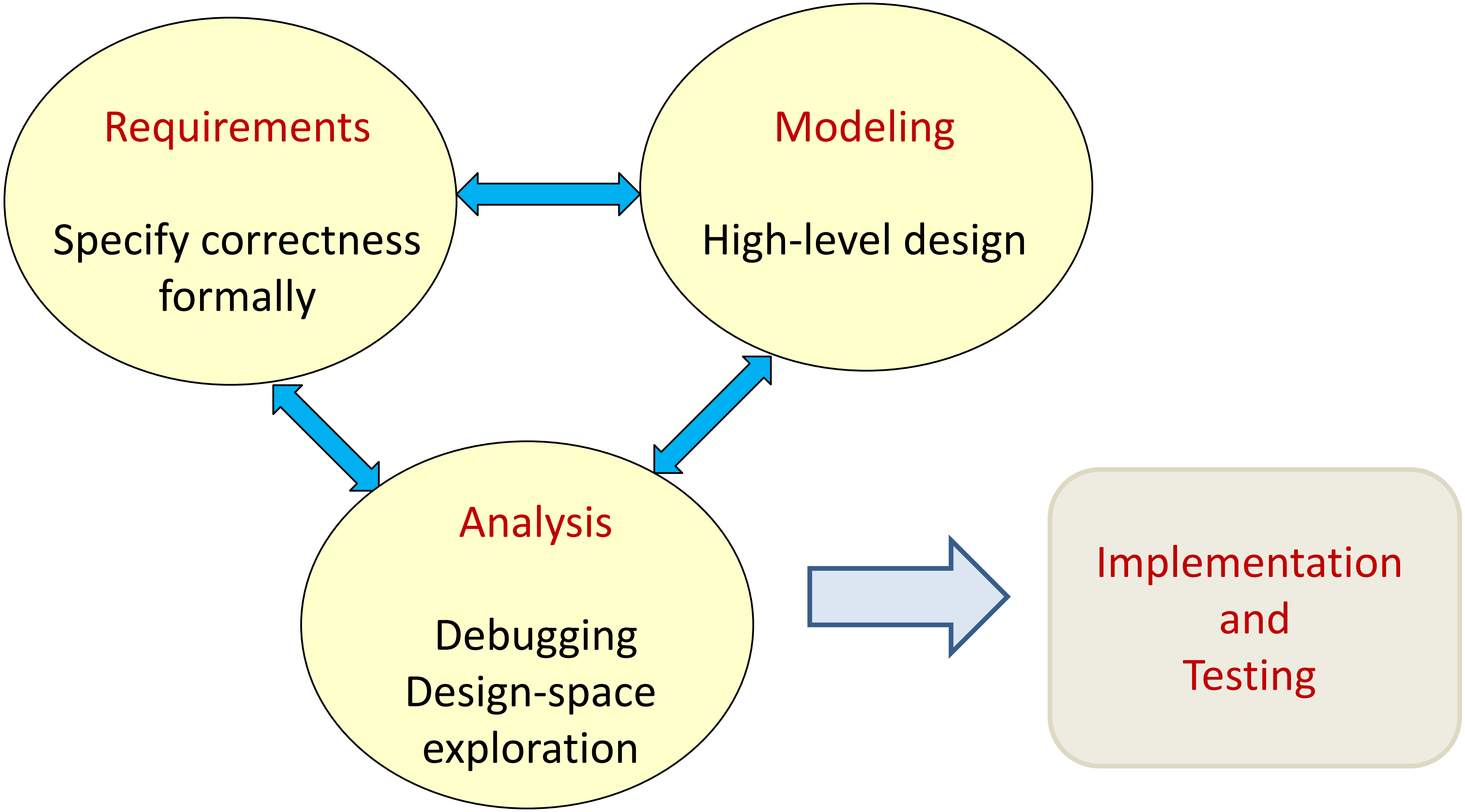
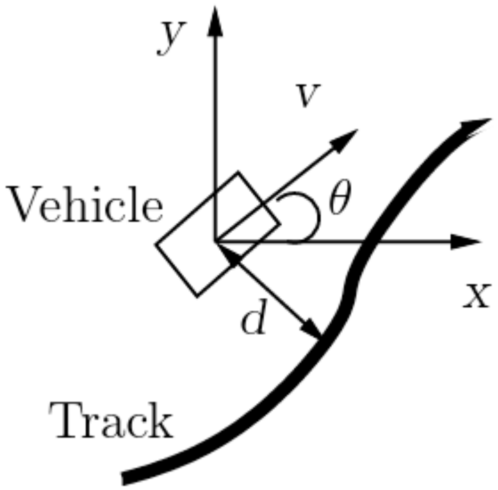
CS:4980
Foundations of Embedded Systems

Hybrid Systems
Part II

Model-Based Design and Analysis

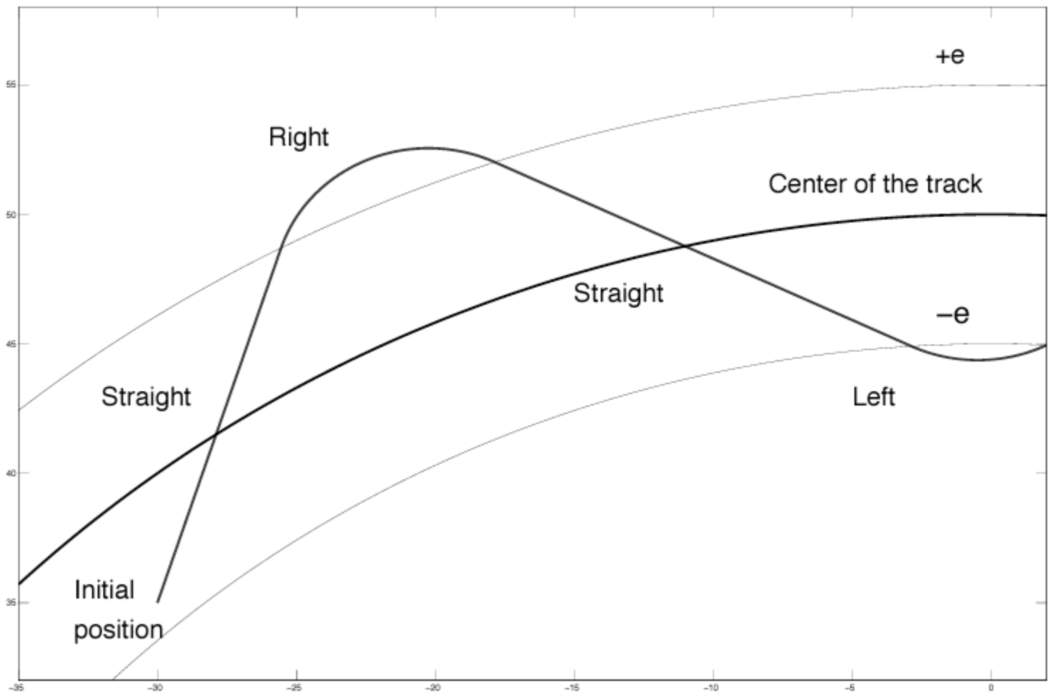


Automated Guided Vehicle



- ❑ Autonomous vehicle on a flat surface, following a visual track
- ❑ Goal of each robot:
  - Move along a track (i.e., center line of a road)
  - Follow track as close as possible
- ❑ Cameras and vision processing algorithms allow vehicle to sense track and measure (signed) distance d from center of the track
- ❑ Two degrees of freedom: move forward and rotate
- ❑ Two velocities: (regular) velocity (v, θ) and angular velocity ω

Automated Guided Vehicle Controller



Inputs: {start, stop} command c, distance d from center of track

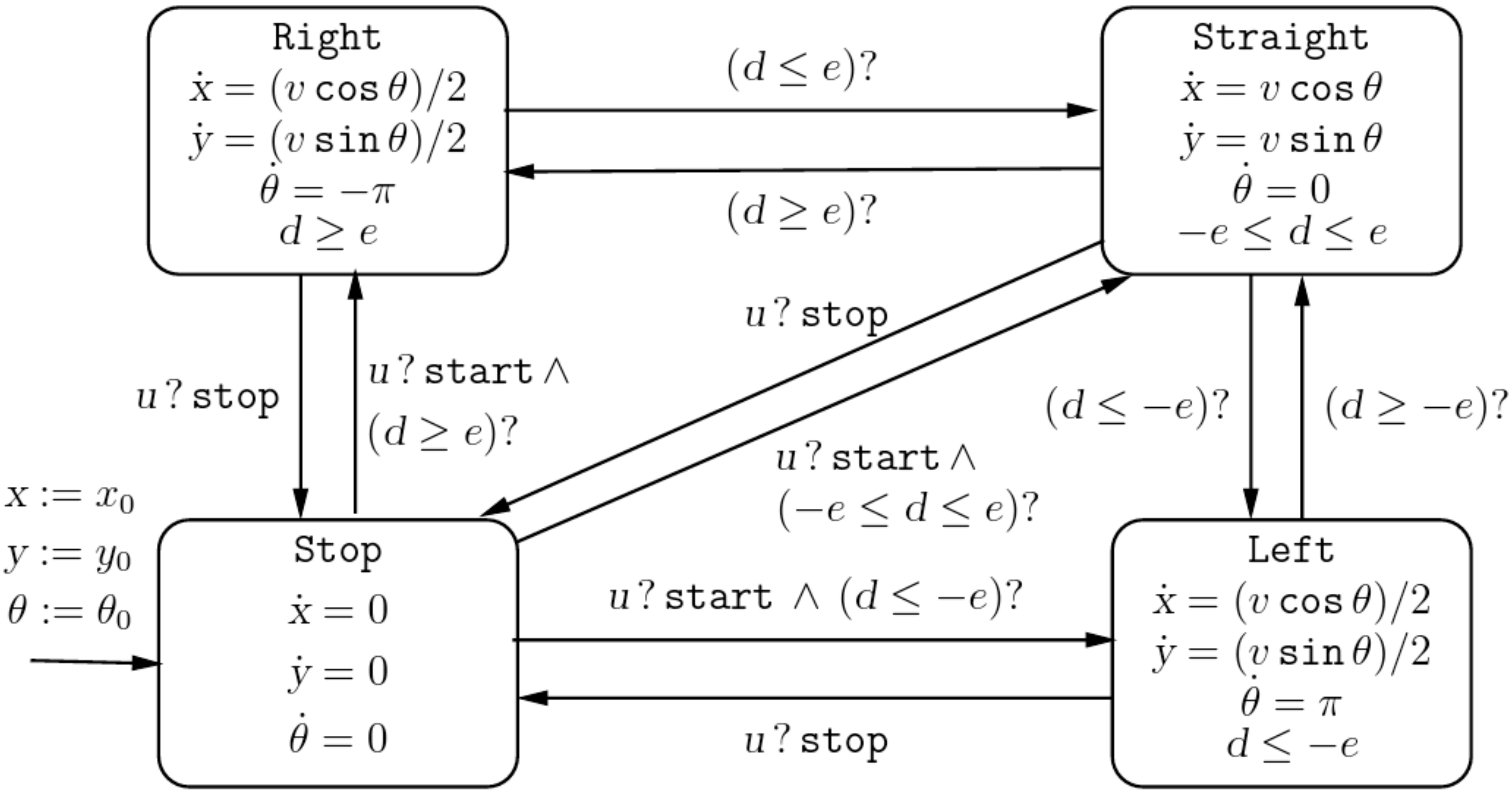
Outputs: speed v, angular speed ω

State: coordinates x, y; angle θ

Modes: Stop, Straight, Left, Right

Simplifications: $v \in \{v_c/2, v_c\}$ and $\omega \in \{-\pi, 0, \pi\}$

Automated Guided Vehicle Controller



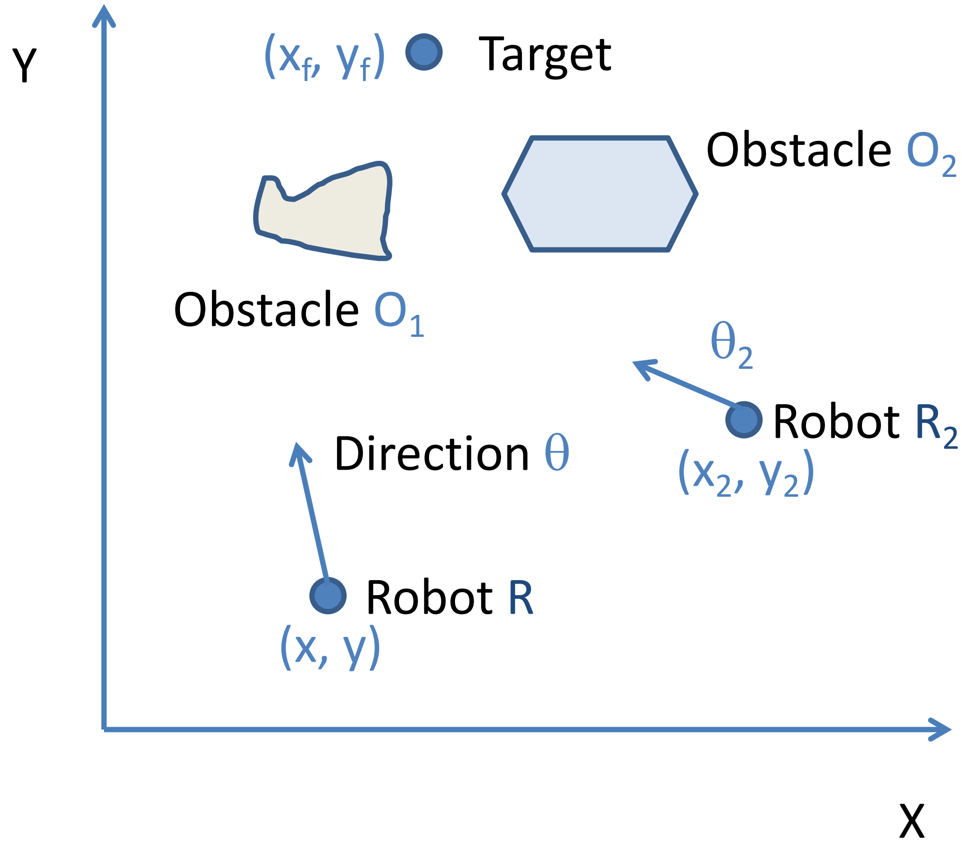
Multi-Robot Coordination

- Autonomous mobile robots in a room
- Goal of each robot:
  - Reach a target at a known location
  - Avoid obstacles (positions of obstacles not known in advance)
  - Minimize distance travelled
- Cameras and vision processing algorithms allow each robot to estimate obstacle positions
  - Estimates are only approximate, and depend on relative position of obstacles with respect to a robot’s position
  - How often should robot update these estimates ?

Multi-Robot Coordination

- Each robot can communicate with others using wireless links
  - How often and what information?
  - How does communication help?
- High-level motion control (path planning)
  - Decide on speed and direction

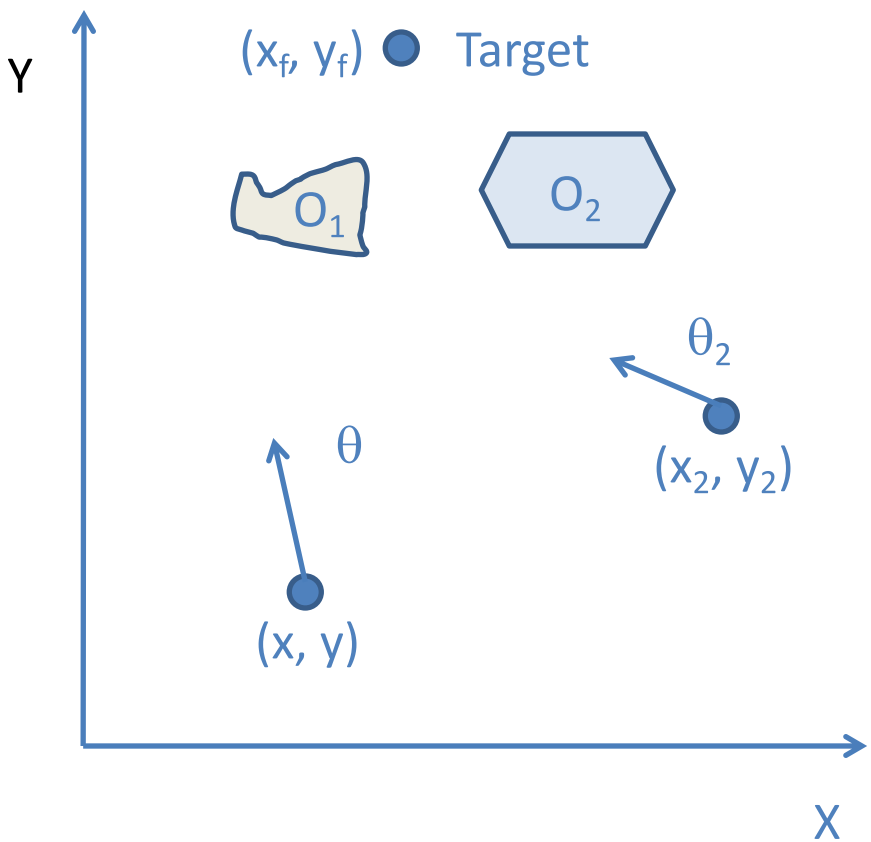
Path Planning with Obstacle Avoidance



Assumptions:

- Two-dimensional world
- Point robots
- Fixed speed v

Path Planning with Obstacle Avoidance



State variables: (x, y), (x_2, y_2)

Initialization:

$(x, y) := (x_0, y_0)$

$(x_2, y_2) := (x_{20}, y_{20})$

Dynamics:

$dx = v \cos \theta$ $\quad dx_2 = v \cos \theta_2$

$dy = v \sin \theta$ $\quad dy_2 = v \sin \theta_2$

Safety requirement:

$(x, y), (x_2, y_2) \notin O_1 \cup O_2$

Liveness requirement:

Eventually $(x, y) = (x_f, y_f)$ and

Eventually $(x_2, y_2) = (x_f, y_f)$

Performance requirement: Reduce distance travelled!

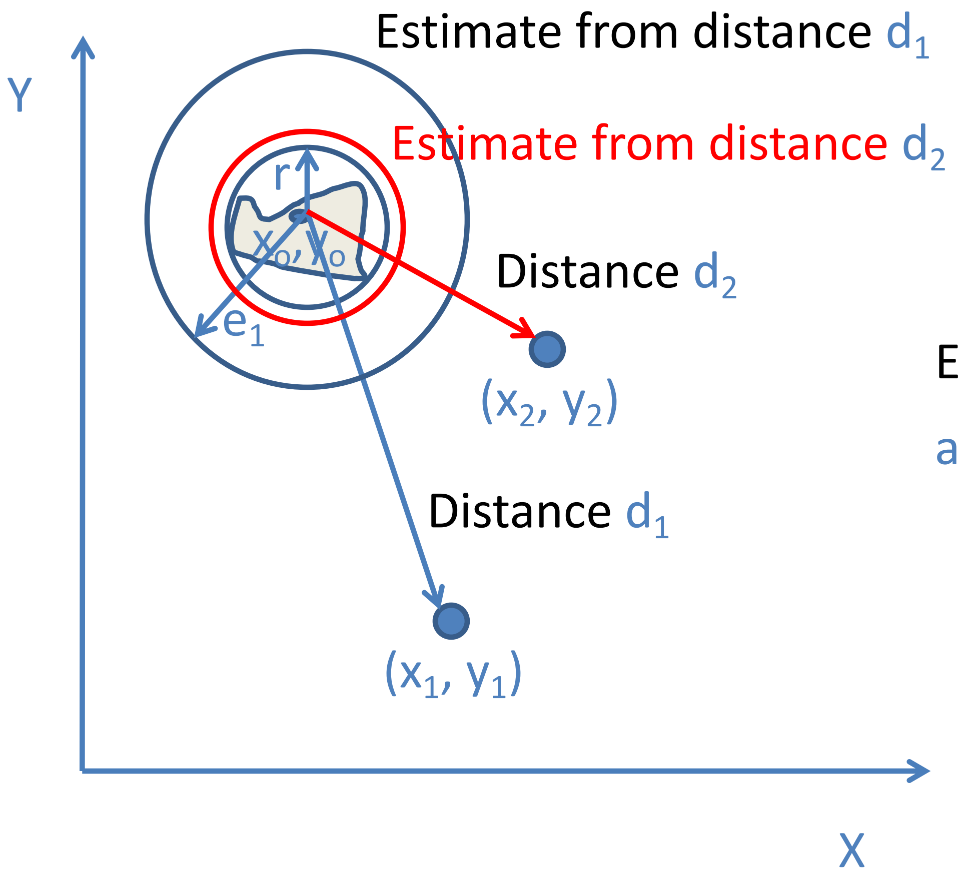
Abstractions

- For modeling and analysis for motion planning, we need to simplify obstacle shapes and complexity of image processing algorithms
  - Simplicity and abstraction: key to modeling
- Assume each robot is a point
  - Can be described by coordinates of point
- Assume each obstacle/estimate is a circle
  - Can be described by coordinates of center and radius
  - Assumption: real obstacle is always contained in estimated circle
  - Alternative: ellipses (more accurate)

Modeling Obstacles

- Consider an obstacle with center (x_o, y_o) and radius r
  - Radius of smallest circle that envelopes the actual obstacle
- Estimate of the obstacle as computed by a robot using image processing algorithms of a robot
  - A circle with center (x_o, y_o) and radius $e > r$
  - The closer the robot to the obstacle, the better the estimate
  - Estimate e decreases with distance of robot from obstacle, and converges to r

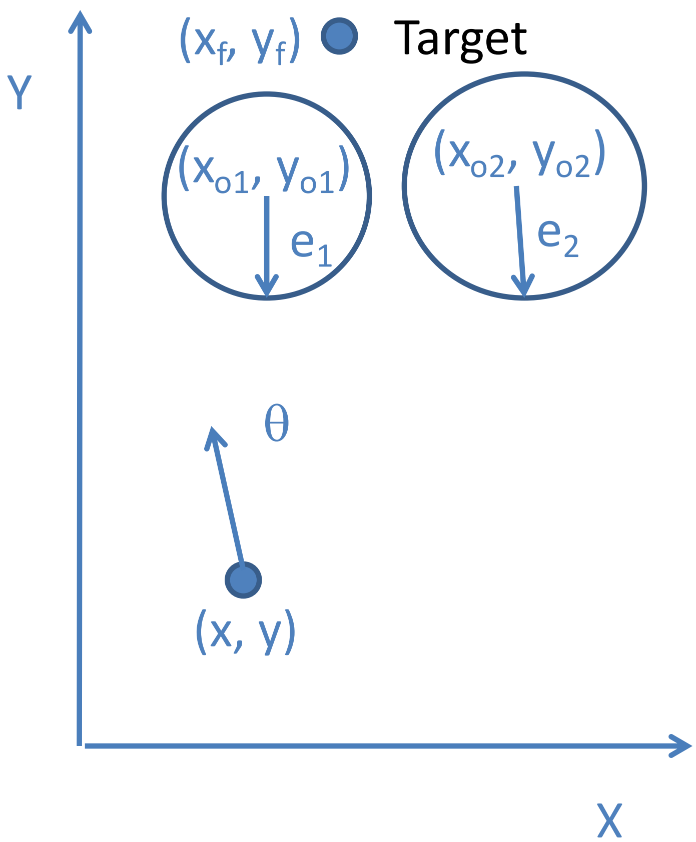
Obstacle Estimation



Estimated radius $e_1 = r + a\,(d_1 - r)$

a is a constant in $(0,1)$

Rule for Obstacle Estimation



Robot R maintains radii e_1 and e_2 that are estimates of the obstacles

Obstacle estimation in reality is done periodically as it is computationally expensive

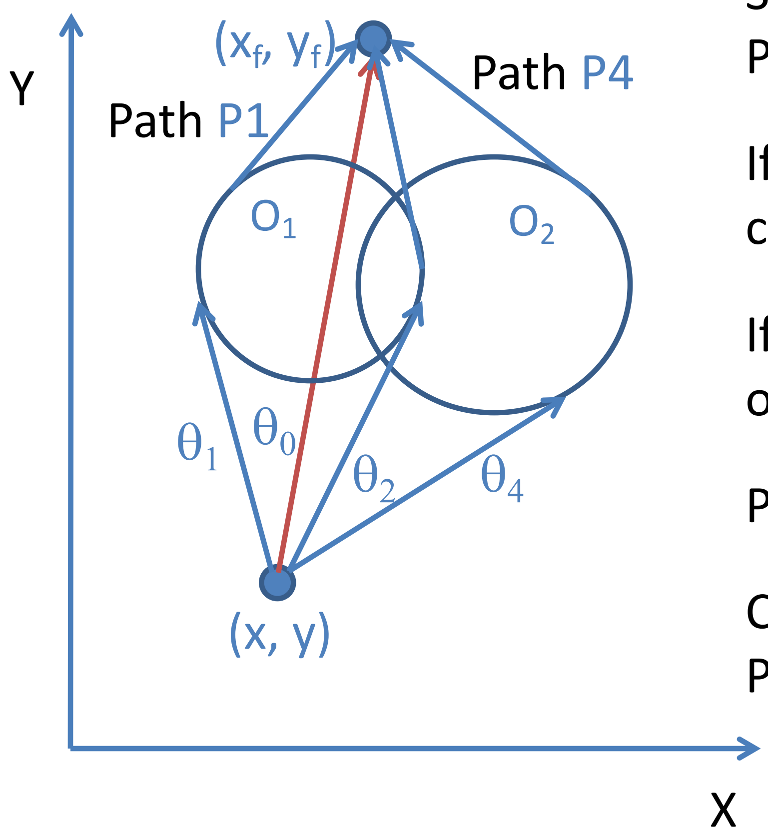
Every t_e seconds, robot model executes discrete updates:

$e_1 := \min (e_1, r_1 + a (dist((x, y), (x_{o1}, y_{o1})) - r_1)$;

$e_2 := \min (e_2, r_2 + a (dist((x, y), (x_{o2}, y_{o2})) - r_2)$

Computation for robot R_2 is similar

Path Planning



Shortest path: straight line to target
Preferred direction: θ_0

If estimate of obstacle O_1 intersects straight path, calculate two paths that are tangents to obstacle

If estimate of obstacle O_2 intersects straight path, or obstacle O_1, calculate tangent paths

Plausible paths: P1 and P4

Calculate which one is shorter:
Planning algorithm returns either θ_1 or θ_4

Path Planning

- Function plan with inputs:
  - current position of robot R_i
  - target position
  - obstacle O_1 position (center and radius estimate)
  - obstacle O_2 position (center and radius estimate)
- Output: Direction for motion
  - Best possible path to target while avoiding obstacles and assuming estimates are correct
- Function plan written in C code (can be embedded in model)
- Does it help to rerun planning algorithm again as robot moves?
  - Yes! Estimates may improve, suggesting shorter paths
  - Invoke planning algorithm every t_p seconds

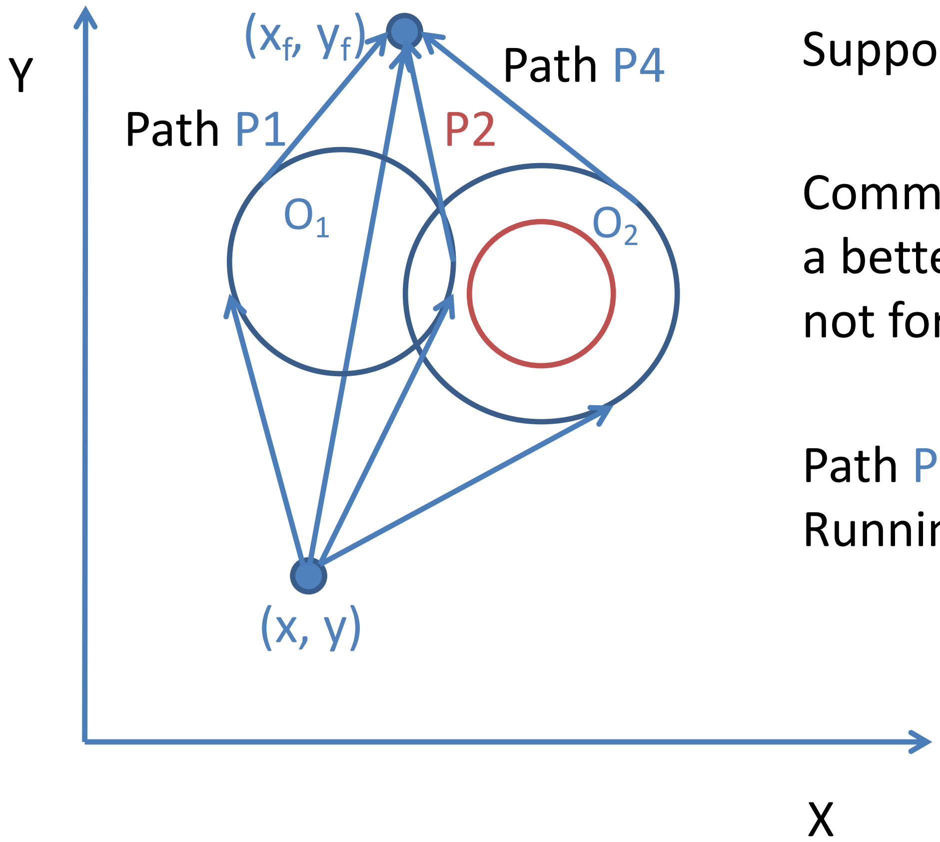
Communication

- Each robot has its own estimate of each obstacle
- Robot R_2's estimates may be better than R_1's own estimates
- Strategy: Every t_c seconds, send your own estimates to the other robot, and receive estimates from it
- If your own estimates are e_{i1} and e_{i2}, and you receive estimates e_{j1} and e_{j2}, set

  $e_{i1} := \min(e_{i1}, e_{j1})$

  $e_{i2} := \min(e_{i2}, e_{j2})$

Effect of Coordination

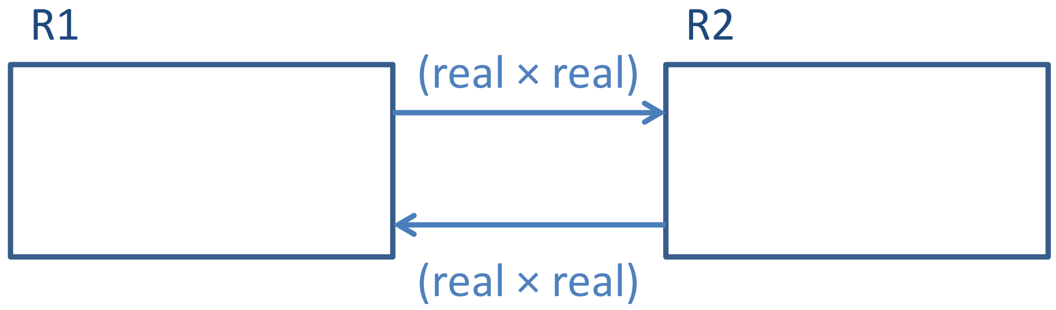


Suppose Path P1 was preferred

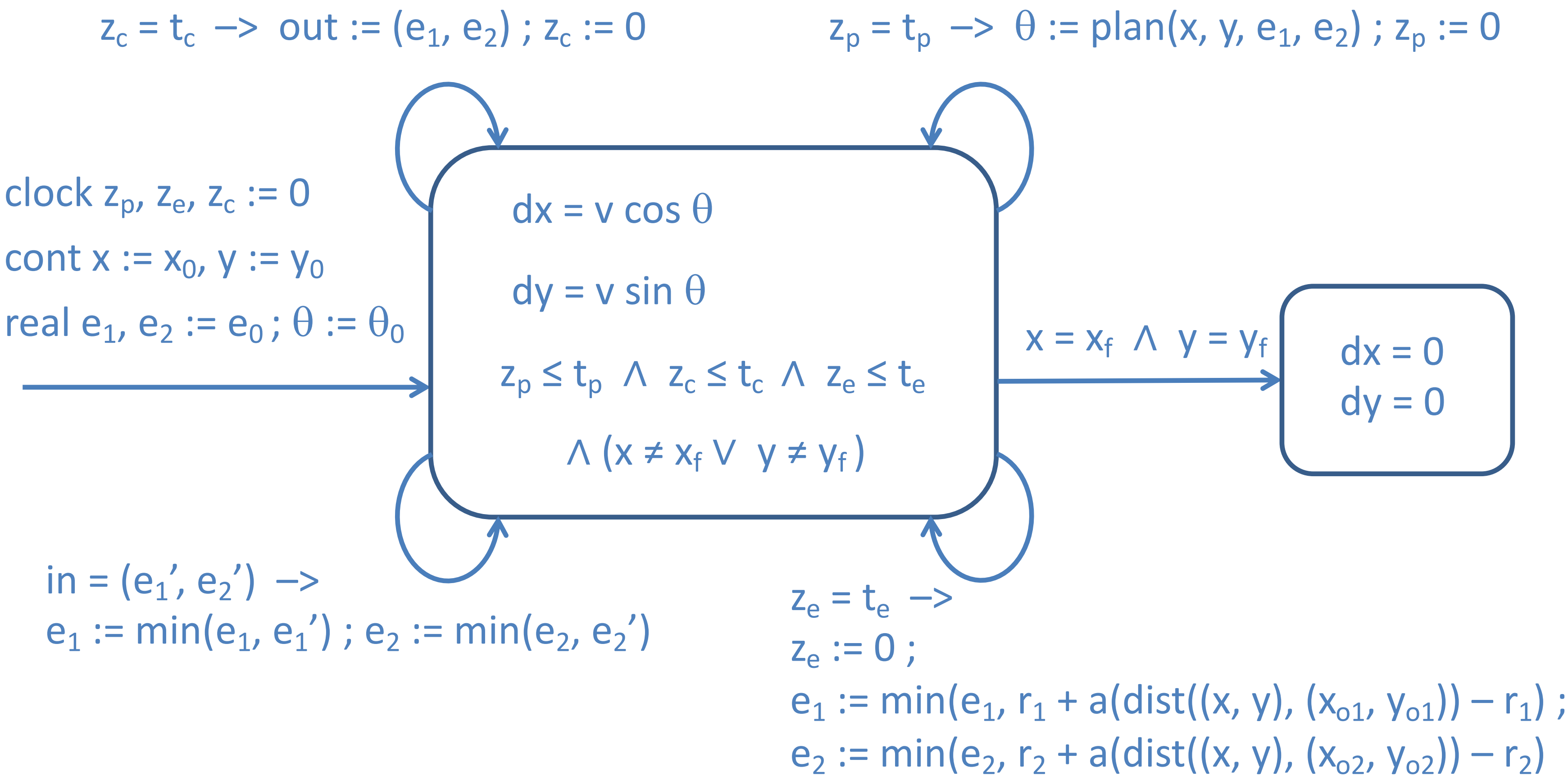
Communication with other robot gives a better estimate of obstacle O_2, but not for obstacle O_1

Path P2 is now viable.
Running planner again could choose path P2

System of Robots



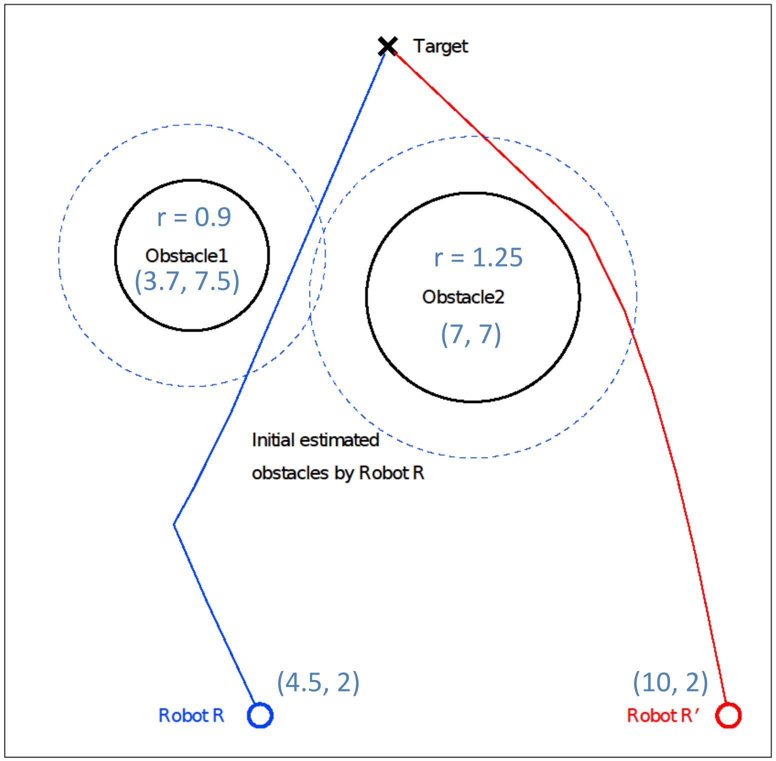
Robot Model

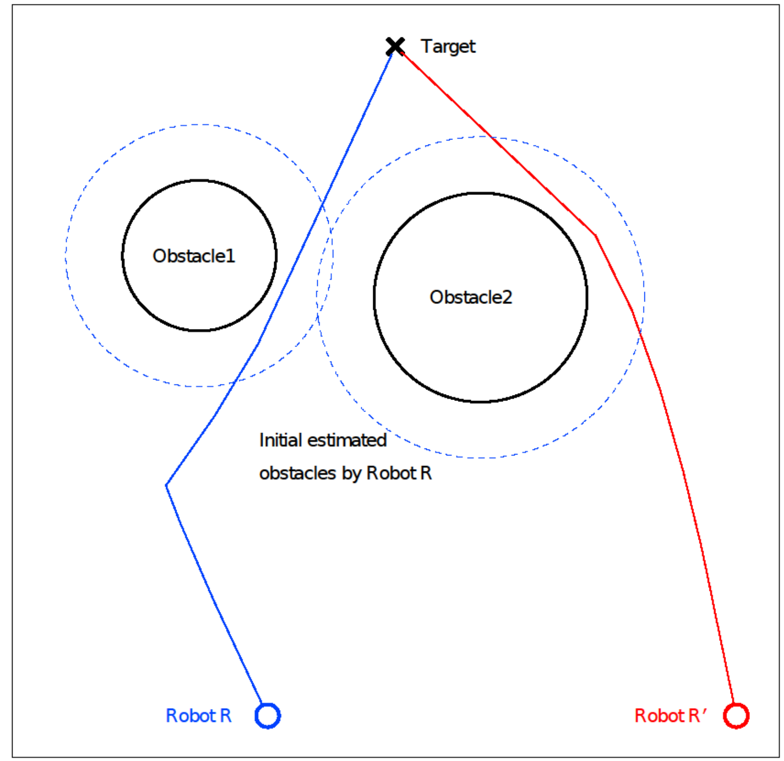

$z_c = t_c$ –> out := (e_1, e_2) ; $z_c := 0$
$z_p = t_p$ –> θ := plan(x, y, e_1, e_2) ; $z_p := 0$
clock $z_p, z_e, z_c := 0$
cont $x := x_0, y := y_0$
real $e_1, e_2 := e_0$; $\theta := \theta_0$
$dx = v \cos \theta$
$dy = v \sin \theta$
$z_p \leq t_p \wedge z_c \leq t_c \wedge z_e \leq t_e$
$\wedge (x \neq x_f \vee y \neq y_f)$
$x = x_f \wedge y = y_f$
$dx = 0$
$dy = 0$
in = (e_1', e_2') –>
$e_1 := \min(e_1, e_1')$; $e_2 := \min(e_2, e_2')$
$z_e = t_e$ –>
$z_e := 0$;
$e_1 := \min(e_1, r_1 + a(\mathrm{dist}((x, y), (x_{o1}, y_{o1})) - r_1)$;
$e_2 := \min(e_2, r_2 + a(\mathrm{dist}((x, y), (x_{o2}, y_{o2})) - r_2)$

Analysis

- Key system parameters
  - How often should a robot communicate?
  - How often should a robot execute planning algorithm
  - How often should a robot execute image processing algorithm to update obstacle estimates?
- Design-space exploration: Choose values of t_c, t_p, t_e
  - Reduce distance travelled, but also account for costs of communication/computation
- Symbolic analysis beyond the scope of current tools, so need to run multiple simulations

Illustrative Execution





- Speed v : 0.5 u/s
- Planning rate t_p : 2 s
- Obstacle estimation rate t_p : 2 s
- Communication rate t_c : **4 s**
- Distance travelled by R' : 9.15 u
- Distance travelled by R : **8.65** u

- Speed v : 0.5 u/s
- Planning rate t_p : 2 s
- Obstacle estimation rate t_p : 2 s
- Communication rate t_c : **>> 4 s**
- Distance travelled by R' : 9.15 u
- Distance travelled by R : **8.81** u

Credits

Notes based on Chapter 9 of

Principles of Cyber-Physical Systems
by Rajeev Alur
MIT Press, 2015