Georgia Tech School of Electrical and Computer Engineering
College of Engineering

GRITS Georgia Robotics and InTelligent Systems Laboratory

Differential-Drive, Odometry, and IR

Control of Mobile Robots: Programming & Simulation Week 2

Jean-Pierre de la Croix
ECE Ph.D. Candidate
Georgia Inst. of Technology

Overview

- The purpose of this week's programming assignment is to implement the functions for the robot to move and sense.
  1. Transform the outputs of our controllers to the control inputs of the mobile robot.
  2. Keep track of where the robot is located.
  3. Convert raw sensor values to distances.

QuickBot

- Mobile Robot

`+simiam/+robot/QuickBot.m`

- Two-wheel differential drive (1 motor per wheel)

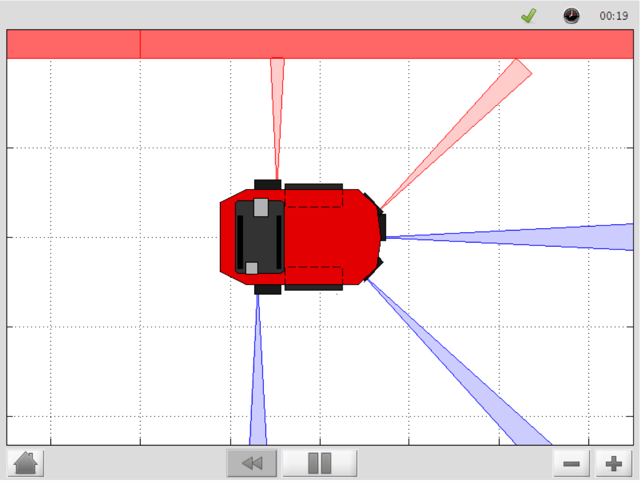
`+simiam/+robot/+dynamics/DifferentialDrive.m`

- Wheel encoders for each wheel with 32 ticks/rev.

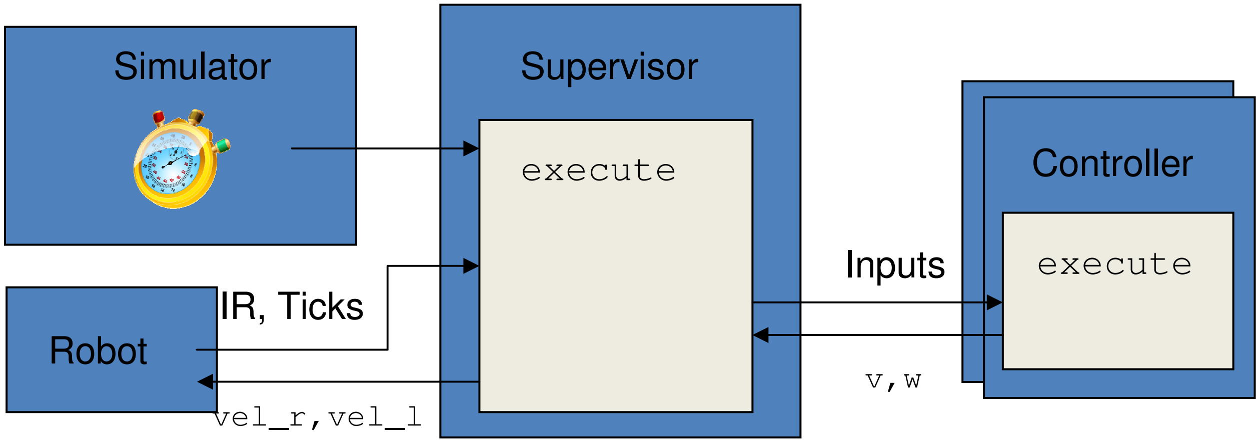
`+simiam/+robot/+sensor/WheelEncoder.m`

- IR distance sensors with a range of 4cm to 30cm.

`+simiam/+robot/+sensor/ProximitySensor.m`



Simulation



- Supervisor

`+simiam/+controller/+quickbot/QBSupervisor.m`

- Controllers

`+simiam/+controller/GoToAngle.m`

Differential Drive

- Our controllers output (v,ω), but the robot can only be controlled by specifying left/right angular wheel velocities!

```
+simiam/+robot/+dynamics/DifferentialDrive.m

function [vel_r,vel_l] = uni_to_diff(obj,v,w)
% Make sure to fix this transformation!
    R = obj.wheel_radius;              % 31.9mm
    L = obj.wheel_base_length;         % 99.25mm

    vel_r = 0;          % in terms of v,w
    vel_l = 0;          % in terms of v,w
 end
```

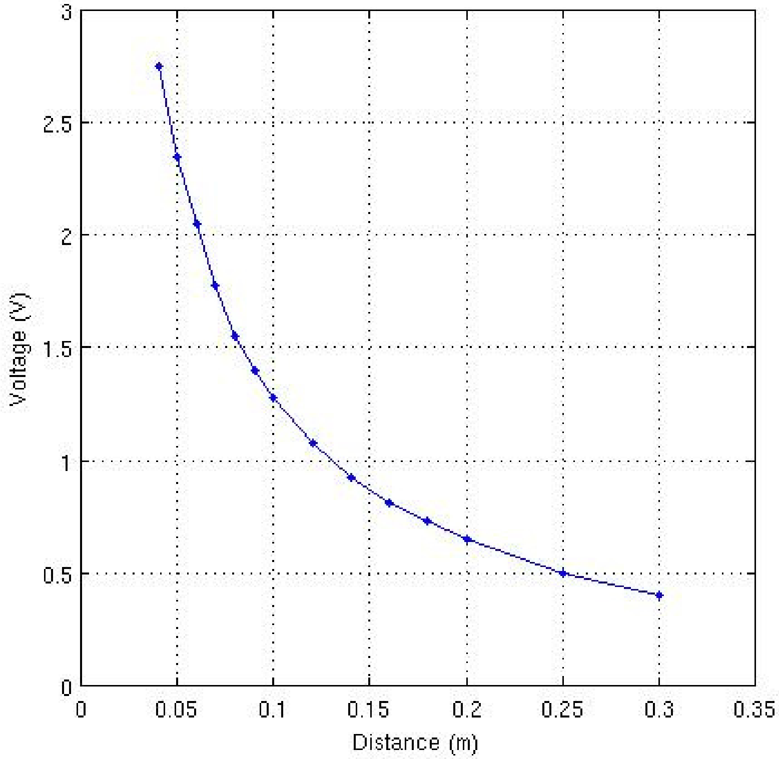
Odometry

- Odometry keeps track of the robot’s position using wheel encoders to measure the distance travelled by each wheel.

```
+simiam/+controller/+quickbot/QBSupervisor.m

function update_odometry(obj)
%% UPDATE_ODOMETRY Approximates the location of the robot.
% obj.update_odometry() should be called from the
% execute function every iteration. The location
% of the robot is updated based on the
% difference to the previous wheel encoder
% ticks.
%
% This is only an approximation.
```

IR Distance Sensors



- IR sensors return a voltage in the range [0.4,2.75] V for distances from [4,30] cm.
- 12-bit, 1.8V ADCs convert voltage into integers in the range [200, 1375].

Computing Measured IR Distances

- Use polyfit and polyval to convert raw IR values to distances from fitted data.

+simiam/+robot/QuickBot.m

```
function ir_distances = get_ir_distances(obj)
    ir_array_values = obj.ir_array.get_range();
    % Convert from bits to volts
    ir_array_voltage = ir_array_values;
    % Hardcode the coefficients from polyfit
    coeff = zeros(1,5);
    ir_distances = polyval(coeff, …
                              ir_array_values);
end
```

Testing

- We have included a Go-to-Angle controller, which steers the robot to a specified angle if everything is implemented correctly.

```
+simiam/+controller/+quickbot/QBSupervisor.m

function obj = QBSupervisor()
   %% SUPERVISOR Constructor
   obj = obj@simiam.controller.Supervisor();
   [ … ]
   obj.v = 0.1;
   obj.theta_d = pi/4;
```

Tips

- Refer to the section for Week 2 in the manual for more details!
- Use the commented out fprintf statements or add your own for debugging.