must be able to maneuver the robot from its current pose on the field, expressed as the position and orientation triplet $X = (x, y, \theta)$, to reach a desired destination, also expressed as pose $X' = (x', y', \theta')$. A sequence of destinations can then form a trajectory for the robot to traverse. An estimate of the current robot pose is provided by a state estimation system from robot sensing given a map of the environment through a process called *localization*. In this assignment, localization is performed external to the robot using overhead cameras. The map of the soccer field is described in the following section. This setup will allow us to focus mostly on the planning problem, whereas the next assignment deals strictly with localization from onboard robot sensing. Note: even though a localization mechanism is provided, it is neither precise nor deterministic. Be careful to process state estimates with some considerations of uncertainty.

Given localization, a map, and a goal destination, you must plan and traverse a path from your current location to the goal. Planning is essentially a search over all possible routes from X to X', or configuration space. Graphs are typically used to express the space of possible poses (as graph vertices) and valid transitions between poses (as graph edges). Your robot client will need to construct graphs in the robot's configurations space and then search this graph to find good paths to traverse. For planning algorithms you have a number of options:

- Dijkstra's Algorithm[1],

[1] http://en.wikipedia.org/wiki/Dijkstra%27s_Algorithm

- A*[2]
- Potential fields (Choset Ch. 4) and Wavefront planning[3]
- Probablistic roadmaps [1] (Kavraki et al. 1996)
- RRT-Connect [2] (Kuffner et al. 2000)

There is pseudocode available for Dijkstra's and A* search on Wikipedia as well as many other sources. There are many verisons of Dijkstra's and A* available on the internet, you are free to use these just be sure to cite anything that is not your own.

4 Physical Soccer Environment

Shown in Figure 2, the course staff has set up the "FC 148" robot soccer field within the Roomba Lab for this assignment's games and individual challenges. The dimensions of this field are roughly 3.8m in width and 2.6m in height, as coarsely illustrated in Figure 1 [4]. The field's origin $(0, 0)$ is in the corner opposite the desk with the lab computers. The direction a robot is facing (*yaw*, or θ) is defined in radians and 0 if the robot is facing the wall opposite the desk with the lab computers. Turning to the right increases the angle of yaw until π, turning to the left decreases the angle of yaw until $-\pi$. That is, as the robot turns to the right the angle of yaw increases towards π until the robot is facing the wall with the desk. If the robot turns even more the angle will not increase up to 2π but rather start decreasing from $-\pi$. The collective field of view for these cameras is shown in Figure 2. Localization for these robots within the field of view is handled by a client-server based application written by the course staff. How to incorporate such localization information in your client program is explained in the next section. The goals are roughly 0.7m in width and slightly extend past the end boundary. A goal will consider being scored if the ball touches the goal material.

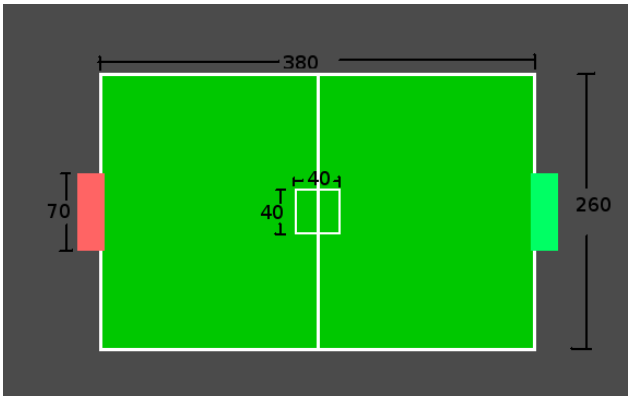


Figure 1: Coarse dimensions of the "FC 148" field in centimeters.

4.1 Localization Interface

4.1.1 Infrastructure

As mentioned above, a total of four cameras were installed on the ceiling. The cameras labeled 1 and 2 are connected to the computer *sandworm* and the ones labeled 3 and 4 are connected to *foxwood*. The course staff wrote a program called `tdlocc` (*Top-Down Localization Client*) that connects to a specific camera and analyzes the pictures taken by it. That is, it does two things: First, it uses the *ARToolkitPlus* library to find the robots with the black and white pattern on top of it and estimate their location and bearing. Second, it uses the *CMVision* library to find the yellow ball and determine its position. It then constructs a data packet which contains pose estimates on all the objects it has spotted and sends it to the localization server over the network using UDP. `tdlocc` repeats this process several times a second and thus delivers up-to-date pose

[2] http://en.wikipedia.org/wiki/A%2A_serach
[3] http://playerstage.sourceforge.net/doc/Player-2.0.0/player/classPlayerCc_1_1PlannerProxy.html
[4] The actual dimensions of the field vary slightly from these dimensions due to issues with ceiling mounting of the cameras.

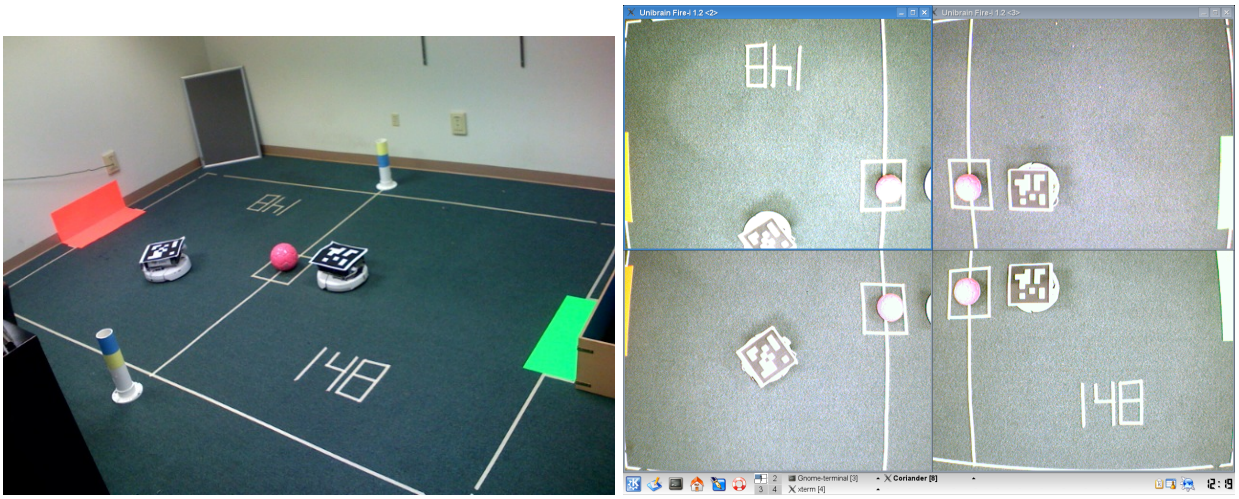


Figure 2: Snapshot of the "FC 148" robot soccer field.

information to the server. The localization server program, called `tdlocs` (*Top-Down Localization Server*), listens on network port 8855 for incoming pose estimates from localization clients and fuses those estimates into a list containing information on each object's last known pose. The server broadcasts this list several times a second via UDP to port 8856.

If you are in room 404 and you want to receive localization information in your client you need to make sure the four localizaiton clients (one for each camera) and the localization server are running. To do so, log into *sandworm* and start the server by executing `/course/cs148/bin/start_tdlocs.sh`. After a few seconds execute `/course/cs148/bin/start_tdlocc_sandworm.sh`. Then log into *foxwood* and and run `/course/cs148/bin/start_tdlocc_foxwood.sh`.

Each black and white pattern, placed on top of the robot, is assigned a unique numerical ID which is written on the back. Your program should take these ids as an input argument. In the list that the server broadcasts robots are identified by these IDs.

In addition to the client and server program the course staff wrote a Java application called smurv stadium that displays the packets it receives from the server graphically. That is, it displays all the objects with their estimates pose on a screen. After you started the server and the client application with the above commands you can run the display application by executing `/course/cs148/bin/start_smurvStadium.sh` on *sandworm*. Note: smurv stadium can't run on the same computer as your client as both need to access the same UDP port.

In order to receive UDP packets from the localization server on machines other than foxwood/sandworm, there is a program called udp-proxy[5] in `/course/cs148/bin` that forwards packets from foxwood or sandworm to the AIBO network. This may be useful for running localization clients on personal laptops or robots connected to AIBO. You can run the program with the following command: `udp-proxy -d 8865 <destination_IP>`. You can even use the broadcast address 10.100.0.255 for `<destination_IP>`, if there are several groups in the lab at the same time who all want to receive the localization packets over AIBO. Note that running udp-proxy will reserve the listen port in the same way that running a localization client does, so you cannot run udp-proxy and a localization client (like smurv stadium or your own robot clients) on the same machine at the same time. As such, please be sure to kill any udp-proxy processes that you start on foxwood or sandworm when you are done in the lab.

[5]The source code and a README are available in /course/cs148/src/udp-proxy

4.1.2 Software Interface

In order for your client to receive localization information it needs to listen for UDP packets (datagrams) on port 8856 and parse those packets. Fortunately for you, you do not have to spend hours of network programming. The course staff kindly offers you code snippets for C++ that do the trick. For those of you who do not trust the snippets or are curious what the packets looks like, here is there definition (in more or less Backus-Naur-compliant form):

```
packet       ::= header payload
header       ::= nr-objects
nr-objects   ::= UINT8
payload      ::= object*
object       ::= object-type object-id pose-age pose
object-type  ::= smurv | ball | obstacle
smurv        ::= UINT8 = 1
ball         ::= UINT8 = 2
obstacle     ::= UINT8 = 3
object-id    ::= UINT8
pose-age     ::= UINT32
pose         ::= pos-x pos-y yaw
pos-x        ::= FLOAT32
pos-y        ::= FLOAT32
yaw          ::= FLOAT32
```

where the terminal `UINT8` is an unsigned 8-bit integer, `UINT32` is an unsigned 32-bit integer in network byte order (big-endian) and `FLOAT32` is a 32-bit float in network byte order. In summary, each packet contains a list of objects where each object is characterized by a type (*smurv*, *ball*, or *obstacle*), a unique ID, the pose estimate (X and Y location as well as the rotation angle, i.e. *yaw*) and the age of that estimate, that is the time in milliseconds that elapsed since the pose estimate was received by the localization server.

4.1.3 C++ Interface

The snippet `/course/cs148/asgn/asgn3/client.cpp` shows you how to receive localization information in C++. You need all the other files in that directory as well, but do not worry, you only have to know what is going on in `client.cpp`. Type `cmake .` to generate a Makefile for you and then you can compile the binary simply by executing `make`.

4.1.4 Visualizing Your Planning

For this assignment it will be helpful to visualize your robot's path planning. A sample client and smurv stadium are available that allows your client to interface with smurv stadium. The source code is located in `/course/cs148/bin/smurvStadium_skel`. This directory contains a client that sets user specified smurv poses and a smurv stadium for visualizing this data. The client allows you to set the *(x, y, θ)* pose of the smurv you want to visualize (lines 74-76), and broadcasts this information (along with the overhead localization data) as UDP packets to smurv stadium. You should first start the client and then run smurv stadium. The code currently displays a light blue robot in the middle of the 404 field.

4.2 Overhead Camera Calibration

The four overhead cameras mounted on the ceiling all need to be calibrated with specific values for the localization system to recognize all objects on the field. Prior to starting the localization server and clients, run `/course/cs148/bin/coriander` on both foxwood and sandworm. Be sure to calibrate for both cameras

listed under `Camera Select` of the `Camera` tab. On the `Controls` tab, set each control to `Manual` if it is not already and specify the following values: `[brightness=305, auto exposure=350, sharpness=100, white balance-blue=100, white balance-red=75]`

5 Waypoints Milestone

Because this assigment is more involved than previous projects, we requrire an intermediate demonstration of your path planning algorithm before the project is due. There is no motion control aspect of the Milestone assignment; your client will not be controlling the robot however you do need to receive overhead localization information to determine a start pose, the goal location and obstacle positions. The Milestone is due on Friday, October 16th. You will need to demonstrate your Mileston implementation either during TA hours or by scheduling a meeting with the course staff.
The challenge is as follow:
An ARTag representing a Smurv object will be placed on the soccer field in 404; this is your start pose and you will need to specify the id of the tag to your client. The yellow ball will be placed on the field and the (x, y) position of the ball is your desired end location. ARTags representing obstacles will also be placed on the field and your calculated path cannot run over the obstacles. Using this information from the overhead localization system, your client will need to generate waypoints from the start location to the end location in the form of (x, y, θ) for each waypoint. This path represents the path your robot will take to get to the desired location. Your client should either output each waypoint to the terminal or provide a visualization of the path.

6 Skills Challenges

Your grade will be determined in large part based on performance in the skills challenges. There are two skills challenges that will be used to test the functionality of your robot controller: Goal Scoring and Collision-free Navigation. Shown in Figure 3, the goal scoring challenge requires your controller to drive your robot (starting from some arbitrary pose) to the ball (at some arbitrary pose) and push the ball into the goal. During this challenge, the robot and the ball must stay inbounds at all times and complete the task within 120 seconds. The outcome of a single scoring challenge trial is the time remaining at completion of the task,

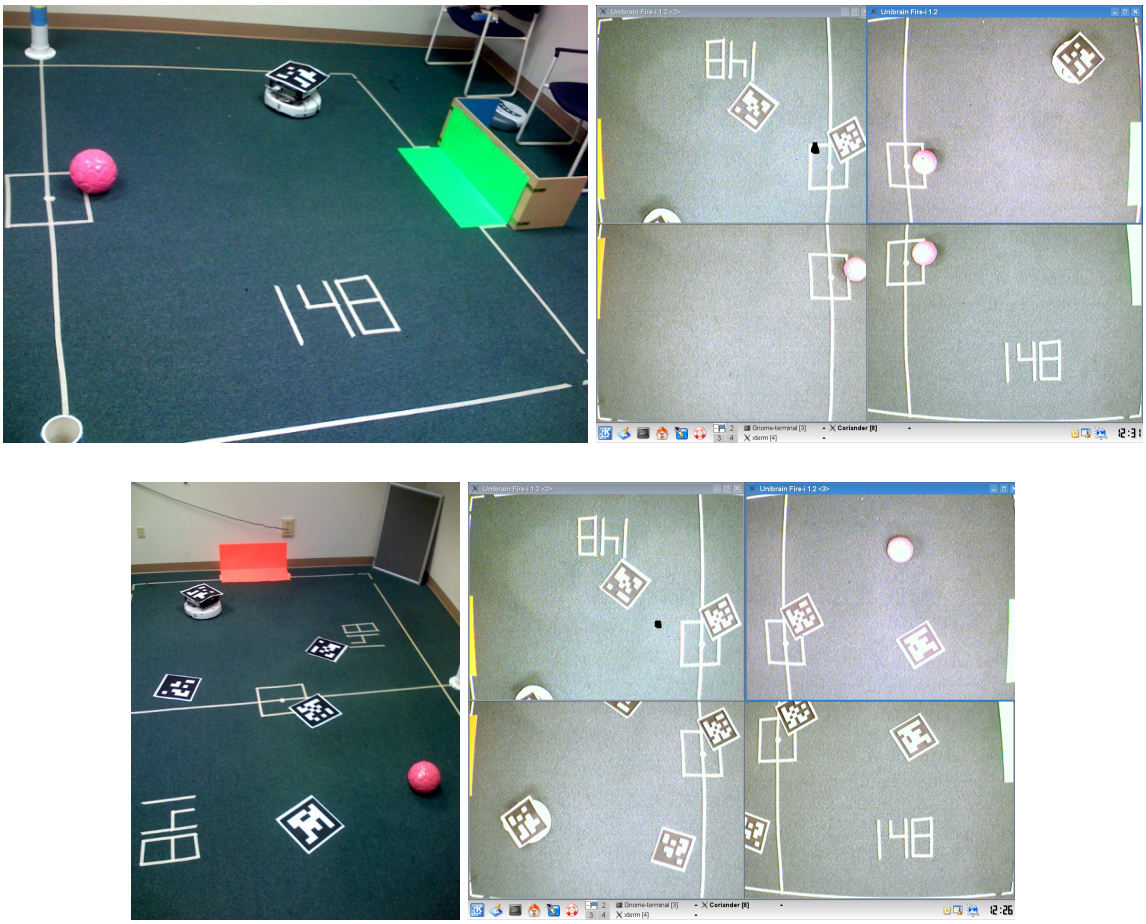


Figure 3: Illustrations of the (top) goal scoring and (bottom) navigation challenges.

where an outcome of zero indicates time has elapsed and the robot went out of bounds. The navigation challenge involves driving the robot from an arbitrary starting location to a ball while avoiding objects on the floor and the field boundary. During this challenge, the robot must complete the task within 120 seconds. The outcome of a single navigation challenge trial is the time remaining at completion of the task, number of collisions, and time spent in collision.

7 Inter-group Soccer Competition

Your robot will be expected to compete in an in-class robot soccer competition. Games in this competition will consist of one 7 minute game. You will be expected to play at least 2 games during round robin schedule. This schedule will be provided the day before the demos. **Do not be late for your games as this will delay the remainder of the competition schedule. Failure to participate in the competition will result in a 10% deduction in your grade for the assignment.**

At the beginning of the game after the scoring of a goal, an offensive robot begins play kicking the ball from the center of the field. The kickoff defender must start in their half until the offensive robot strikes the ball. This start could be done manually through starting or keying an event to your robot client. Afterward, no set roles are assigned to the robots. Make sure your client can be set to defend either goal from the command line.

The main play penalty is directly pushing the other player. When such a penalty is assessed, your robot will be taken off the field for 5 seconds and put on their 148 logo when returning. If your robot leaves the field and does not return within 3 seconds, it will be set on the "148" logo on the side of the field it is defending with a 5 second penalty assessed.

8 Grading

Note: Demonstrations of the challenges and the soccer competitions will be held during class and the preceding hour (12-2pm).

Your grade for Assignment 3 will be determined by equal weighting of your group's implementation (50%) and your individual written report (50%). The weighted breakdown of grading factors for this assignment are as follows:

Project Implementation	
- Localization → Does your robot know where it is given overhead fiducial recognition? → Does your robot know where the goal, the ball, and other players are?	5%
- Goal Attainment → Can your robot drive to a given location on the field? → How close to optimal is the robot's path? → How long does it take for your planner to compute?	15%
- Obstacle Avoidance → Can your robot plan paths in the environment to reach a goal without traversing over obstacles? → Can your robot traverse a given path without hitting objects in the environment?	15%
- Soccer Proficiency → How well does your robot player soccer in the given environment?	10%
- Controller Robustness → Does your controller run without interruption or crashing?	5%
Written Report	
- Introduction and Problem Statement → What is your problem? → Why is it interesting?	7%
- Approach and Methods → What is your approach to the problem? → How did you implement your approach and algorithms? → Could someone reproduce your algorithms?	15%
- Experiments and Results → How did you validate your methods? → Describe your variables, controls, specific tests, and results from these test. → How much do the results reflect your observations instead of measurable events? → Could someone reproduce your results?	20%
- Conclusion and Discussion → What conclusions can be reached about your problem and approach? → What are the strengths of your approach? → What are the shortcomings of your approach?	8%