Obstacle Avoidance

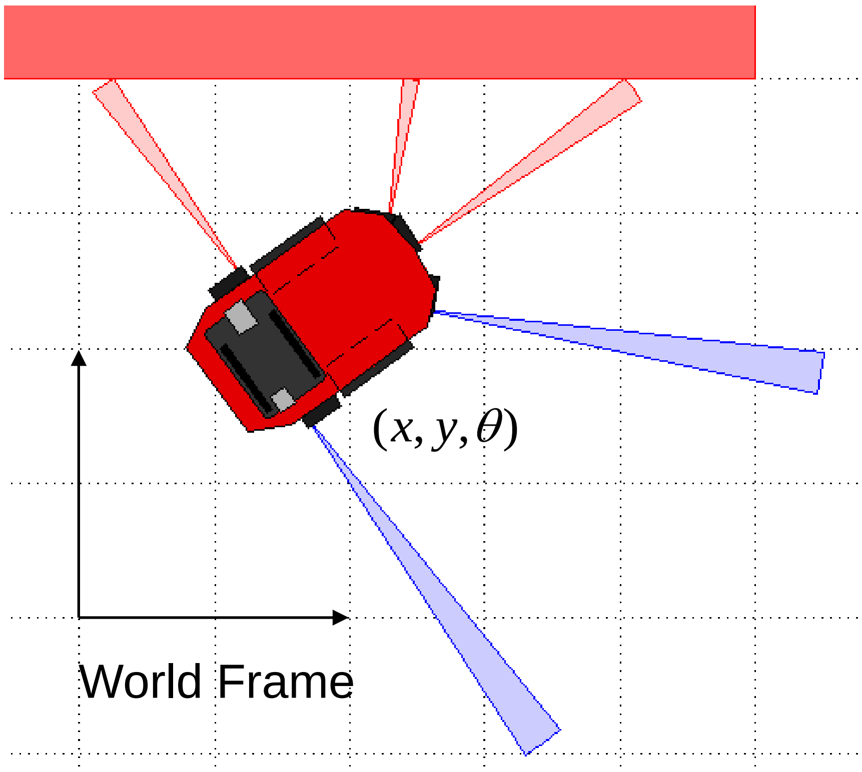
Control of Mobile Robots: Programming & Simulation Week 4

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

Overview

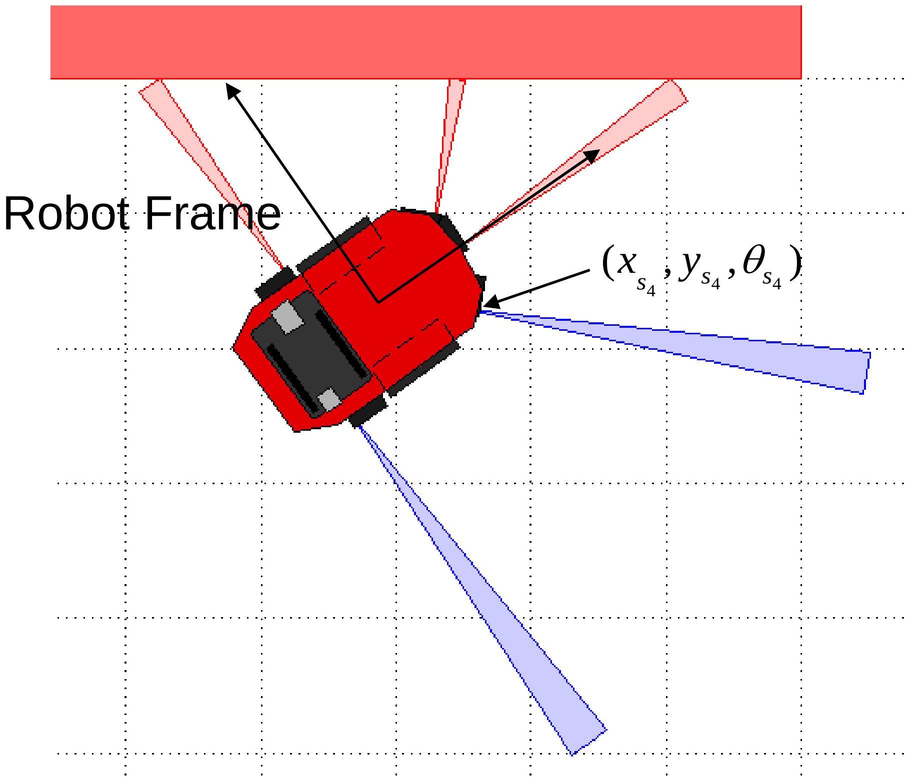
- The purpose of this week's programming assignment is to avoid any obstacles near the robot.
  1. Transform the IR distance to a point in the robot's coordinate frame.
  2. Transform this point from the robot's coordinate frame to the world coordinate frame.
  3. Compute a vector that points away from any obstacles.

Coordinate Frames



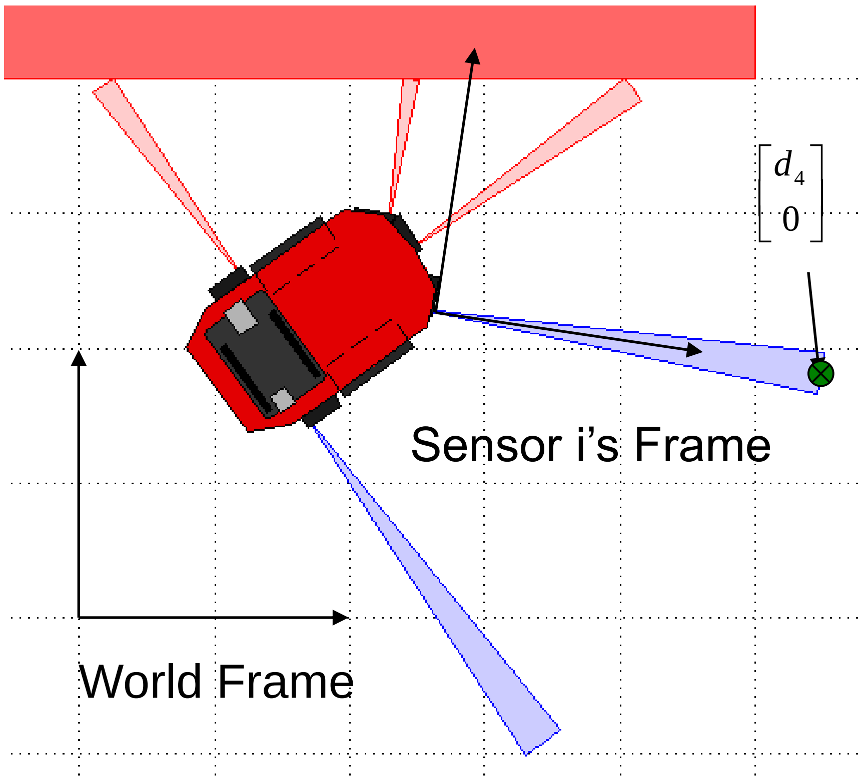
- Robot's coordinates are in the world frame.

Coordinate Frames



- Sensor’s coordinates are in the robot’s frame (centered at robot with robot’s orientation).

Coordinate Frames



- IR distance is defined in a sensor's frame (centered at sensor, with sensor's orientation)

Rotation and Translation in 2D

$$R(x', y', \theta') = \begin{bmatrix} \cos(\theta') & -\sin(\theta') & x' \\ \sin(\theta') & \cos(\theta') & y' \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x_{d_i} \\ y_{d_i} \\ 1 \end{bmatrix} = R(x, y, \theta) R(x_{s_i}, y_{s_i}, \theta_{s_i}) \begin{bmatrix} d_i \\ 0 \\ 1 \end{bmatrix}$$

Obstacle Avoidance

- We will implement the obstacle avoidance in a new controller.

+simiam/+controller/AvoidObstacles.m

```
classdef AvoidObstacles < simiam.controller.Controller
%% AVOIDOBSTACLES steers the robot away from
% any nearby obstacles (i.e., towards free space)
```

Transforming the IR distances

```
function ir_distances_wf = apply_sensor_geometry(obj, ir_distances,
    state_estimate)

  % 1. Apply the transformation to robot frame.
  ir_distances_rf = (3,numel(ir_distances));
  for i=1:numel(ir_distances)
    x_s = obj.sensor_placement(1,i);
    y_s = obj.sensor_placement(2,i);
    theta_s = obj.sensor_placement(3,i);

    %% START CODE BLOCK %%
    R = obj.get_transformation_matrix(0,0,0);
    ir_distances_rf(:,i) = zeros(3,1);
    %% END CODE BLOCK %%
  end
```

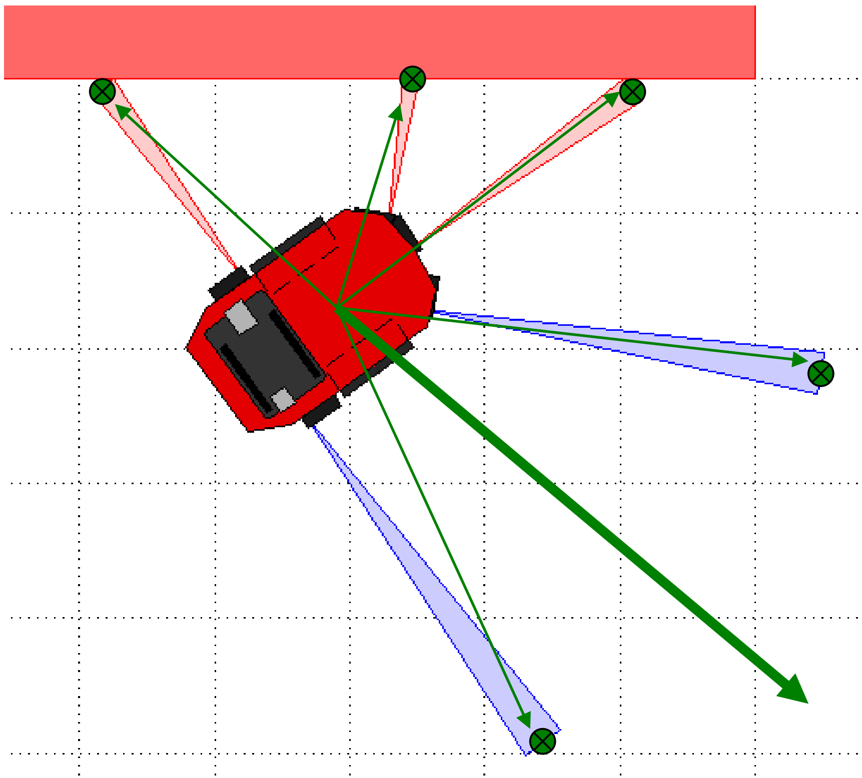
Transforming the IR distances

```
% 2. Apply the transformation to world frame.

[x,y,theta] = state_estimate.unpack();

%% START CODE BLOCK %%
R = obj.get_transformation_matrix(0,0,0);
ir_distances_wf = zeros(3,9);
%% END CODE BLOCK %%
```

Summing up Vectors



- Sum up the vectors from robot to the transformed IR distances (`u_i`).
- PID controller steers robot to the orientation (`theta_ao`) of this vector, `u_ao`.

Computing u_ao and theta_ao

```
%% START CODE BLOCK %%

% 3. Compute the heading vector

sensor_gains = [1 1 1 1 1];
u_i = zeros(2,5);
u_ao = sum(u_i,2);

% Compute the heading and error for the PID controller
theta_ao = 0;
e_k = 0;

%% END CODE BLOCK %%
```

Testing

- Robot should wander aimlessly around the world without colliding with any obstacle or the walls.

Tips

- Refer to the section for Week 4 in the manual for more details!
- Keep in mind that this hard work will pay off next week!