ETH Zürich

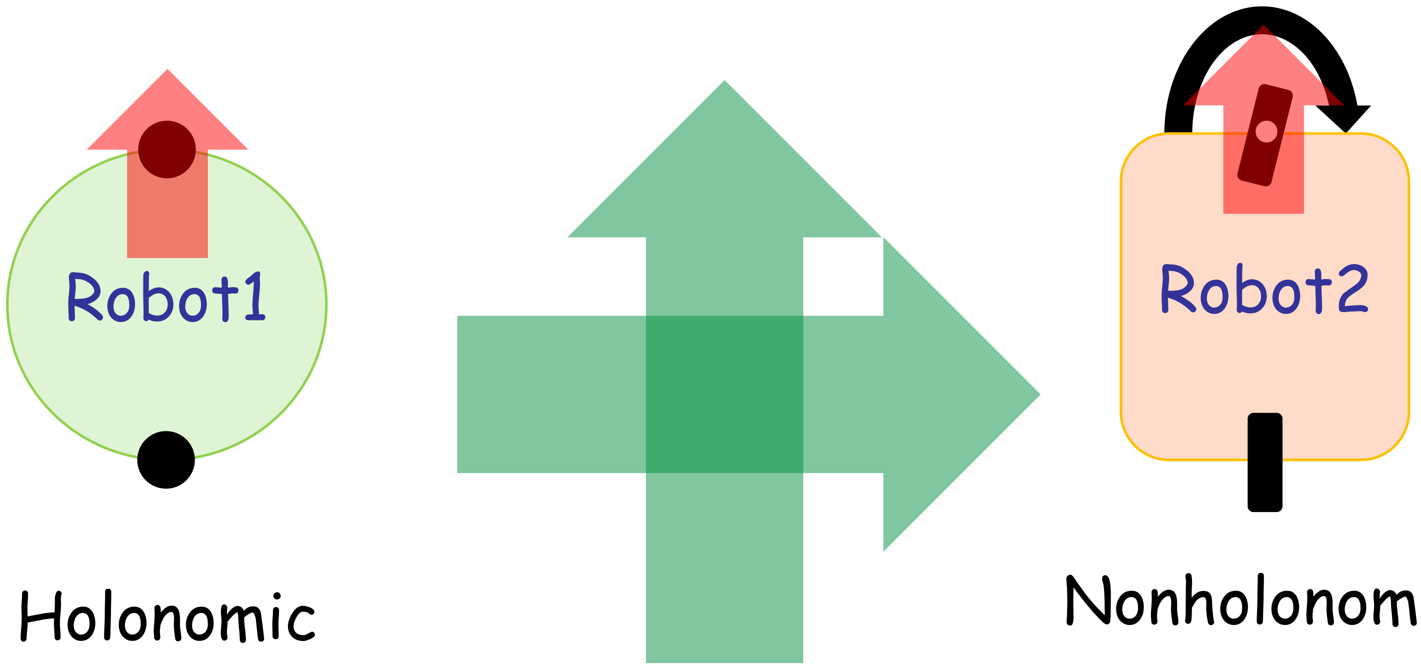
Chair of Software Engineering

Robotics Programming Laboratory

Bertrand Meyer
Jiwon Shin

Lecture 4: Robot Control and Obstacle Avoidance

Go forward, go right



Holonomic

DDOF=DOF

Nonholonomic

DDOF<DOF

DOF: Ability to achieve various poses

DDOF: Ability to achieve various velocities

Differential drive

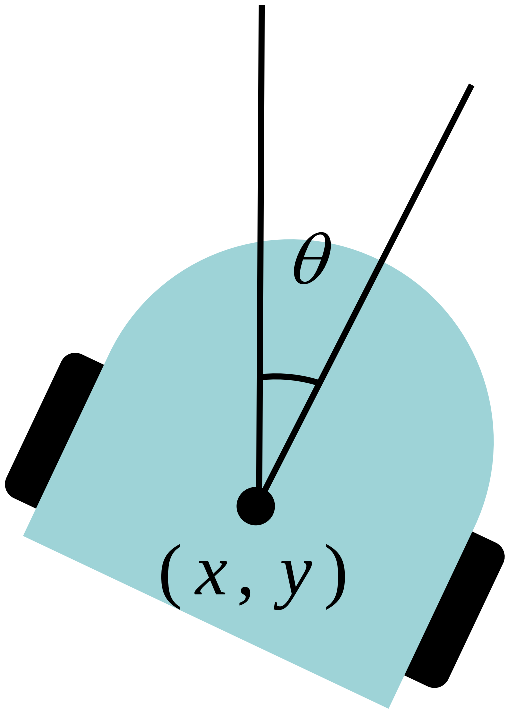
Forward: $\dot{\varphi}_L = \dot{\varphi}_R > 0$

Backward: $\dot{\varphi}_L = \dot{\varphi}_R < 0$

Right turn: $\dot{\varphi}_L > \dot{\varphi}_R$

Left turn: $\dot{\varphi}_L < \dot{\varphi}_R$

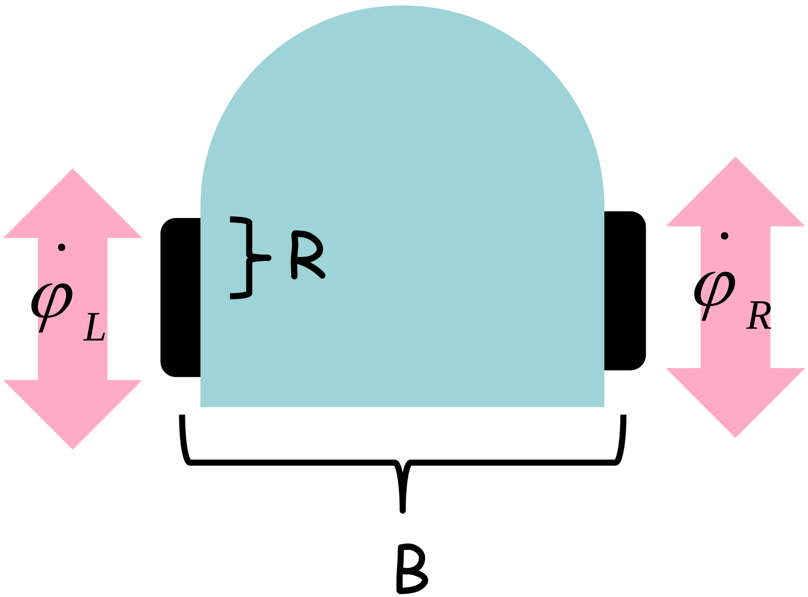
Differential drive



Input: (v, ω)

$$\dot{x} = v\cos\theta$$
$$\dot{y} = v\sin\theta$$
$$\dot{\theta} = \omega$$

Differential drive

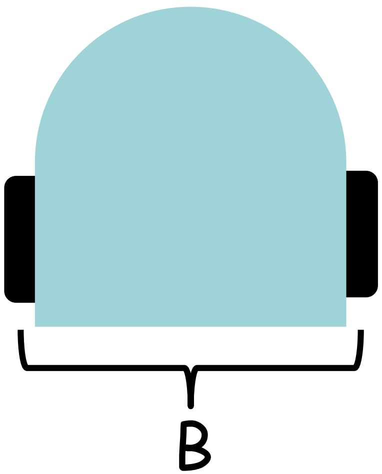


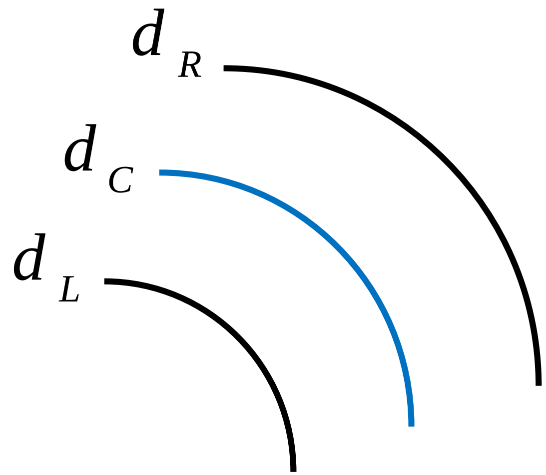
$$\dot{x} = R\,\frac{(\dot{\varphi}_L + \dot{\varphi}_R)}{2}\cos\theta$$

$$\dot{y} = R\,\frac{(\dot{\varphi}_L + \dot{\varphi}_R)}{2}\sin\theta$$

$$\dot{\theta} = \frac{R}{B}(\dot{\varphi}_R - \dot{\varphi}_L)$$

Odometry for small t





$$d_C = \tfrac{1}{2}(d_L + d_R)$$

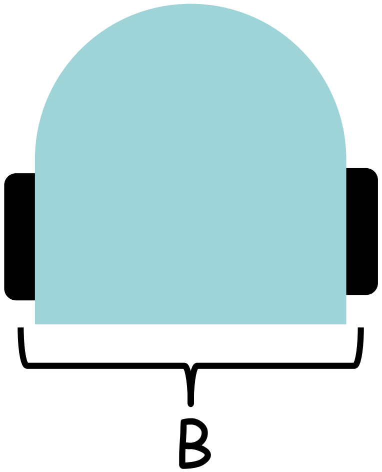
$$\theta_C = \frac{d_R - d_L}{B}$$

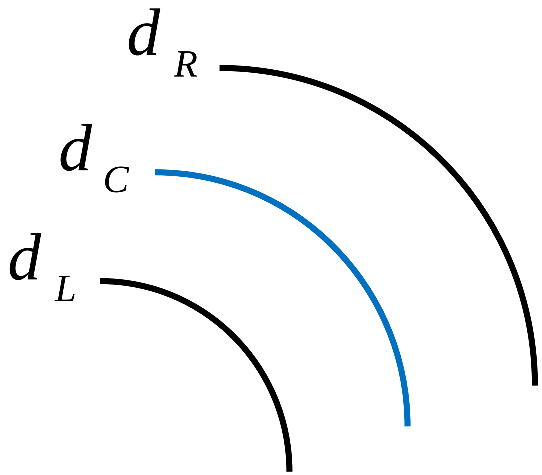
$$x(t) = x(t-1) + d_C \cos\theta(t)$$

$$y(t) = y(t-1) + d_C \sin\theta(t)$$

$$\theta(t) = \theta(t-1) + \theta_C$$

More accurate odometry for small t





$$d_C = \tfrac{1}{2}(d_L + d_R)$$

$$\theta_C = \arctan\left(\frac{d_R - d_L}{B}\right)$$

$$x(t) = x(t-1) + d_C \cos\left(\theta(t-1) + \tfrac{1}{2}\theta_C\right)$$

$$y(t) = y(t-1) + d_C \sin\left(\theta(t-1) + \tfrac{1}{2}\theta_C\right)$$

$$\theta(t) = \theta(t-1) + \theta_C$$

Wheel encoder

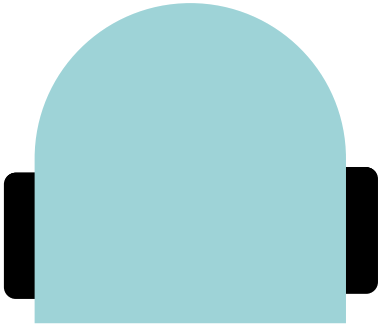
How do we get the distance each wheel has moved?

- If the wheel has N ticks per revolution:

$$\Delta n_{tick} = n_{tick}(t) - n_{tick}(t-1)$$

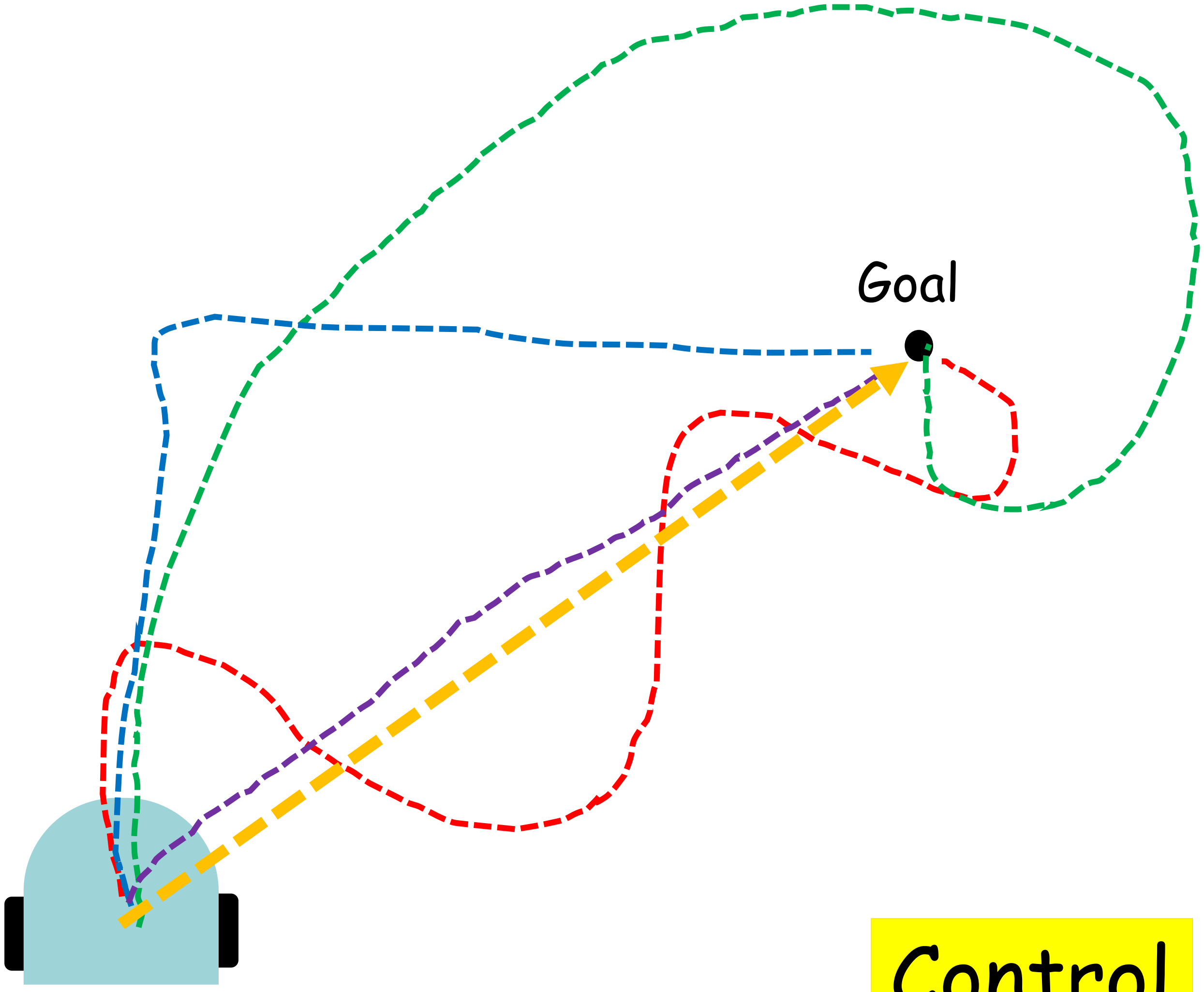
$$d = 2\pi R \frac{\Delta n_{tick}}{N}$$

- Thymio: $d = \dot{d}\,\Delta t$



DRIFT

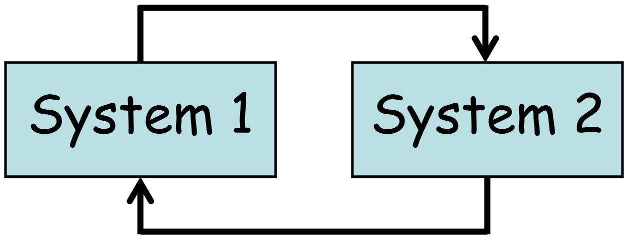
Go to goal



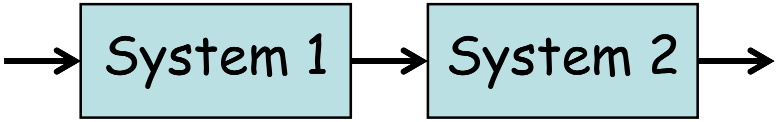
Control

Feedback

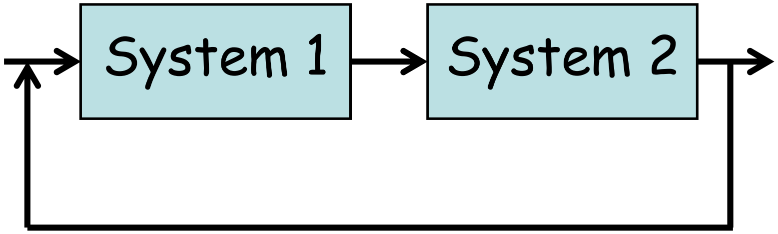
A collection of two or more dynamical systems, in which each system influences the other, resulting in strongly-coupled dynamics



- Open loop: the systems are not interconnected (no feedback)

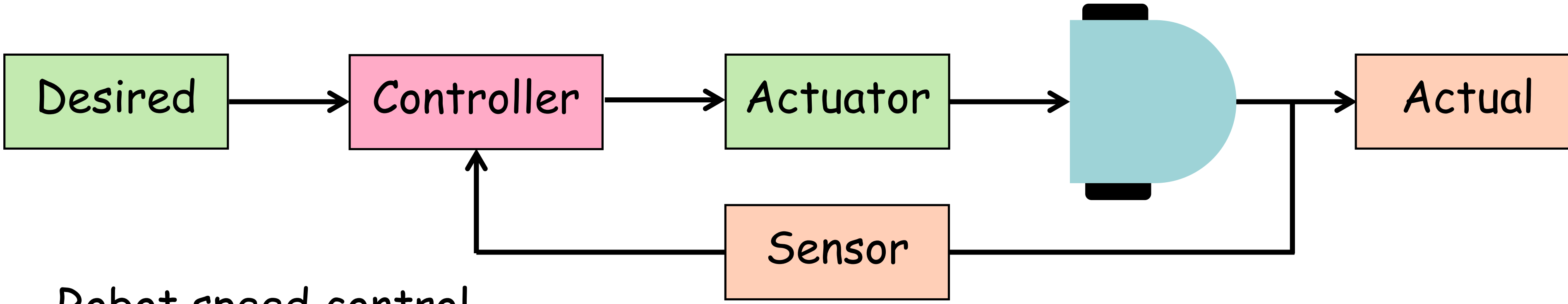


- Closed loop: the systems are interconnected (with feedback)



Control

The use of algorithms and feedback in engineered systems

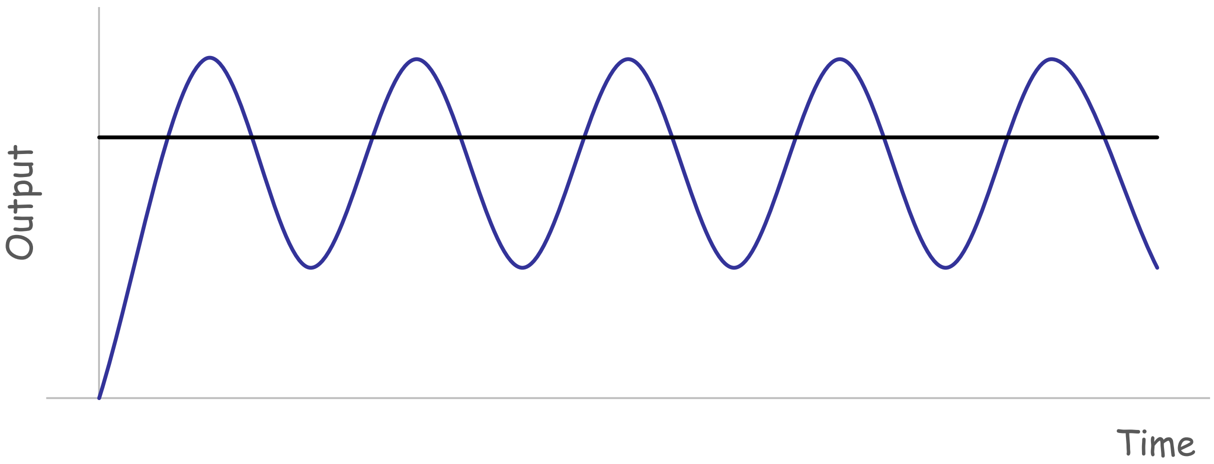


Robot speed control

- Actuator: set the robot's speed
- Sensor: sense the robot's actual speed
- Control goals: set the robot's speed such that:
  - Stability: the robot maintains the desired speed
  - Performance: the robot responds quickly to changes
  - Robustness: the robot tolerates perturbation in dynamics

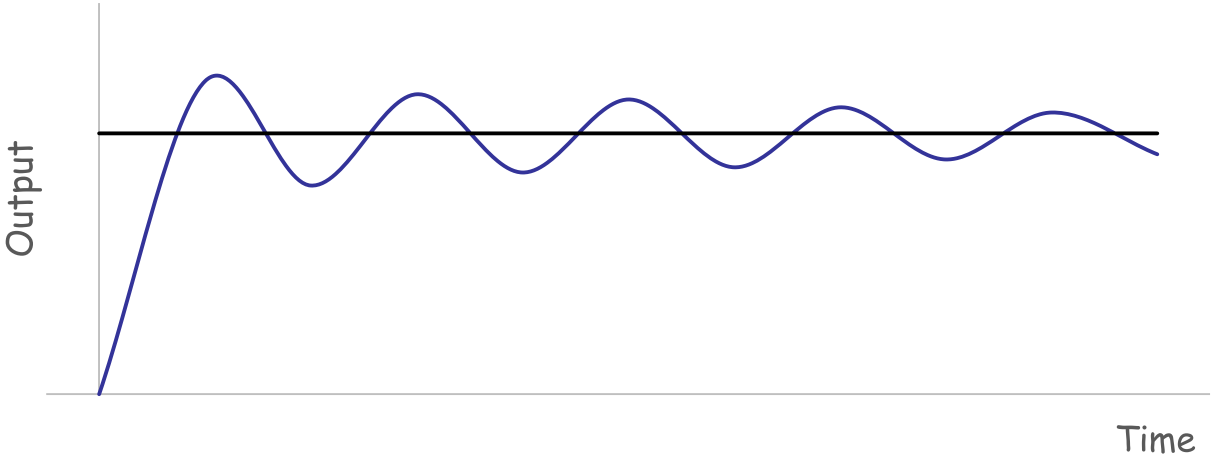
On-off controller

$$u = \begin{cases} u_{max} & \text{if } e > 0 \\ u_{min} & \text{if } e < 0 \end{cases}$$



Proportional controller

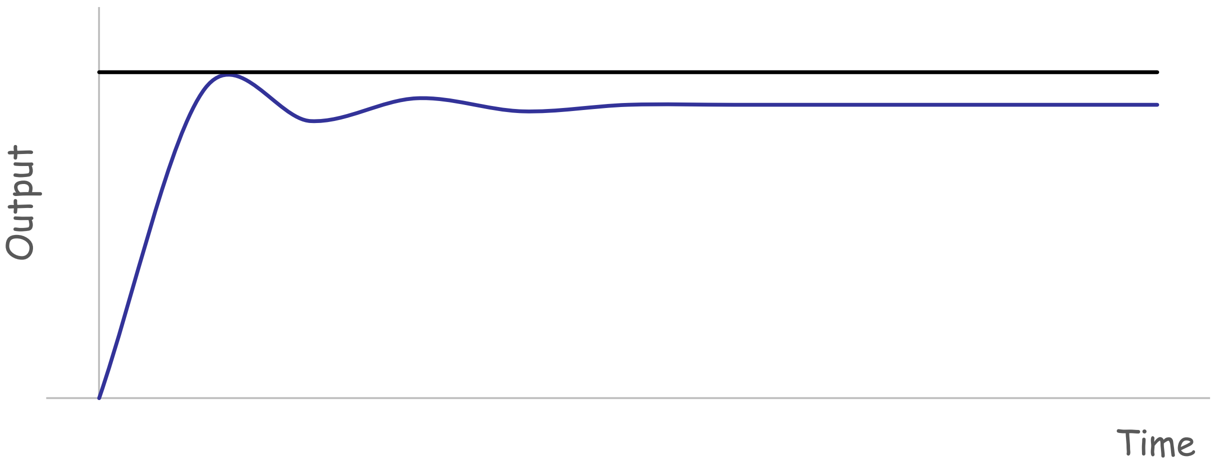
$$u(t) = k_p e(t)$$



More about control gains

Proportional derivative controller

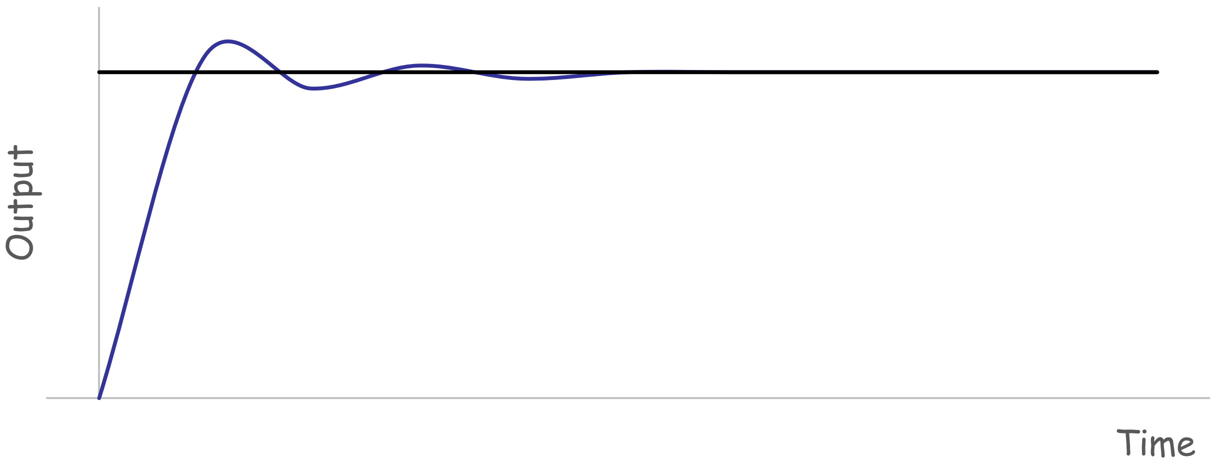
$$u(t) = k_p e(t) + k_d \frac{de(t)}{dt}$$



More about control gains

Proportional integral derivative controller

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)\, d\tau + k_d \frac{de(t)}{dt}$$



More about control gains

Control gains

Ziegler-Nicols method

- Set K_i and K_d to 0.
- Increase K_p until K_u at which point the output starts to oscillate.
- Use K_u and the oscillation period T_u to set the control gains.

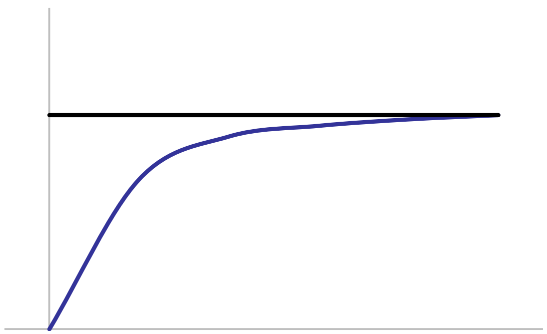
Control Type	**K_p**	**K_i**	**K_d**
P	$0.50K_u$	-	-
PI	$0.45K_u$	$1.2K_p/T_u$	-
PID	$0.60K_u$	$2K_p/T_u$	$K_pT_u/8$

Manual tuning!

P, PI, PID, ….?

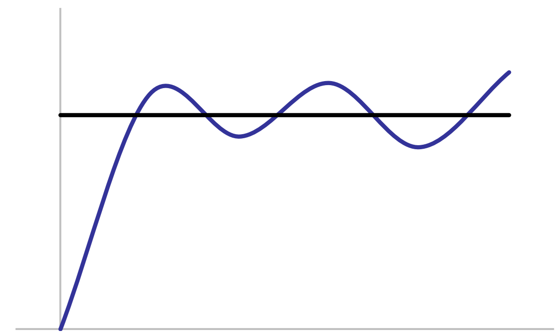
$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)\, d\tau + k_d \frac{de(t)}{dt}$$

a.



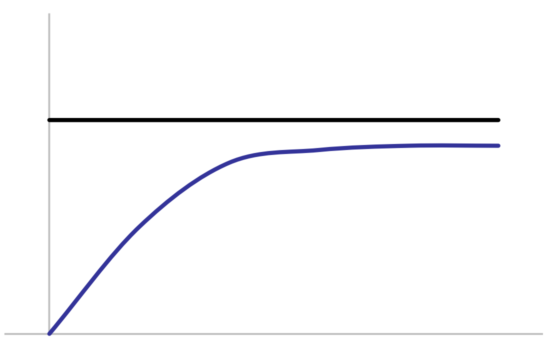
$k_p, k_i, k_d \neq 0$

c.



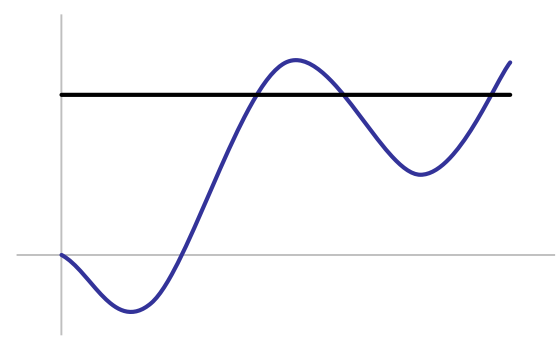
$k_d = 0$

b.



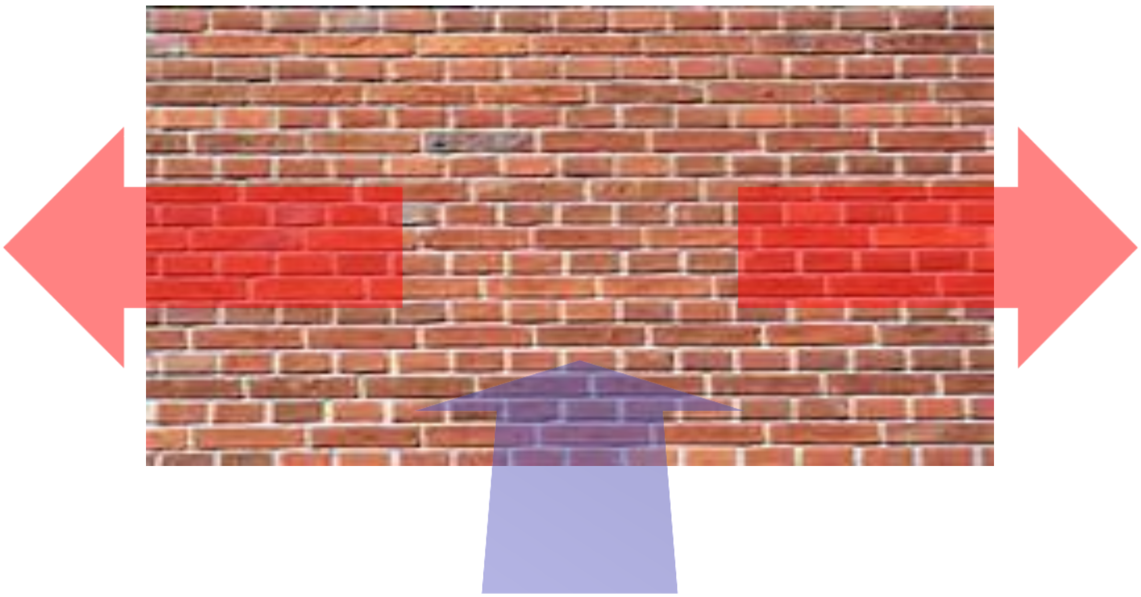
$k_i = 0$

d.

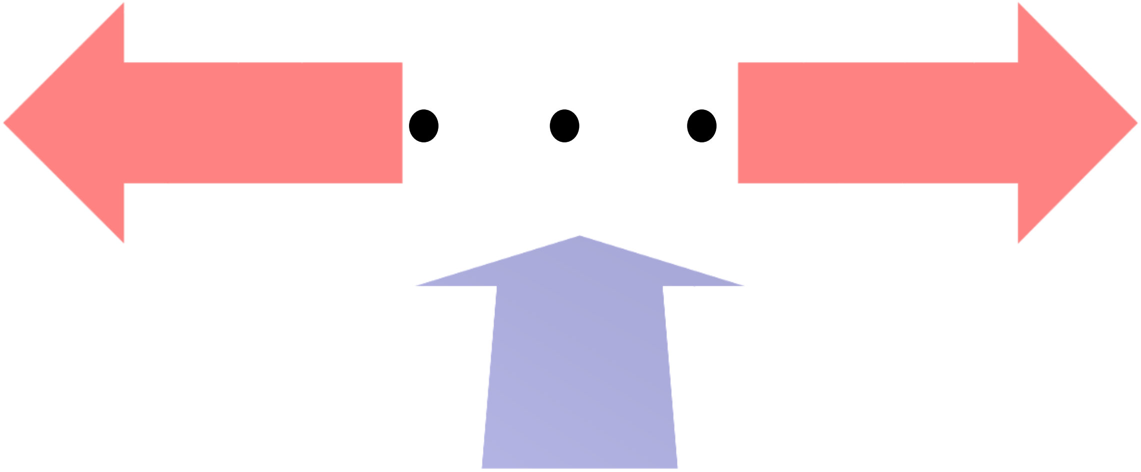


$k_p = 0$

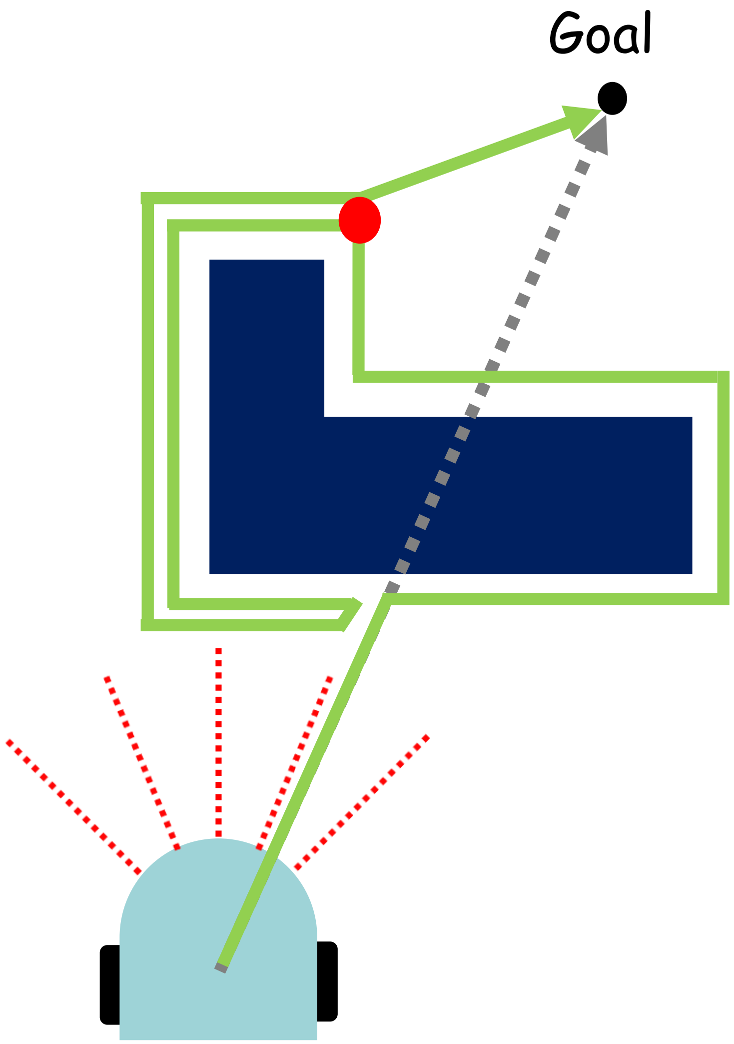
Obstacle avoidance



Obstacle avoidance



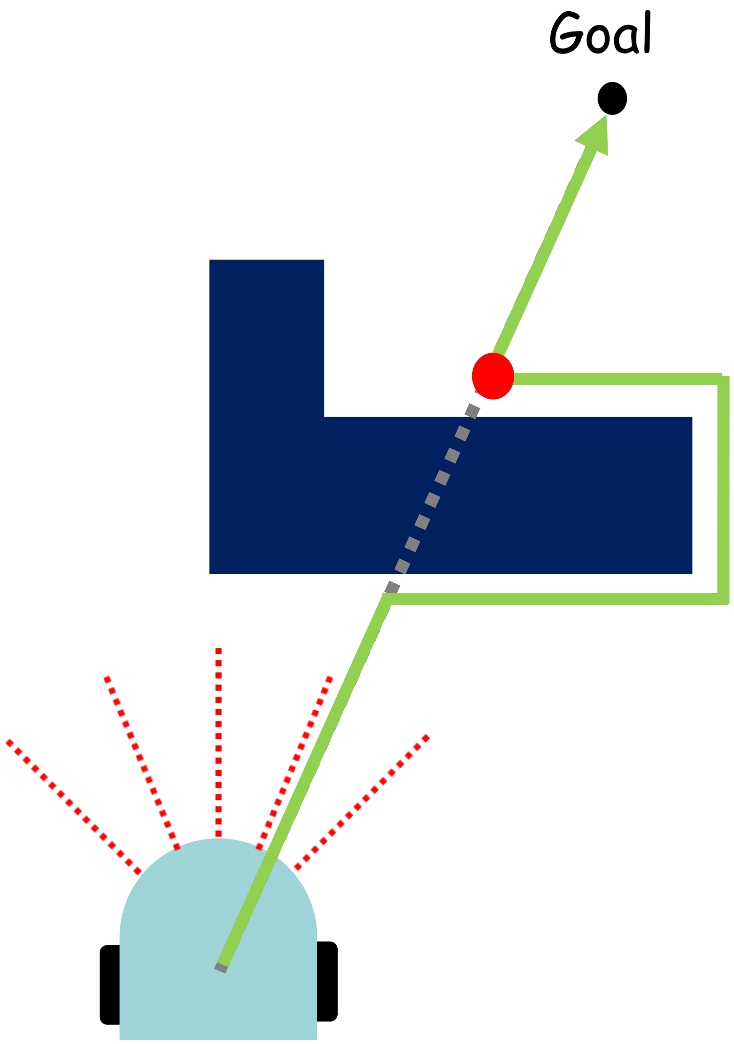
Bug 1



1. Move toward the goal:
   1. If the goal is reached: Stop
   2. If an obstacle is in the way: Go to step 2
2. Follow the obstacle boundary:
   1. Mark the closest
   2. After a complete loop: Go to the closest point to the goal then go back to step 1.

Lumelsky , V. & Stepanov, A. "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," . Algorithmica 2:403-430. 1987

Bug 2



1. Move toward the goal:
   1. If the goal is reached: Stop
   2. If an obstacle is in the way:
      Go to step 2
2. Follow the obstacle boundary:
   1. If the goal line is crossed:
      Go to step 1.

Lumelsky , V. & Skewis, T. "Incorporating range sensing in the robot navigation function," IEEE Transactions on Systems, Man, and Cybernatics 20(5): 1058-1068, 1990.

Is Bug 2 always better than Bug 1?

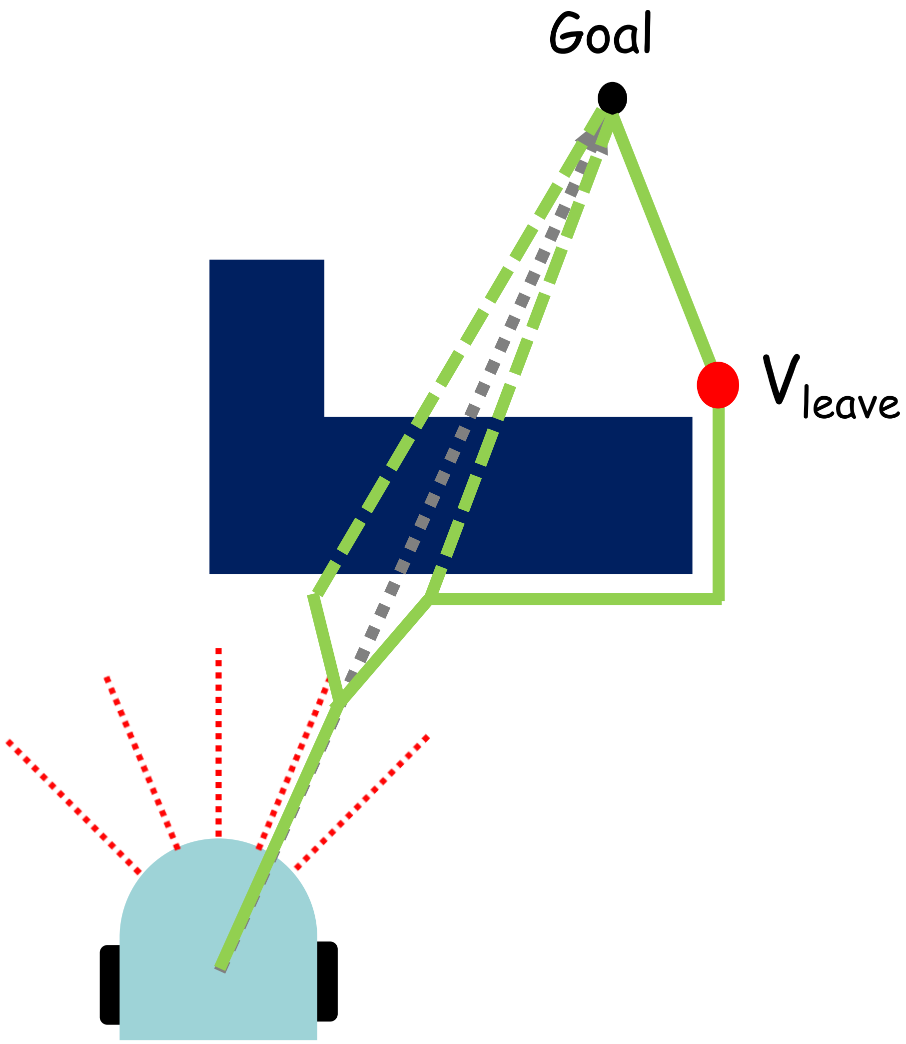
Bug 1

- Exhaustive search: analyze all choices before committing

Bug 2

- Greedy search: take the first viable choice

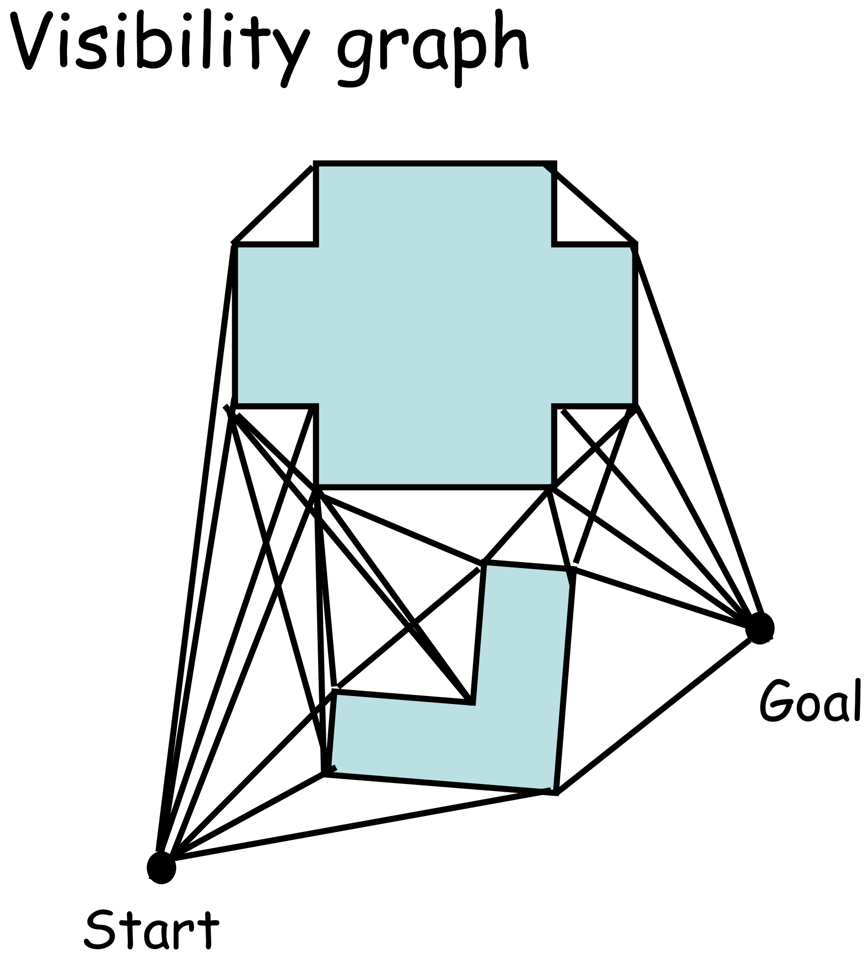
TangentBug

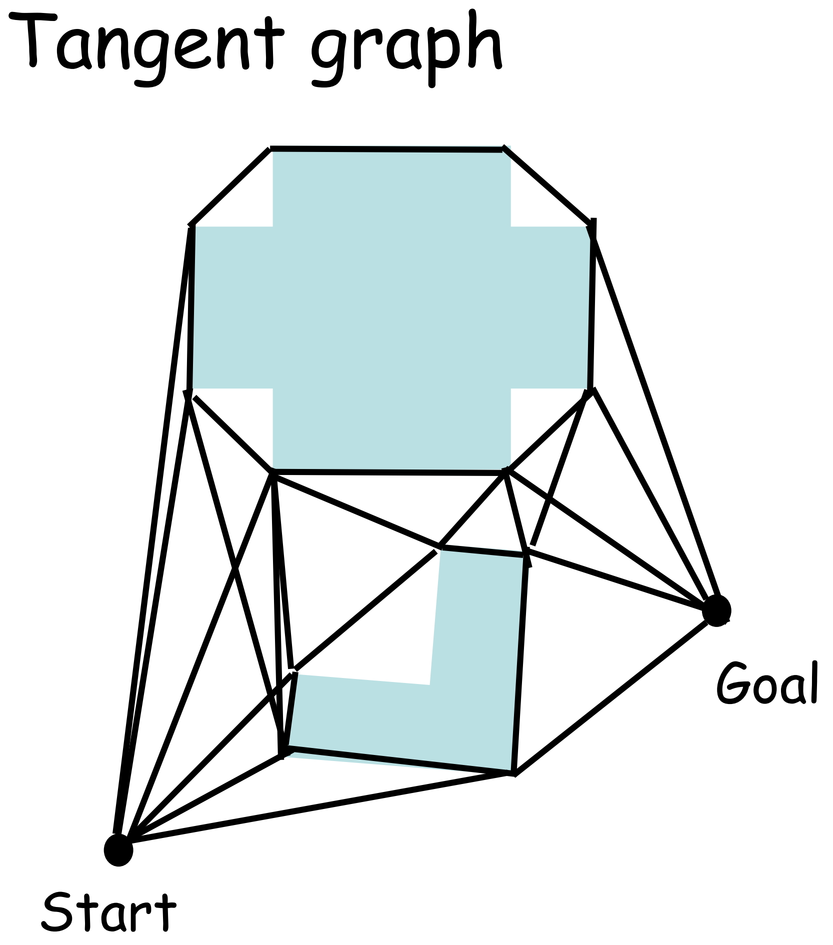


1. Move toward the goal:
   1. If the goal is reached: Stop
   2. If a local minimum is detected: Go to step 2
2. Move along the boundary marking d_{min}:
   1. If the goal is reached: Stop
   2. If $d(V_{leave}, goal) < d_{min}$: Go to step 3
3. Perform the transition phase:
   1. Move directly towards V_{leave} until Z, where $d(Z, goal) < d_{min}$: Go to step 1

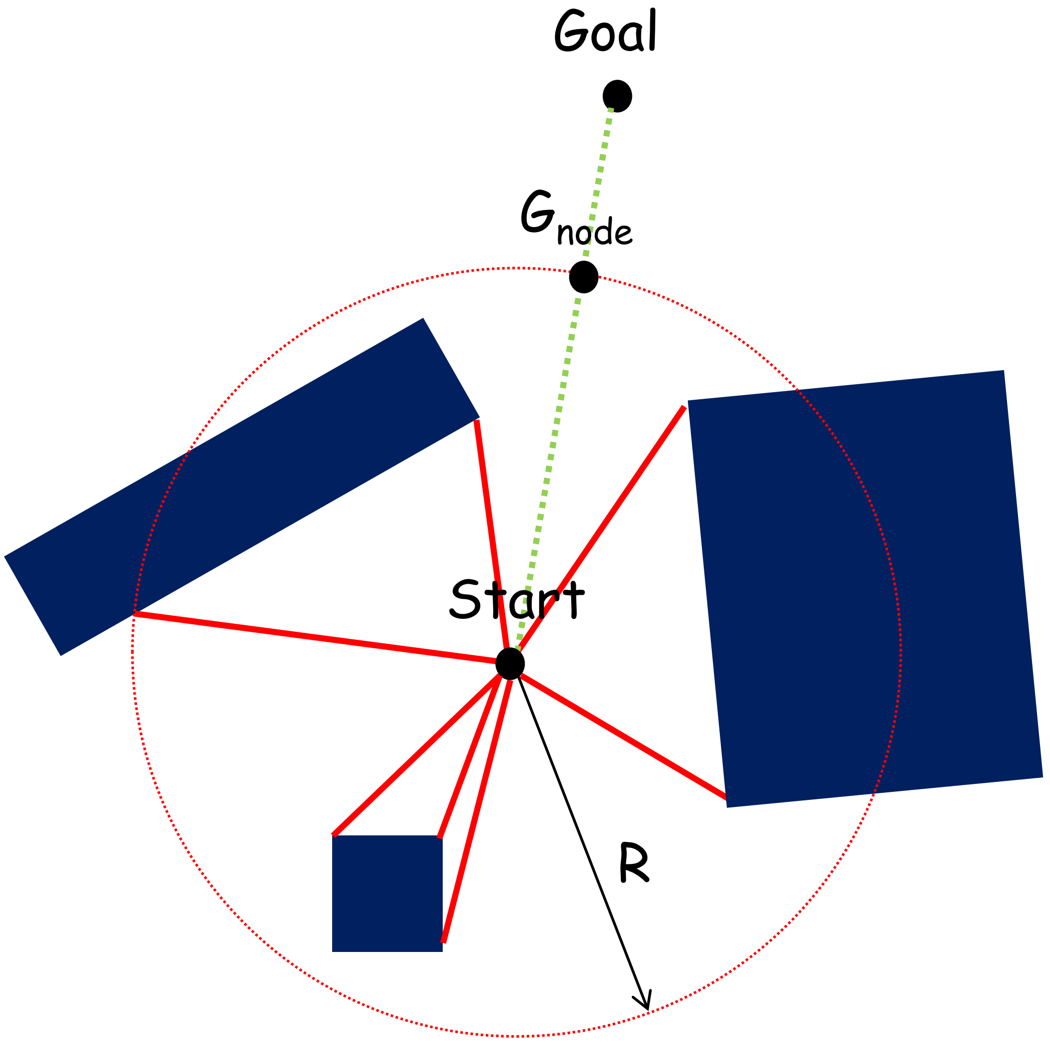
Kamon, I., Rimon, E. & Rivlin, E. "TangentBug: A Range-Sensor-Based Navigation Algorithm," The International Journal of Robotics Research. 17(9): 934-953, 1998.

Visibility graph & tangent graph

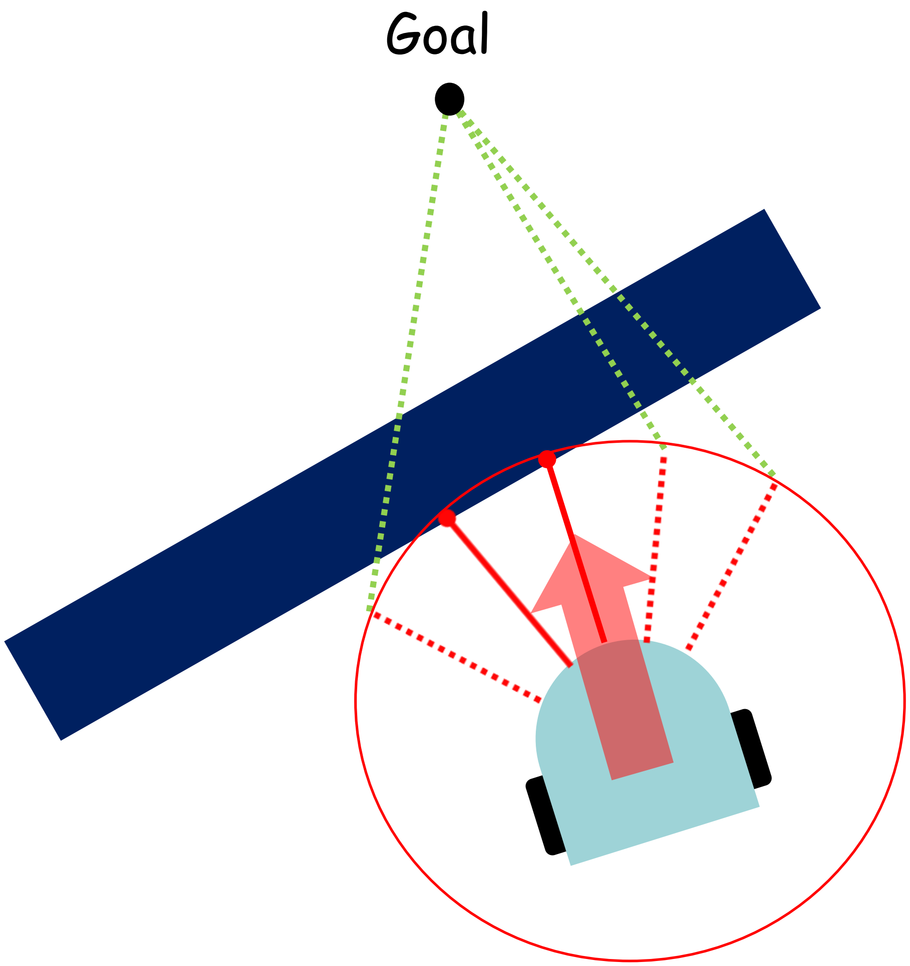




Local tangent graph

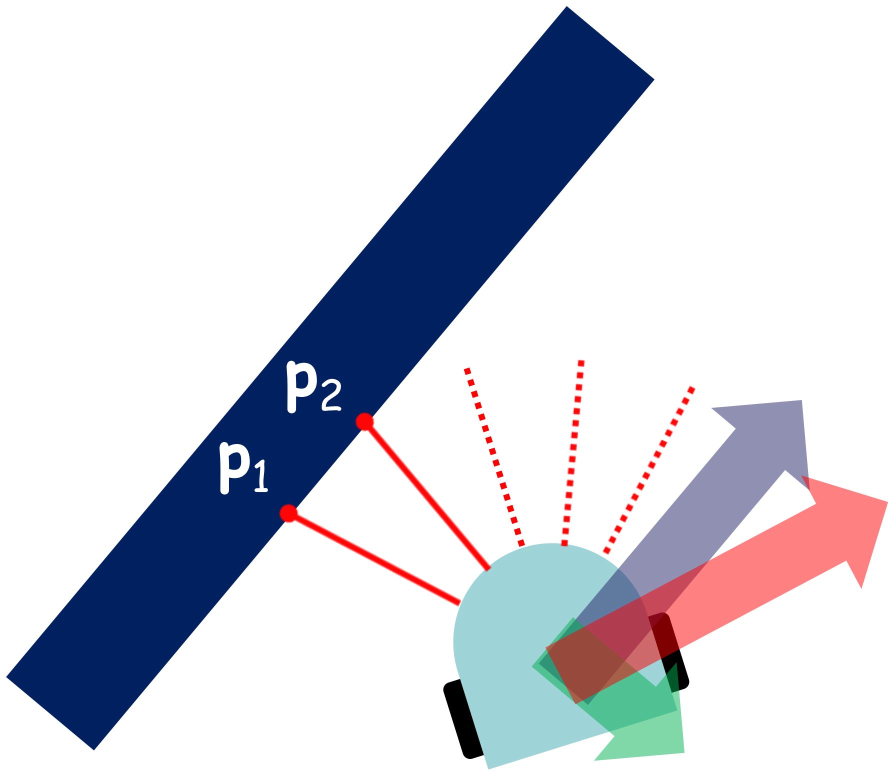


Local minimum detection



d(V, goal) < d(x, goal) for all V

Wall Following

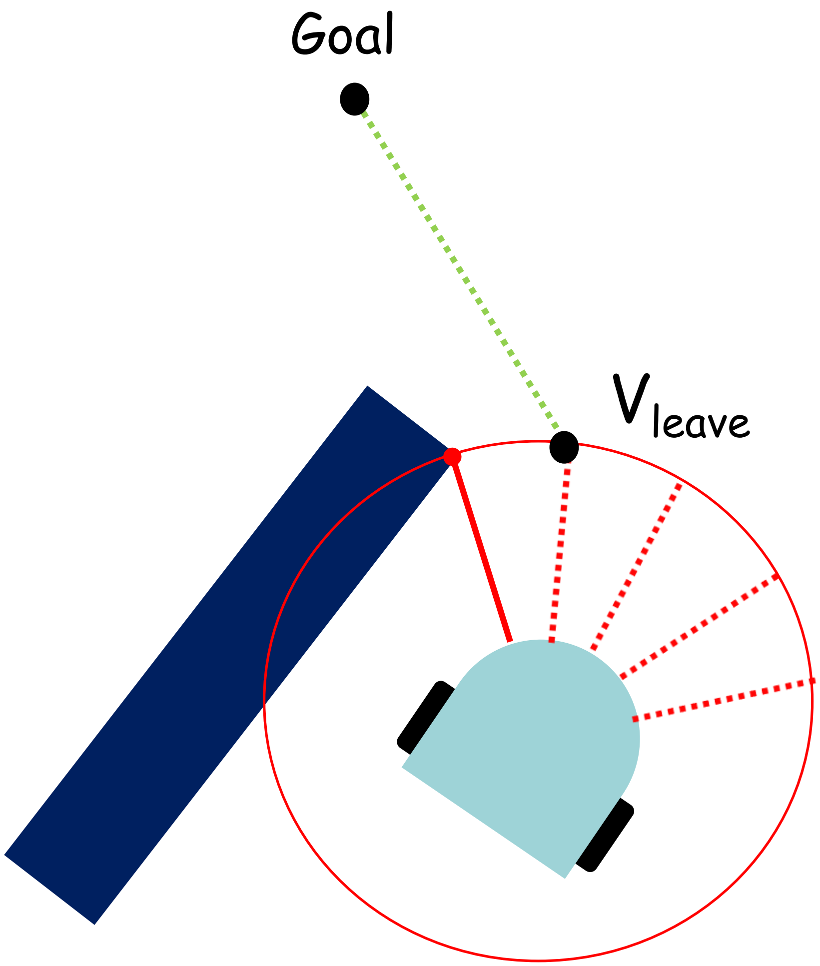


$\mathbf{v}_{wall} = \mathbf{p}_2 - \mathbf{p}_1$

$\mathbf{v}_{distance} = (d_{current} - d_{desired})\ \mathbf{v}_{perpedicular}$

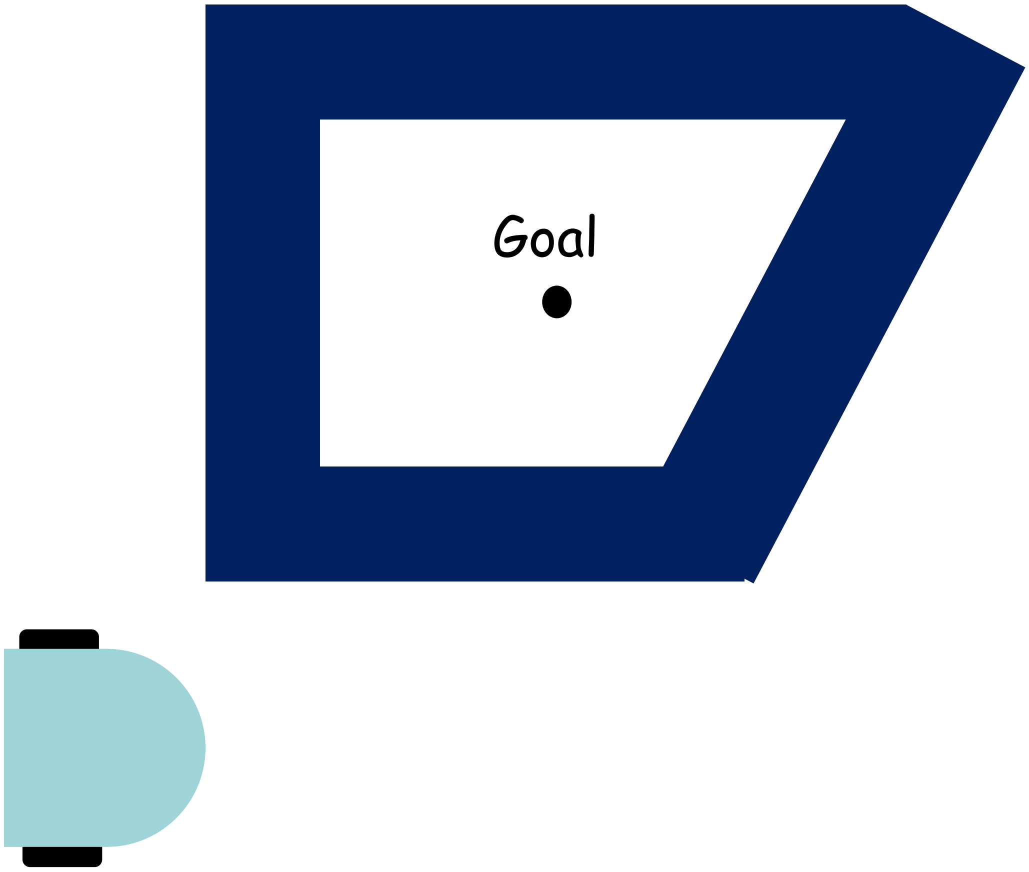
$\mathbf{v}_{robot} = d_{desired}\ \mathbf{v}_{wall} + \mathbf{v}_{distance}$

Leave condition detection

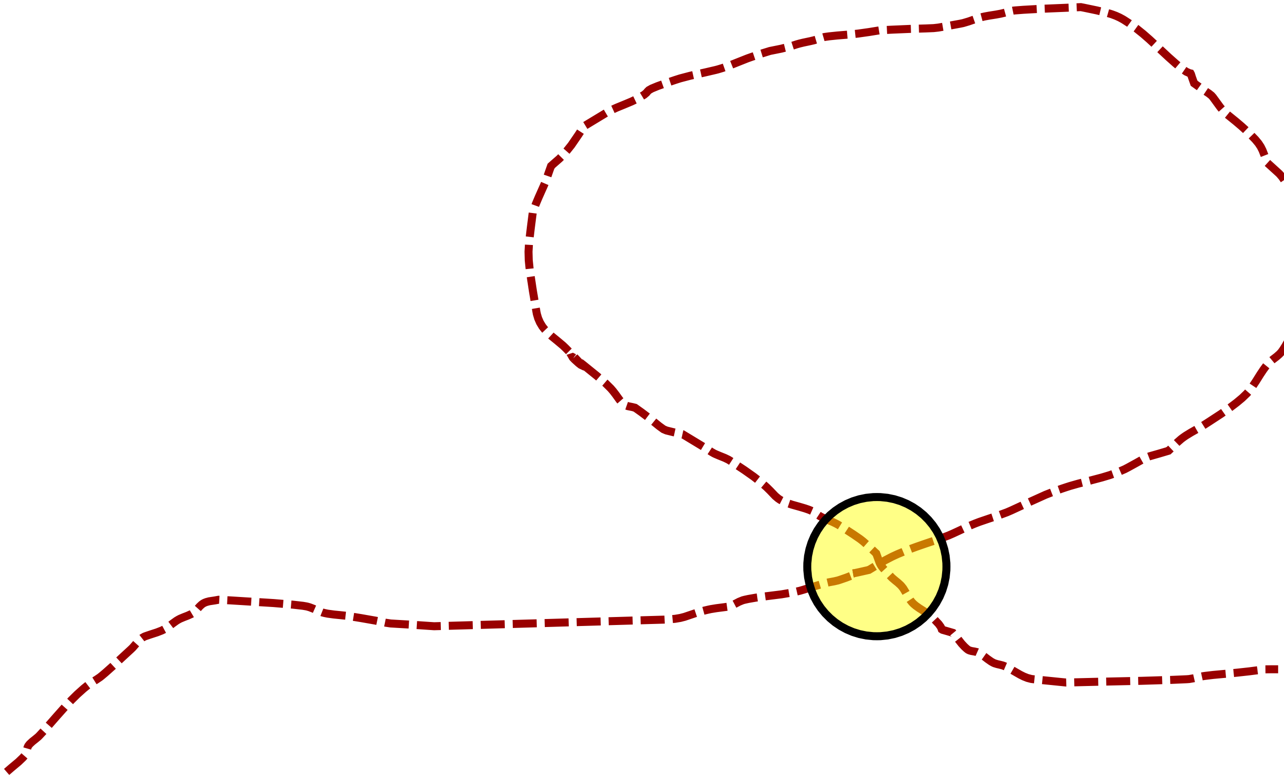


$$d(V_{leave}, goal) < d_{min}$$

Unreachable goal



Loop closure



Challenging!

- Drift
- Limited sensor information