

Developing Robot Programming Lab Projects

Xuejun Liang
 Department of Computer Science
 Jackson State University
 Jackson, MS, USA

F ECS 2012

Outlines

- A. Overview of the Robotics Course
- B. Robot Programming with Player/Stage
- C. Robot Programming with Tekkotsu
- D. Robot Programming Projects
 - ❖ P1. Waypoints Following
 - ❖ P2. Target Searching
 - ❖ P3. Path Planning
 - ❖ P4. Localization and Navigation
 - ❖ P5. Another Version of P4
- E. Conclusion

F ECS 2012

A. Course Overview

- o CSC499/539 Introduction to Robotics.
 - ❖ Elective course for
 - ❖ Both senior undergraduate students and graduate students
- o Prerequisites:
 - ❖ CSC 325 Operating System
 - ❖ CSC 323 Algorithm Design and Analysis
- o Catalog Description
 - ❖ Introduction to robotics and the key artificial intelligence issues involved in the development of intelligent mobile robotics, including software control architectures, localization, navigation, sensing, planning, and uncertainty

F ECS 2012

Course Overview (Cont.)

- o Lectures (3 Hours)
 - ❖ Covering major topics on AI mobile robotics
 - ❖ Reviewing C++ and Introducing robot programming with
 - Player/Stage and Tekkotsu
- o Homework Assignments
 - ❖ Assignments on major robotics topics
- o Robot Programming Projects
 - ❖ Five major robotics programming projects
 - ❖ Real robot (iRobot Create) and simulation


F ECS 2012

Course Overview (Cont.)

- o Midterm and Final Examinations
- o Required Textbook
 - ❖ Introduction to AI Robotics, by Robin R. Murphy
- o Reference Book
 - ❖ The Robotics Primer, by Maja J Mataric
 - ❖ Behavior-Based Robotics, by Ronald C. Arkin
- o Course Website
 - ❖ <http://www.jsums.edu/robotics/>

F ECS 2012

iRobot Create/ASUS Netbook Platform

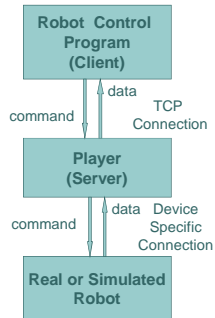


- o Differential Drive
- o Buttons
 - ❖ Power, Play, Advance
 - ❖ Wheel Drops (Front, Left, Right)
 - ❖ Bumps (Left, Right)
- o Sensors
 - ❖ Wall, IR
 - ❖ Cliffs (Left, Front Left, Right, Front Right)
 - ❖ Encoders (Distance, Angle)

F ECS 2012 Leds

B. Player Sever for Robot Control

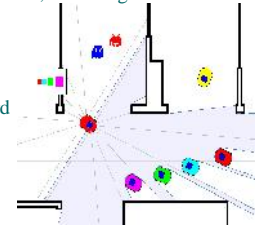
- Player provides a **network interface** to a variety of real or simulated robots and sensor hardware.
- Player's **client/server** model allows robot control programs to be written in any programming language and to run on any computer with a network connection to the robot
- Player supports multiple concurrent client connections to devices



FECs 2012

Stage Simulator

- Stage simulates a population of mobile robots moving in and sensing a two-dimensional bitmapped environment.
- Various sensor models are provided, including
 - Sonar, scanning laser rangefinder, pan-tilt-zoom camera with color blob detection and odometry
- Stage devices present a standard Player interface.
 - Few or no changes are required to move between simulation and hardware



FECs 2012

Two Important Files

- World (.world) file**
 - Needed when doing simulation using Stage
 - Things are available in the world, including robots, items, and layout of the world
- Configuration (.cfg) file**
 - Robot information: Drivers
 - Real robot driver is build in Player already
 - Simulation driver is always Stage
 - Items in .world file if your code interacts with them.

FECs 2012

Creating C++ Client Program

- Server/Client Control Structure**
 - Player is server, your program is client
 - Your code controls hardware on robot through something called **proxy**.
- Program structure**
 - Include the Player header file
 - Establish a Player client
 - Connect your code to the device proxies
 - Control Loop (Interacting with Proxies)

FECs 2012

An Example of Client Program: Random Walk

```
int main(int argc, char *argv[])
{
    int randcount = 0;
    double speed, turnrate;
    Vector random(0,0);

    PlayerClient robot("localhost");
    Position2dProxy pp(&robot, 0);
    pp.SetMotorEnable (true);
    while(true)
    {
        // update the proxies
        robot.Read();

        // generate a random vector
        wander(randcount, random);

        // compute the speed and the turn rate
        translate(random, speed, turnrate);

        // command the motors
        pp.SetSpeed(speed, turnrate);
    }
}
} FECs 2012
```

C. Tekkotsu Framework

- It provides lower level primitives for
 - Sensory processing,
 - Smooth control of effectors, and
 - Event-based communication
- It provides higher level facilities, including
 - A hierarchical state machine formalism for managing control flow in the application,
 - A vision system,
 - An automatically maintained world map, and
 - Newly added Tekkotsu crew, including localization and path planning, etc.
- It provides housekeeping and utility functions, such as timers and profilers

FECs 2012

Programming with Tekkotsu

- Application programmer simply define subclasses that inherit from the Tekkotsu base classes, and override any member functions requiring customization
- Behaviors and Events**
 - Behaviors: Construct, Activate, Deactivate, Listen Events, Process Events,
 - Events: Generator, Source, Type
- State Machine**
 - Robot moves from state to state.
 - Each state has an associated action: speak, move, etc.
 - Transitions triggered by sensory events or timers.

F ECS 2012

Tekkotsu State and Transition

- State nodes are behaviors**
 - Enter (Activate), Leave (Deactivate), Listen Events, Process Events
- Transitions are also behaviors**
 - A transition starts to work whenever its source state node becomes active.
 - Transitions listen for sensor, timer, or other events, and when their conditions are met, they fire.
 - When a transition fires, it deactivates its source node(s) and then activates its destination node(s).

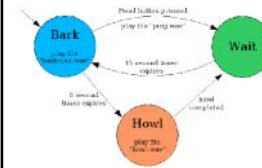
F ECS 2012

Shorthand Notation

- A shorthand notation is used instead of C++ code to build state machines. The shorthand is turned into C++ by a state machine compiler.
- Node definition:
`label: ClassName`args)[inits]`
- Transition definition:
`=label:TransAbbrev`args)[inits]=>`
- Node Class definition
`$nodeclass ClassName(parameters) : ParentName(args) :`
`initializers : methodname {`
`body`
`}`

F ECS 2012

An Example: Annoying Dog



```

#include "Behaviors/StateMachine.h"
$nodeclass AnnoyingDog : StateNode {
  virtual void setup() {
    $statemachine{
      launch: StateNode
      bark: SoundNode($, "barkmed.wav")
      howl: SoundNode($, "howl.wav")
      wait: StateNode
      launch =N=> bark
      bark =T(5000)=> howl
      bark =B(RobotInfo::GreenButOffset)
      [setSound("ping.wav")]=> wait
      wait =T(15000)=> bark
      howl =C=> wait
    }
  }
}
REGISTER_BEHAVIOR(AnnoyingDog);
  
```

F ECS 2012

Advanced Knowledge of Tekkotsu

- Transit from one state to multiple states simultaneously so as to support parallel actions or behaviors,
- Transit from one state to one of multiple states based on different conditions so as to make a conditional transition, and
- Pass and/or share data among states so as to provide approaches of the data flow and the memory

F ECS 2012

D. Robot Programming Projects

Projects

- P1. Waypoints Following
- P2. Target Searching
- P3. Path Planning
- P4. Localization and Navigation
- P5. Another Version of P4

Discussing Aspects

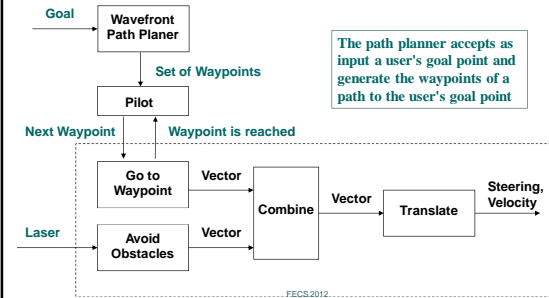
- Task and steps to accomplish the task,
- Programming model (structure) and skills,
- Concepts, algorithms, and mathematical formulas,
- Issues regarding to failures, uncertainty, and real-time constraints

F ECS 2012

Target Searching (Cont.)

- o Mathematical formulas and Concepts
 - ❖ Generate a random number in a given range
 - ❖ Compute distance and angle
 - ❖ Coordinate Transformation (Rotation and Shift)
 - ❖ Vector addition
- o Real-time constraints
 - ❖ Set up a distance and an direction to go for about 2 seconds
 - ❖ Generate a random distance and a random direction every 3 seconds
- o Programming skills
 - ❖ List operations
 - ❖ Vector class
 - ❖ Pass-by-value V.S. Pass-by-reference

P3. Path Planning



Wavefront Path Planer

- o Read the input map into the grid map
 - o Convert starting and goal points from the world coordinate into the image coordinate
 - o Check if the goal point is in the wall
 - o Label the goal point on the grid map
 - o Grow the grid map a few times
 - o Propagate the wavefront until reaching to starting point or until the wavefront can not be propagated
 - o Compute waypoints
 - o Relax waypoints
 - o Store waypoints in the grid map
 - o Print the grid map into the output map
 - o Convert waypoints from the image coordinate into the world coordinate and then return them
- FECs 2012

Wavefront Path Planer (Cont.)

- o The occupancy grid (Study the functions `inputMap` and `outputMap`)
 - ❖ The `inputMap` function reads in the map into a two-dimensional binary occupancy grid array in which 0's represent free space and 1's represent obstacles (walls).
 - ❖ The `outputMap` function writes out the occupancy grid information into a scaled map, which can be used to visualize your planner's results
 - o The conversion between the World Coordinates and the Image Coordinates (Study the functions `computeHW` and `computeXY`)
- FECs 2012

Wavefront Path Planer (Cont.)

- o The function `grow()`
 - ❖ This function grows the obstacles in the grid map for a single step, i.e. one grid cell farther
 - o The function `nextWaypoint(...)`
 - ❖ This function computes the next waypoint.
 - ❖ The next waypoint is a neighboring cell of the current waypoint and its value is 1 less than the value of the current waypoint cell.
 - ❖ This function is called first with the current waypoint equal to the robot's starting point.
 - ❖ It is called continuously until the new waypoint is the goal point.
- FECs 2012

Wavefront Path Planer (Cont.)

- o The function `propagate(...)`
 - ❖ This function propagates the wavefront one grid cell farther. It starts from the grid cells with value `i` and the propagated cells get the value `i+1`.
 - ❖ If the cell of robot's starting point has the same value of the start parameter, then the wavefront propagation should be over (`done = 1`).
 - ❖ Otherwise, propagate the wavefront to its neighbors.
 - ❖ if there is no room to propagate the wavefront, then the robot's goal point is unreachable (`done = 2`);
 - ❖ otherwise, the wavefront can be propagated continuously (`done = 0`)
- FECs 2012

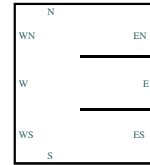
Wavefront Path Planer (Cont.)

- o Mathematical formulas and Concepts
 - ❖ Grid map representations
 - ❖ Conversion between the World Coordinates and the Image Coordinates
 - ❖ The wavefront path planning algorithm
- o Programming skills
 - ❖ Queue
 - ❖ Two-dimensional array
 - ❖ Pass-by-value V.S. Pass-by-reference

F ECS 2012

P4. Localization and Navigation

- o This is the task for the 2010 Robotics Competition held along with the 2nd Annual ARTSI Student Research Conference
- o The task is to get a robot to localize itself within a maze, navigate efficiently between locations, observe objects in the maze, and report on what it has observed.



F ECS 2012

P4 (Cont.): Milestones

- o What students need to know first
 - ❖ Following wall
 - ❖ Recognizing color balls (**objects in the maze**)
 - ❖ Detecting and visiting bi-color navigation marks
 - ❖ Memorizing what objects have been seen already
- o Milestones
 - ❖ Travel though the maze by using "following wall".
 - Able to start "follow wall behavior" and able to stop
 - ❖ Travel though the maze and report detected color balls
 - Report only once per ball, so need memory
 - ❖ Travel though the maze and report detected color balls and their locations

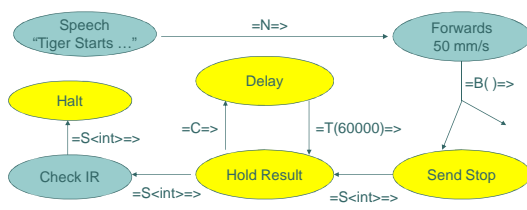
F ECS 2012

P4 (Cont.): Skills and Challenges

- o Basic Programming Skills
 - ❖ Instantiate a class
 - ❖ Derive a subclass
- o Tekkotsu Programming Challenges
 - ❖ New state machine programming language
 - ❖ New semantics of state machine
 - ❖ Setup computer camera setting for a specific light condition
- o Uncertainty and Failure
 - ❖ Bi-color markers are difficult to detect
 - ❖ Dealing with false color balls

F ECS 2012

P4 (Cont.): Follow Wall



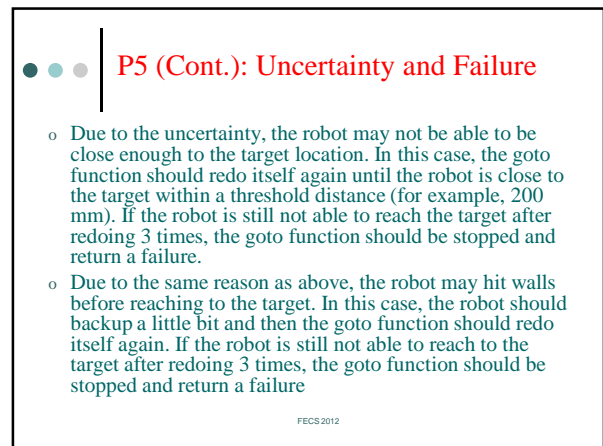
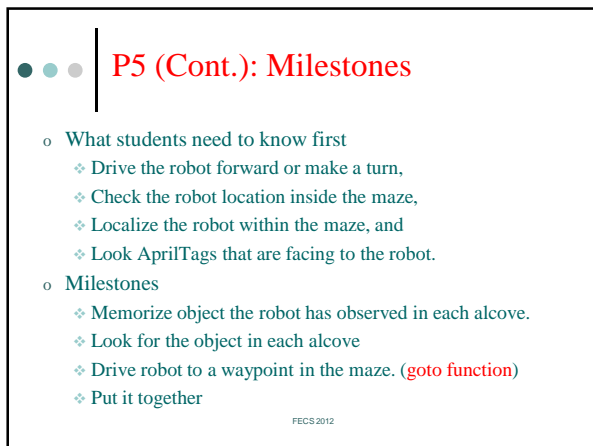
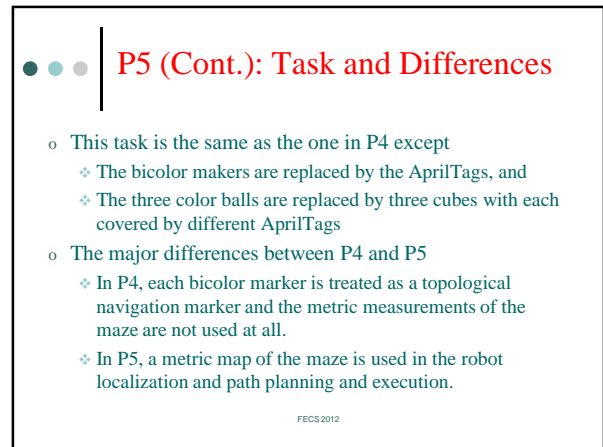
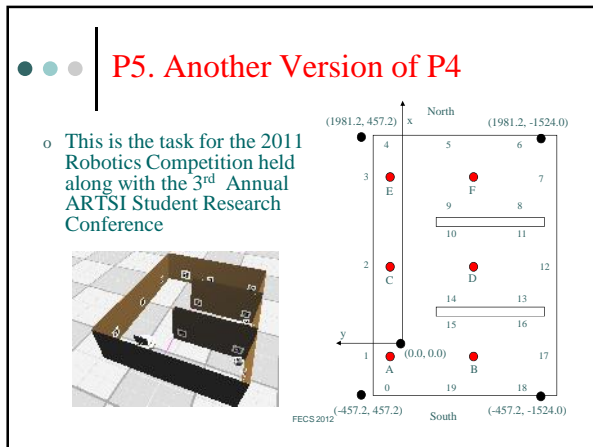
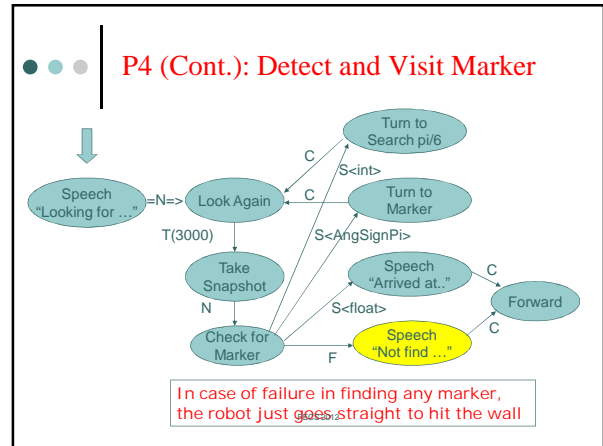
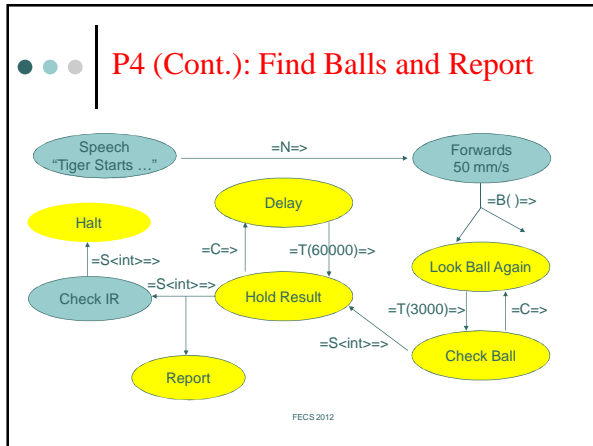
The robot can be placed in anywhere and can be stopped after a given period of time

F ECS 2012

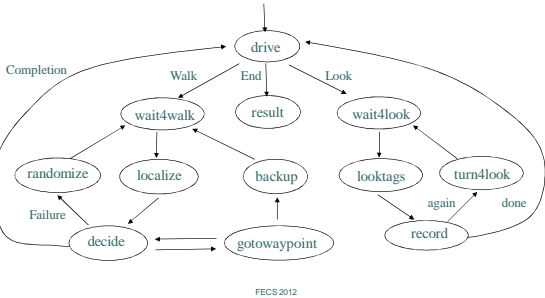
P4 (Cont.): Memorize Color Balls

- o Color ball encoding:
 - ❖ Orange(Red): 1
 - ❖ Blue: 2
 - ❖ Green: 4
- o The color ball combinations can be encoded as follows
 - ❖ {Red, Blue} = 3,
 - ❖ {Red, Green} = 5,
 - ❖ {Blue, Green} = 6,
 - ❖ {Red, Blue, Green} = 7
- o The color ball sequences can be encoded as follows
 - ❖ (Red, Blue, Green)= 124,
 - ❖ (Blue, Green, Red) = 241, and etc.

F ECS 2012



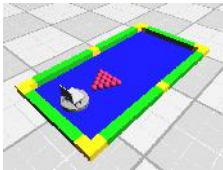
P5 (Cont.): Put It Together



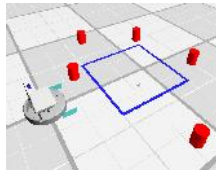
Conclusion

- o The robotics course is for both undergraduate students and graduate students
 - o It covers major topics on intelligent mobile robots, including robot control architectures, localization, navigation, sensing, planning, and uncertainty
 - o Robot programming is a key component of this course
 - o Five major robot programming projects have been adopted, developed, and taught
 - o More teaching resources are needed such as textbook, robot programming materials, etc.
- FECS 2012

Moore Projects Using Tekkotsu



Shoot balls



Move canisters into a "pen"

FECS 2012