

# LAB 1 RECITATION

Getting to Know Your Robot: Locomotion and Odometry

Demo, Code and Memo Due Thursday, 3/12/10

ECE497 Lab 1 Recitation: Getting To Know Your Robot

1



## PURPOSE

- Introduce locomotion and odometry concepts
- Install all necessary software and connect to the Traxster II
- Introduction to Visual C# programming and IDE
- Use the demo code to confirm that all of the actuators, sensors and peripherals on your robot are working
- Create your first program to get the robot moving
- Examine problems with raw odometry for pose estimation

ECE497 Lab 1 Recitation: Getting To Know Your Robot



## LOCOMOTION

- Locomotion refers to the way that a robot moves from place to place
- In *locomotion*, the environment is fixed and the robot moves by impart force on the environment
- Wheels are more energy efficient than legs and simpler to control and the effector of choice in robotics
- Wheel may not necessarily be holonomic
- Holonomic means that the robot can control all of its available degrees of freedom



## STEERING

- The ability to drive wheels separately and independently through separate motors is differential drive
- The ability to steer wheels independently is differential steering
- If both wheels are driven at the same speed the robot moves forward or backward
- If the wheels have the same speed but opposite direction, the robot spins
- If one wheel is driven faster than the other, the robot moves in a circle or turns

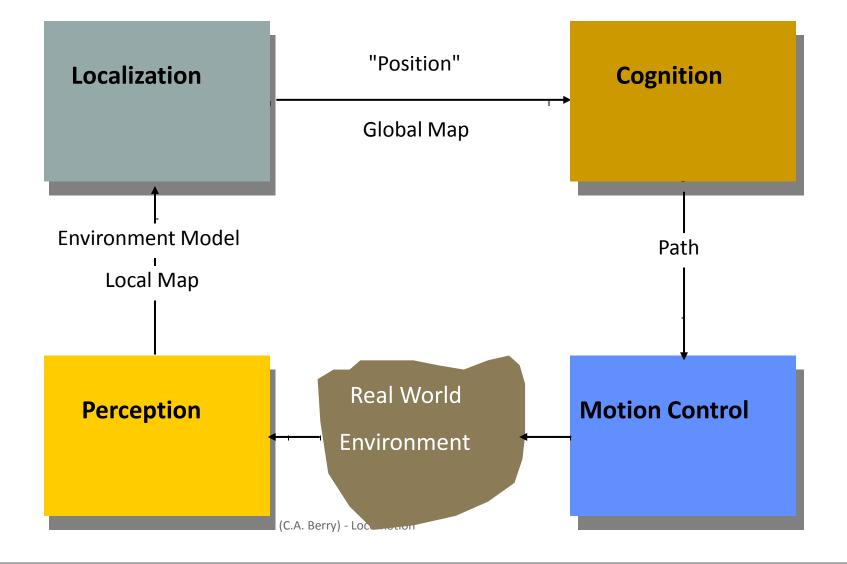


#### **TRAJECTORY AND MOTION PLANNING**

- Two concerns in locomotion
  - Getting the robot to a particular location (goal)
  - Having the robot follow a trajectory
- *Navigation* is concerned with getting to a goal
- Trajectory planning (motion/path planning) is more difficult than moving the robot to a particular location. This is related to forward and inverse kinematics.
- Optimal trajectory deals with finding the safest, shorts, or most efficient path

## LOCOMOTION CONCEPTS: PATH PLANNING







# **MOBILE ROBOT KINEMATICS**

- *Mobile robot kinematics* is the dynamic model of how a mobile robot behaves
- Kinematics is a description of mechanical behavior of the robot for design and control
- Mobile Robot Kinematics is used for:
  - Position estimation
  - Motion estimation
- Mobile robots move unbounded with respect to their environment
  - There is no direct way to measure robot position
  - Position must be integrated over time
  - The integration leads to inaccuracies in position and motion estimation



## **ODOMETRY**

- Odometry is a means of implementing Dead Reckoning
- A way of determining a robot's position based upon previous known position information given a specific course heading and velocity
- Periodically requires error measurement to be 'fixed' or reset
- Meant for short distance measurements



#### RELATIVE POSITIONING: ODOMETRY AND KINEMATICS

- Given wheel velocities at any given time, compute position/orientation for any future time
- Advantages
  - Self-contained
  - Can get positions anywhere along curved paths
  - Always provides an "estimate" of position
- Disadvantages
  - Requires accurate measurement of wheel velocities over time, including measuring acceleration and deceleration
  - Position error grows over time



# **ODOMETRY ERRORS**

- Systematic
  - Unequal wheel diameters
  - Misalignment of wheels
  - Finite encoder resolution
  - Finite encoder sampling rate
- Non-systematic
  - Travel over uneven floors
  - Unexpected objects in the floor
  - Wheel slippage due to
    - Over acceleration
    - Slippery floor
    - skidding



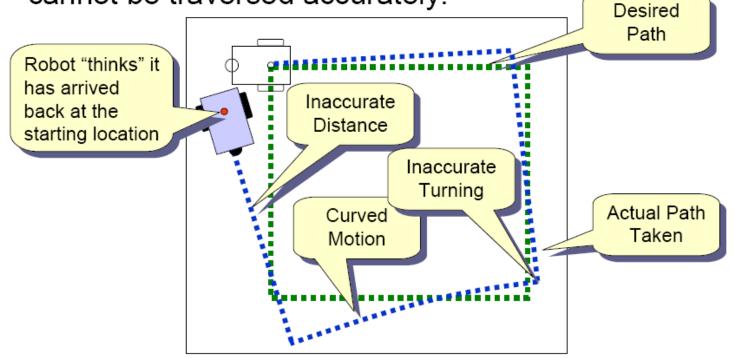
# **ODOMETRY ERRORS**

- Imprecise measurements
  - Discrepancy with actual speed and turn angles
- Inaccurate control model
  - Tracks/Wheels/Motors are not perfectly aligned or do not make contact at a single point
- Immeasurable physical characteristics
  - Friction
  - Wobbling wheels
  - Surface is not perfectly smooth and hard
  - Sliding



#### **DEAD RECKONING**

 As a result of these error factors, a simple path cannot be traversed accurately.





# **OPEN LOOP CONTROL**

- Open Loop Control does not use sensory feedback, and the robot state is not fed back into the system
- Feed-forward control
  - The command signal is a function of some parameters measured in advance
- Feed-forward systems are effective only if
  - They are well calibrated
  - The environment is predictable and does not change



#### PROGRAMMING

- This is not a programming class but programming is an integral part of this course
- The best way to learn to program in any language is to practice, practice, practice
- The best reference for Visual C# is the online MSDN library, programming books and websites



## VISUAL C #

- Object-oriented language with a class library of pre-built components
- Event-driven visual programming language created using an IDE
- Program responds to timer expirations and user events such as mouse clicks and keystrokes
- The toolbox can be used to drag and drop objects such as labels, textboxes and buttons onto the windows application

|                                 | 14  |
|---------------------------------|-----|
| 🗉 All Windows For 🔺             | 17  |
| Common Controls                 | 8   |
| 🕅 Pointer                       |     |
| 🐵 Button                        | L . |
| CheckBox                        | L . |
| 🔝 CheckedListBox                | L . |
| 📑 ComboBox                      | L . |
| 📅 DateTimePicker                | L . |
| A Label                         | L . |
| 🔺 LinkLabel                     | L . |
| 🗈 ListBox                       | L . |
| 📰 ListView                      | L . |
| 😐 MaskedTextBox                 | L . |
| 🔝 MonthCalendar                 | L . |
| 🔤 NotifyIcon                    | L . |
| 🔝 NumericUpDown                 | L . |
| 📓 PictureBox                    | L . |
| 🚥 ProgressBar                   | L . |
| <ul> <li>RadioButton</li> </ul> | L . |
| 🔩 RichTextBox                   | L . |
| 🛯 TextBox 🚽 🚽                   | 1   |
| 🎍 ToolTip                       |     |
| 🔃 TreeView                      |     |
| 🗔 WebBrowser                    |     |
|                                 |     |

Toolbo)



## **PRELIMINARY STUFF**

- All course materials are on the course Angel folder
- Does your laptop have?
  - Microsoft Visual Studio with C#
  - Bluetooth radio
- Create a course lab folder on your computer
- Download
  - Demo\_App.zip
  - Unzip all files to the Lab 1 folder



#### **VISUAL C# IDE**

- Start Page
  - Displays IDE and webbased resources
  - Links to recent projects and getting started
- View
  - Solution Explorer
  - Toolbox
  - Properties

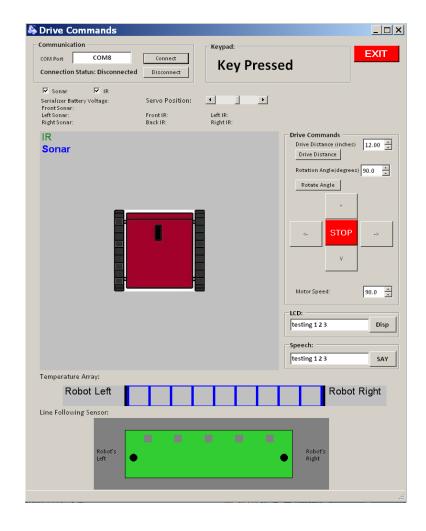
- Open the project Demo\_App\IRSonar\_App\Demo\_App.sIn
- Select tab DemoForm.cs
  - View Code
  - View Designer
  - Build the solution (F7)
  - Run the solution (F5)



## **DEMO APP GUI**

- Use this application to test all of the robot's actuators, sensors and peripherals
- Controls or displays
  - Communication
  - Servo
  - IR
  - Sonar
  - Drive Commands
  - LCD
  - Speech
  - Keypad
  - Temperature Array
  - Line Following Sensor







# **DEMO APP CODE (HEADING)**

- All code must have a heading with the name of the solution, a description of the functionality, the names of the authors and dates of last revision. Use single-line comments (//) or delimited comments (/\*)
- The *using* directive tells the compiler where to look for a class in a *namespace* which includes a collection of related classes

| Demo_App - Microsoft Visual          | Studio   |  |
|--------------------------------------|--|--|
| File Edit View Refactor Project Buil | d Debug Data Tools Test Window Help  |  |
| 🛅 • 🖽 • 📂 📕 🥔 👗 🛍 🖄 🕫 • 🖓 •          | 🗉 - 🖳 🕨 Debug 🔍 Any CPU 🚽 🎯 console.write 🚽 💐 🐨 🖄 📯 🛃 🖬 - 🔤 i 🕮 🛗 i 📥 🖕  |  |
| 113 🐁 🖢 🚈 (第 第 1 🗄 🖆 💭 🗣 😂 (1        |  |  |
| Solution Explorer - Demo App 🚽 🕂     |  |  |
|                                      | / Demoi of mitts [Demoirof mitts [Design]   Start Page   |  |
| Solution 'Demo_App' (1 project)      | Periode App.DemoForm     Serializer  |  |
| Berno App                            | //Demo_App.sln   |  |
| E- Properties                        | //Traxster II Demonstration Application  |  |
|                                      | //Modified from Robotics Connection original serializer library code   |  |
| - 🕙 AssemblyInfo.cs                  | //this GUI attempts to encapsulate all of the Traxster II sensors, peripherals and actuators   |  |
| 🖶 📾 Resources.resx                   | <pre>//in one interface for testing and demonstration purposes. These components //include the LCD screen (LCD03), speech syntehsizer (SP03), sonar, infrared sensor</pre> |  |
| 🗉 📄 Settings.settings                | //include the LCD screen (LCDO3), speech syntensizer (SPO3), sonar, infrared sensor<br>//motor control, line following sensor, servo, and temperature array.               |  |
| 🖅 🔄 References                       | //.A. Berry - Summer 2009 - Winter 2010  |  |
| - Service References                 | //The SPOSEX and LCDO3 classes and related code were written by  |  |
|                                      | //J. Nibert - Summer 2009  |  |
| 🖷 🗀 bin                              | //Future work will include the integration of the CMU camera and turret (cmuCam2.cs)   |  |
| 🖷 🗀 obj                              |  |  |
| 😹 ClassDiagram1.cd                   | using System;  |  |
| E- III DemoForm.cs                   | using System.Collections;  |  |
| DemoForm.Designer.cs                 | using System.Collections.Generic;<br>using System.ComponentModel;  |  |
|                                      |  |  |
| 🐿 DemoForm.resx                      | using System.Data;   |  |
| Program.cs                           | using System.Drawing;<br>using System.Ling;  |  |
| - 🔳 RoboImage.ico                    | using System. Bind;<br>using System. Threading;  |  |
| 🖩 RobotImage.ico                     | using System.Text:   |  |
|                                      | using System.Timers;   |  |
|                                      | using System. Windows.Forms:   |  |
|                                      | using RoboticsConnection;  |  |
|                                      | using RoboticsConnection.Serializer;   |  |
|                                      | using RoboticsConnection.Serializer.Sensors;   |  |
|                                      | using RoboticsConnection.Serializer.Controllers;   |  |
|                                      | using RoboticsConnection.Serializer.Components;  |  |
|                                      | using RoboticsConnection.Serializer.Ids;   |  |
|                                      | Lusing CustomComponents; //LCD and speech  |  |
|                                      |  |  |



# **DEMO APP CODE (BODY)**

- Visual Studio generates some code automatically that creates and initializes the GUI
- You will insert your variables in the public partial class
- You will insert your code in the form() function
- Define the required sensor objects

- Define the timer for the controller events (100 ms)
- Timer also used for keyPad presses
- Define the event handlers for the objects
- Feel free to modify this code and eliminate unnecessary code or GUI objects or create your own GUI from scratch



# **DEMO APP (DRIVE COMMAND)**

- There are two drive commands
  - dmc-differential motor control
  - pmc PID motor control
- The PID motor control provides feedback from the encoders and should be more accurate

- For driving a distance use
  - TravelAtSpeed with sleep or
  - Distance and TravelDistance()
- For turning an angle
  - TravelAtSpeed with sleep or
  - RotationAngle and Rotate()