

# Robotics: Principles and Practice

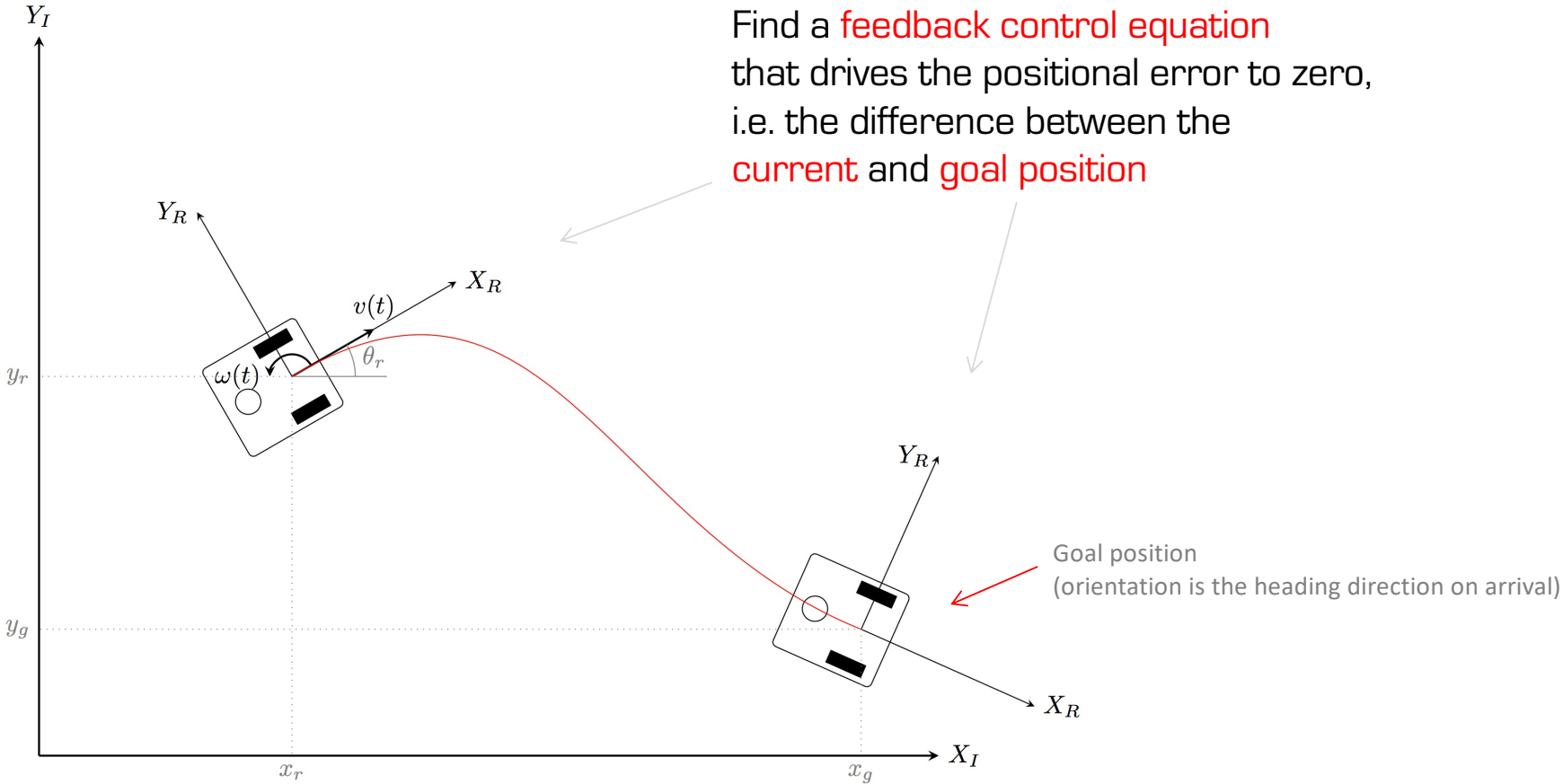
## Module 3: Mobile Robots

### Lecture 6: The go-to-position problem; divide-and-conquer controller

David Vernon  
Carnegie Mellon University Africa

[www.vernon.eu](http://www.vernon.eu)

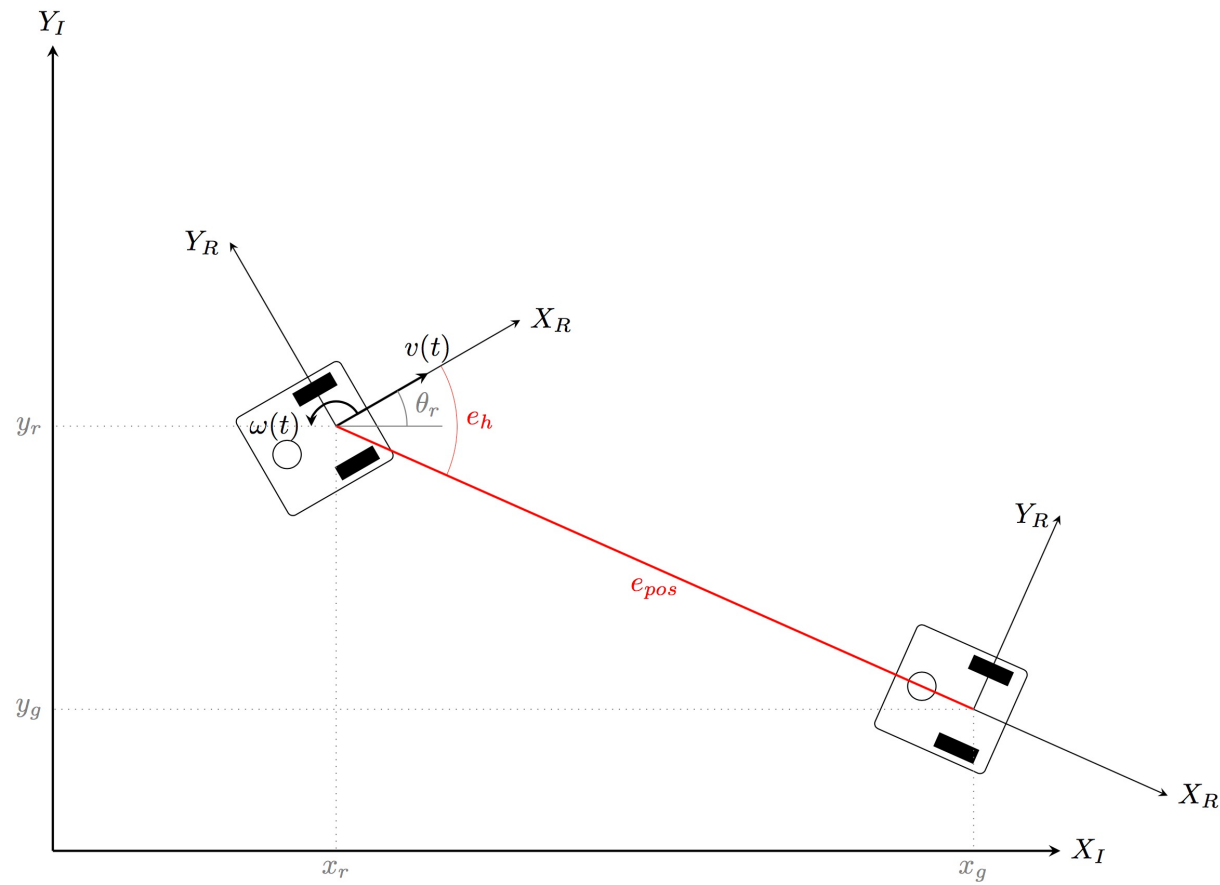
# The Go-to-Position Problem



# The Go-to-Position Problem

- The robot knows its own global current position
  - $(x_r, y_r, \theta_r)$
- It knows the global position of the goal
  - $(x_g, y_g)$
- Compute error in position and heading
  - $(e_{pos}, e_h)$

# The Go-to-Position Problem



# The Go-to-Position Problem

- The robot knows its own global current position
  - $(x_c, y_c, \theta_c)$
- It knows the global position of the goal
  - $(x_g, y_g)$
- Compute error in position and heading
  - $(e_{pos}, e_h)$
- Reduce both errors to zero
  - by generating the appropriate forward and angular velocities  $(v, \omega)$  or, alternatively,
  - by generating the appropriate angular velocities of the wheels,  $(v_R, v_L)$  i.e.  $(\dot{\phi}_1, \dot{\phi}_2)$

# Go-to-Position as a Control Problem

## Solution 1: Divide and Conquer

Decompose 2D problem into two 1D problems

- First, correct the heading:

rotate to reduce the orientation error to zero

- Second, correct the position:

translate straight ahead to reduce the position error to zero

# Go-to-Position as a Control Problem

## Solution 1: Divide and Conquer

Algorithm `goto1(xg, yg)` using proportional control

Global variables: the current robot position and orientation  $(x_r, y_r, \theta_r)$

Arguments:

- the goal position of the robot  $(x_g, y_g)$
- the proportional gains for controlling position and heading

$$K_p^{pos}$$

$$K_p^h$$

- the tolerances on position error  $\Delta_{pos}$  and orientation error  $\Delta_h$

# Go-to-Position as a Control Problem

## Solution 1: Divide and Conquer

Version A: Control forward and angular velocities of the robot

Do

Compute the current position of the robot  $(x_r, y_r)$

Compute the distances from the robot position to the target position  $(d_x, d_y)$

$$d_x = x_g - x_r$$

$$d_y = y_g - y_r$$

Compute the position and heading errors  $(e_{pos}, e_h)$

$$e_{pos} = \text{sqrt}(d_x^2 + d_y^2)$$

$$e_h = \text{atan2}(d_y, d_x) - \theta_r$$

The difference between the desired heading and the current robot orientation

If not oriented correctly, correct heading

Otherwise, correct position

if  $|e_h| > \Delta_h$

set forward velocity:  $v = 0$

set angular velocity:  $\omega = K_p^h e_h$

else

set forward velocity:  $v = K_p^{pos} e_{pos}$

set angular velocity:  $\omega = 0$

or some maximum velocity  $v_{max}$

Send velocities  $(v, \omega)$  to the robot

Pause some time

while  $|e_{pos}| > \Delta_{pos}$

Send velocities  $(0, 0)$  to the robot



# Go-to-Position as a Control Problem

## Solution 1: Divide and Conquer

Version B: Control angular velocities of the wheels

Do

Compute the current position of the robot  $(x_r, y_r)$

Compute the distances from the robot position to the target position  $(d_x, d_y)$

$$d_x = x_g - x_r$$

$$d_y = y_g - y_r$$

Compute the position and heading errors  $(e_{pos}, e_\theta)$

$$e_{pos} = \text{sqrt}(d_x^2 + d_y^2)$$

$$e_h = \text{atan2}(d_y, d_x) - \theta_r$$

If not oriented correctly,  
correct orientation

Otherwise,  
correct position

if  $|e_h| > \Delta_h$

set right wheel angular velocity:  $\dot{\phi}_1 = -K_p^h e_h$

set left wheel angular velocity:  $\dot{\phi}_2 = -\dot{\phi}_1$

Rotate: left and right wheels go in opposite direction

else

set right wheel angular velocity:  $\dot{\phi}_1 = K_p^{pos} e_{pos}$

set left wheel angular velocity:  $\dot{\phi}_2 = \dot{\phi}_1$

or some maximum velocity

Translate: left and right wheels go in same direction

Send velocities  $(\dot{\phi}_1, \dot{\phi}_2)$  to the robot

Pause some time

while  $|e_{pos}| > \Delta_{pos}$

Send velocities  $(0, 0)$  to the robot