FRA532 : Mobile Robot
Lecture 3
Mobile Robot Controller

Kitti Thamrongaphichartkul

Institute of Field Robotics
King Mongkut’s University of Technology Thonburi
Bangkok, Thailand

แผนการสอน

Week	Date	Lecture	Topic	Module	LAB / HW		Instructor	หมายเหตุ
					Assign	Due		
1	16-Jan-2025	1	Introduction to Mobile Robot (Motivation)				Aj.Nook	
2	23-Jan-2025	2	Kinematics of Mobile Robot		LAB 1		Aj.Nook	
3	30-Jan-2025	3	Mobile Robot Control				Aj.Nook	
4	6-Feb-2025	4	30 ปี ฟีโบ้				Aj.Nook	
5	13-Feb-2025	5	State Estimator				Aj.Nook	
6	20-Feb-2025		EXAM 1					
7	27-Feb-2025	6	MAP (Slam, Localization)		LAB 2	LAB 1	Aj.Nook	
8	4-March-2025	7	EXAM 1 / Hackathon Exam (24 Hour)				Aj.Nook	Project : Proposal
9	13-March-2025	8	Introduction to motion planning and discrete planning				Aj.Tee	
10	20-March-2025	9	Configuration space and sampling-based motion planning			LAB 2	Aj.Tee	
11	27-March-2025	10	Motion plannings used in self driving cars				Aj.Tee	
12	3-April-2025		EXAM 2					
13	10-April-2025	11	Project : Update 1				Aj.Nook	
14	18-April 2025	12	Special Topic I				Aj.Nook / Dummy	
15	24-April 2025	13	EXAM 2 / CBS + Nav2				Aj.Nook / Dummy	
16	1 May 2025	14	Special Topic III				Aj.Nook / Dummy	
17	8 May 2025	15	Project : Update 2				Aj.Nook	
18	15 May 2025	-	-					
19	22 May 2025	-	-					
20	29 May 2025	16	Project : Demo				Aj.Nook	

เนื้อหา

Link : https://github.com/kittinook/MobileRobotics2025/tree/main

Introduction to Mobile Robot

LAB 1

Kinematics of Mobile Robot

YOU are HERE

Robot Control

State Estimation

Exam I

LAB 2

Project : Proposal

Introduction to motion planning and discrete planning

Configuration space and sampling-based motion planning

Motion plannings used in self driving cars

Exam II

Project : Update 1

Special Topic

Special Topic

Special Topic

Special Topic

Project : Update 2

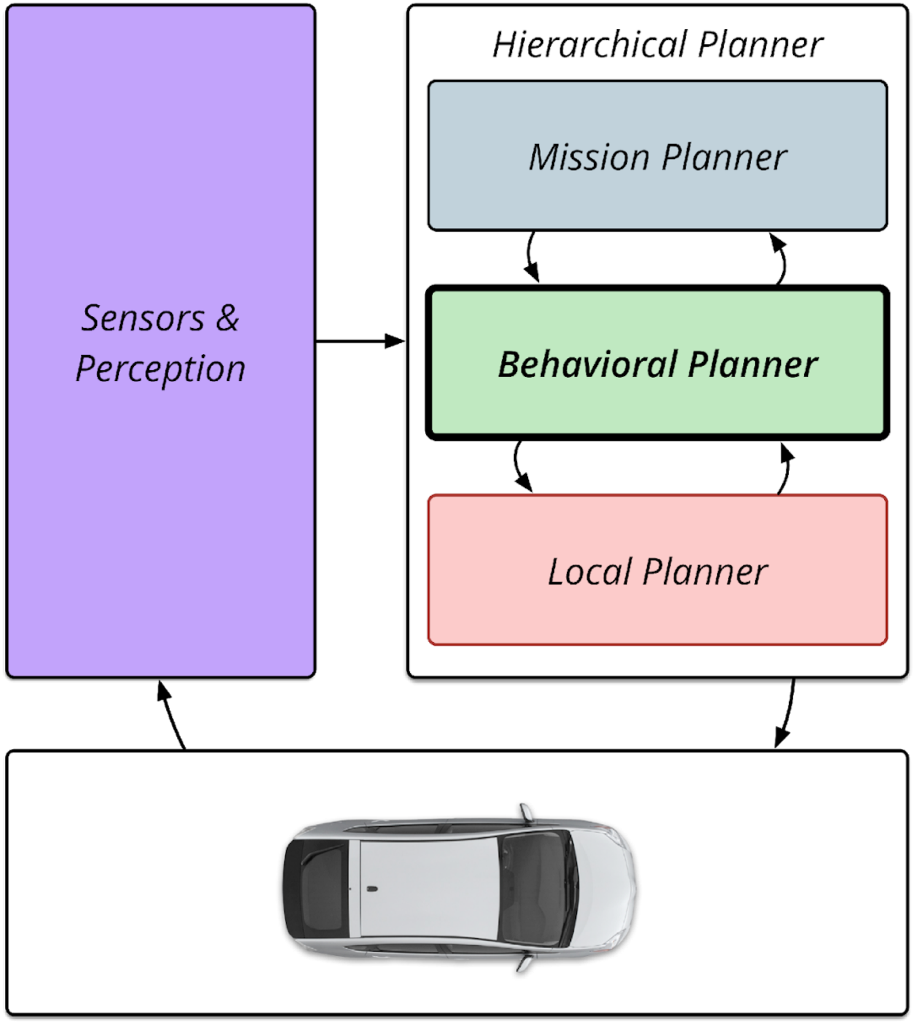
Project : Demo

Exam III

Agenda

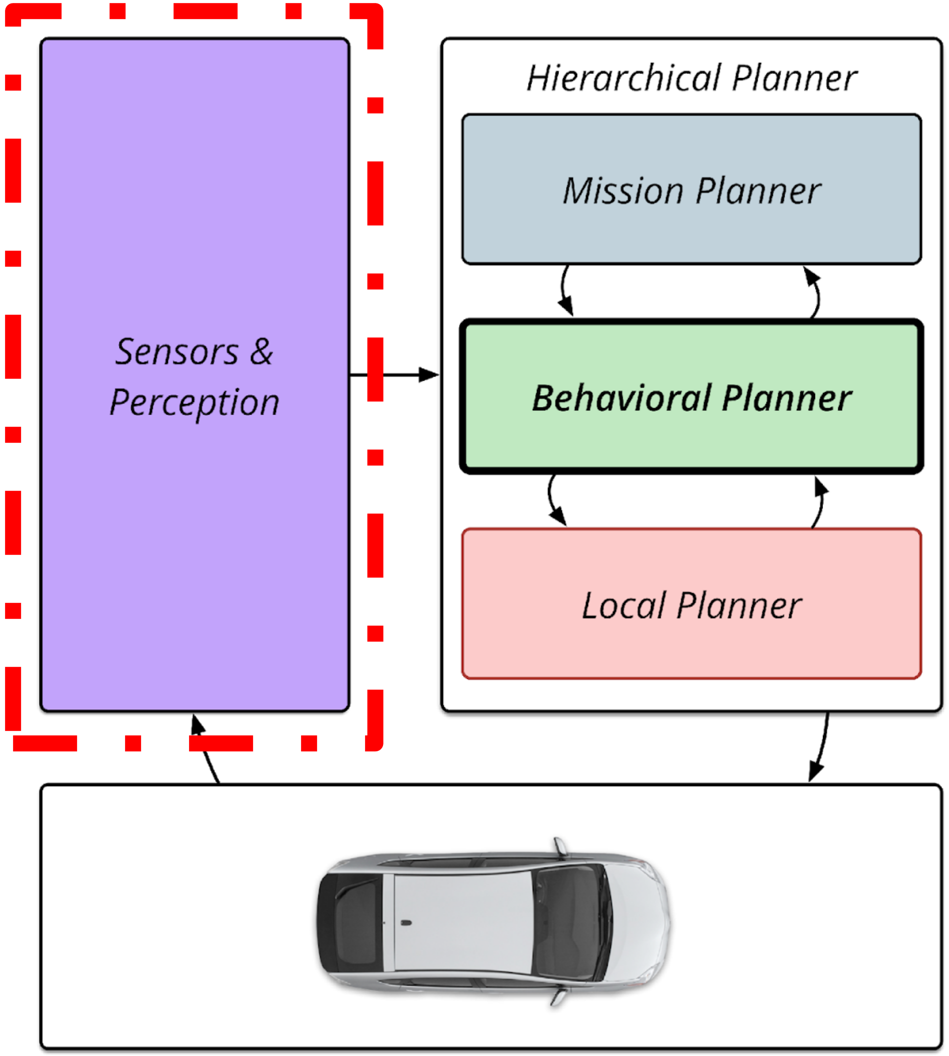
- Mobile Robot Framework
- Pure Pursuit Algorithm
- Potential Field
- Virtual Force Field

Mobile Robot Framework



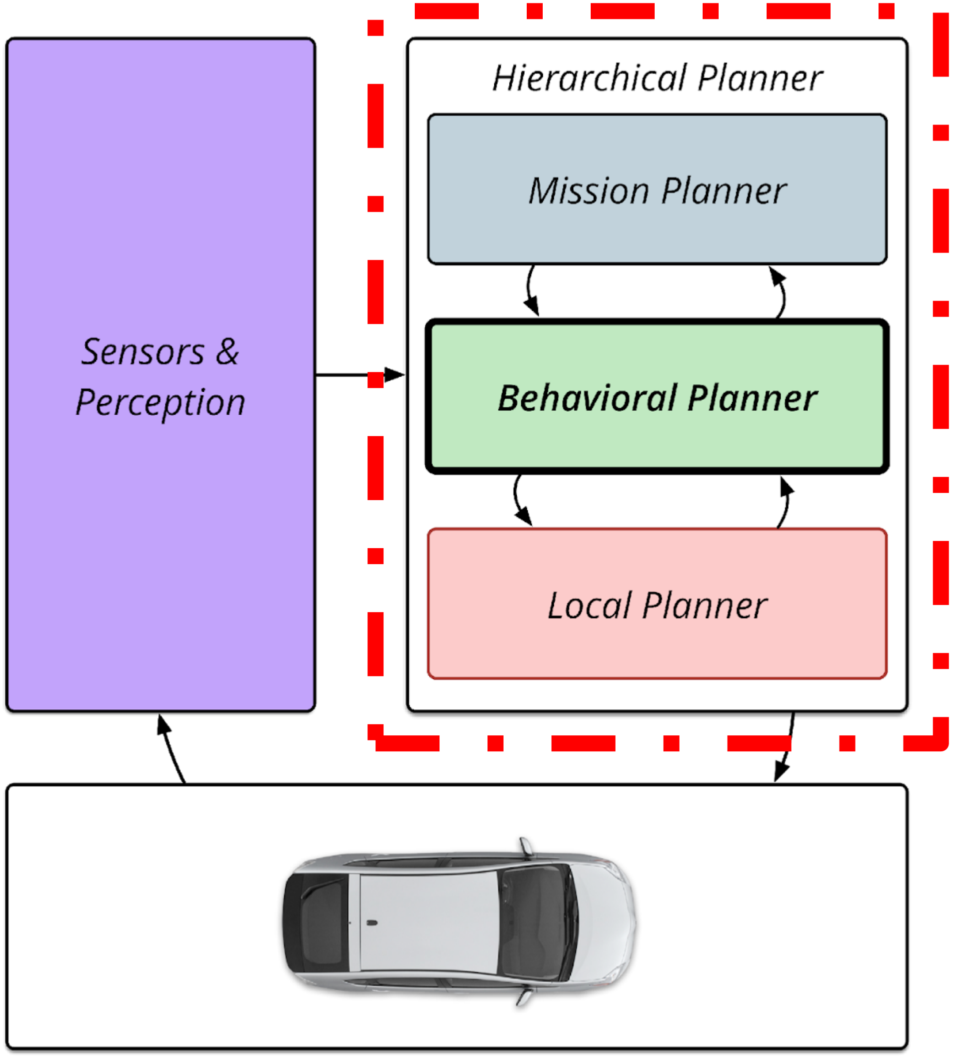
Sensors & Perception Module

- Camera
- Lidar
- Ultrasonic
- Contact
- Encoder



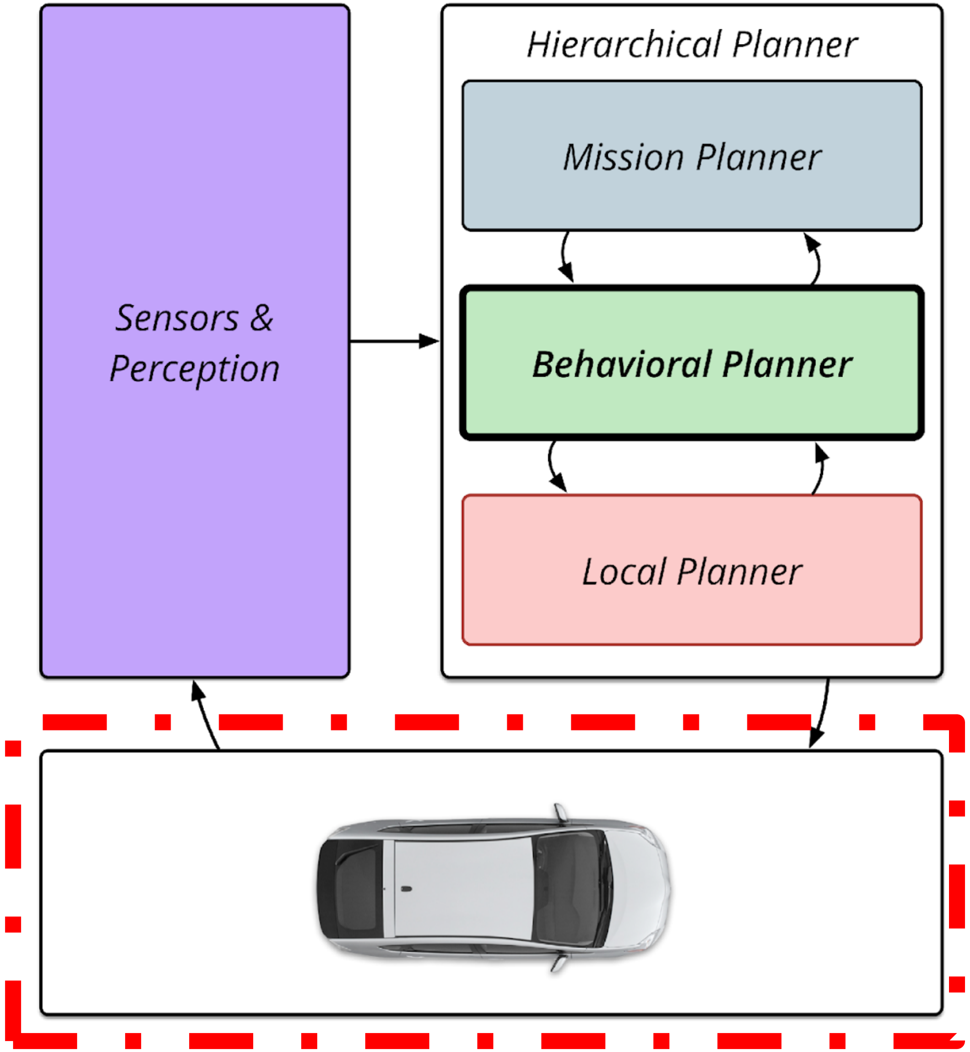
The Planning Module

- **Mission Planner**: what is the overall goal of the vehicle? **(Global Planner : Path)**
- **Behavioral Planner**: what rules should the vehicle follow in different situations? **(State Machine / Behavior Trees)**
- **Local Planner**: what is the optimal trajectory from position to a goal? **(Path tracking)**



The Control Module

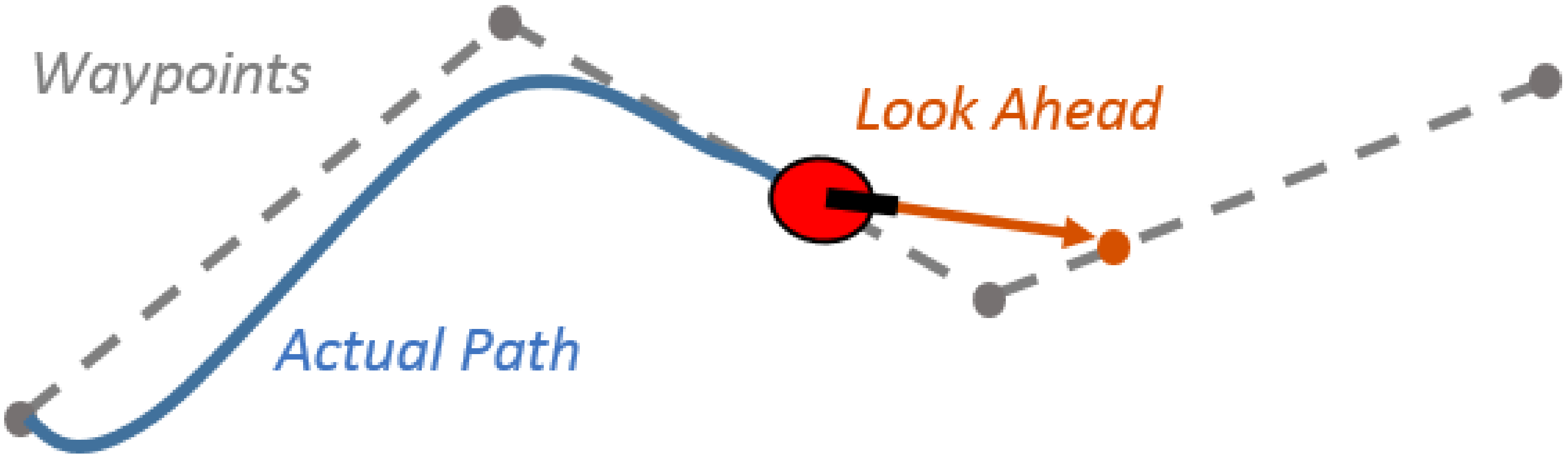
- How do we track a given trajectory?
- How do we correct for actuation errors?



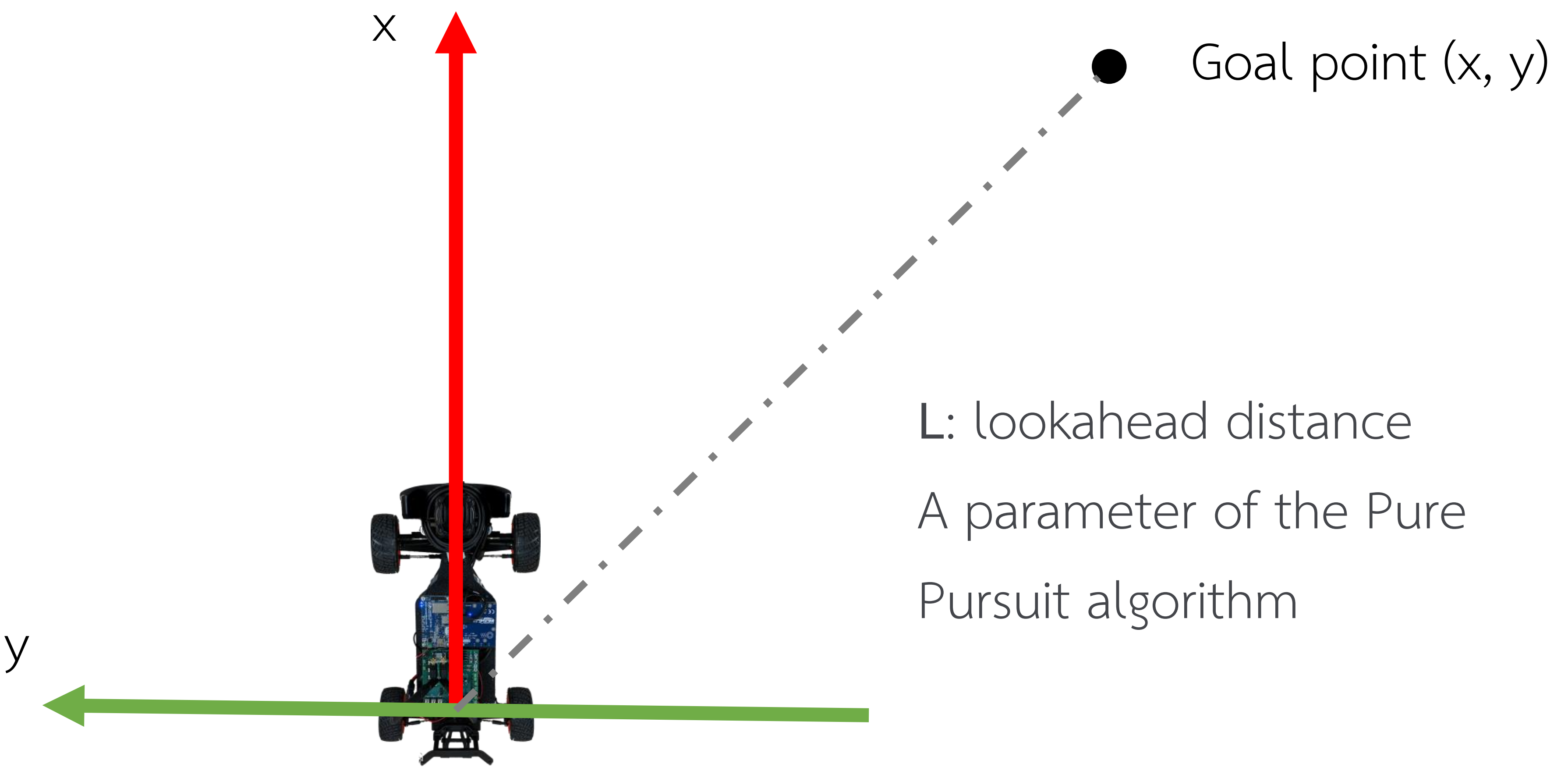
Pure Pursuit Algorithm

Pure Pursuit : Assumptions

- Vehicle is given a sequence of 2D positions, *i.e. waypoints,* to follow
- Vehicle knows where the given waypoints are in the vehicle's frame of reference
  - Underlying assumptions that the vehicle can localize itself
- Goal is to follow these waypoints



Pure Pursuit : Geometric Interpretation



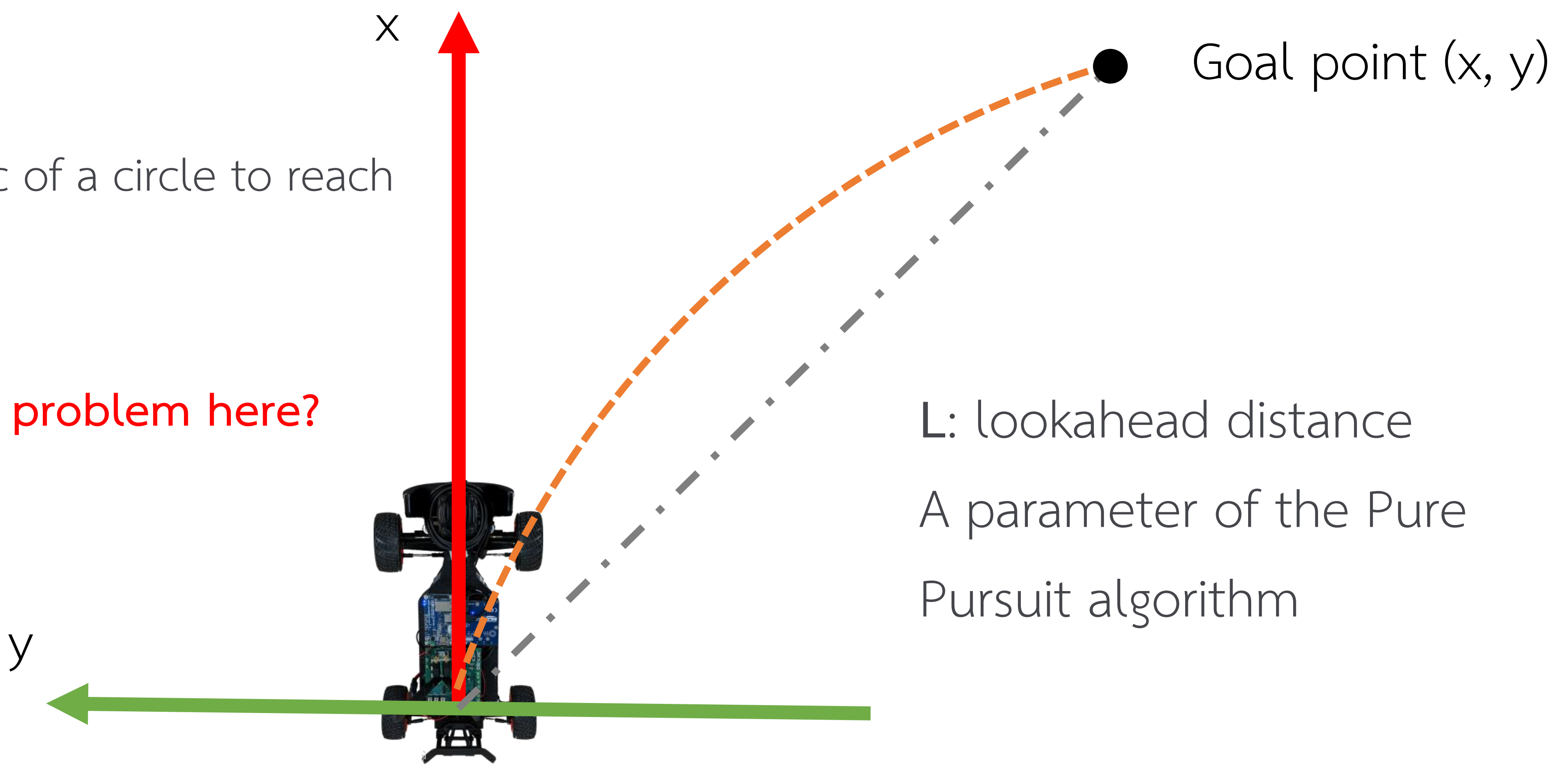
L: lookahead distance

A parameter of the Pure Pursuit algorithm

Pure Pursuit : Geometric Interpretation

Follow the arc of a circle to reach the goal point

Do you see a problem here?



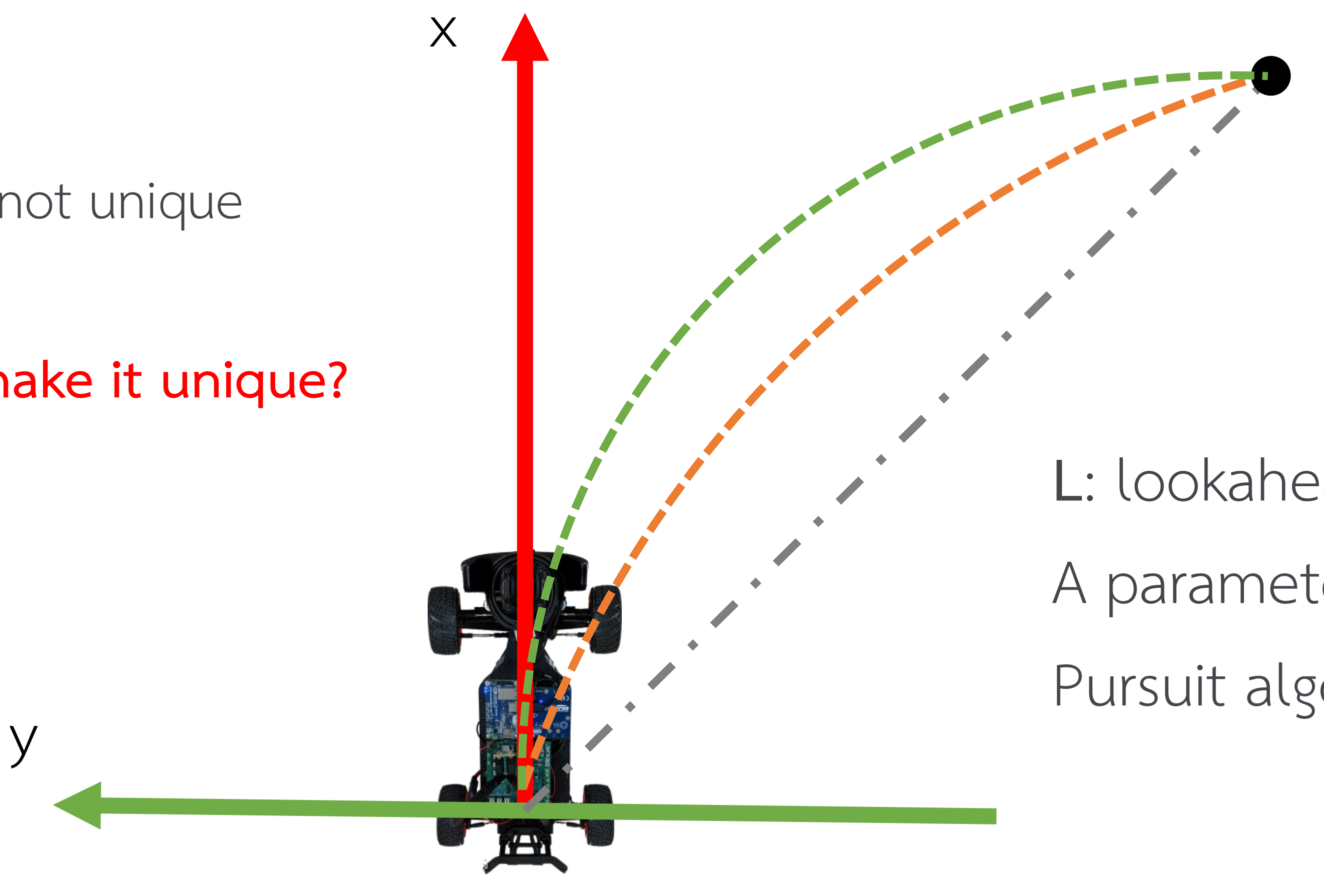
L: lookahead distance

A parameter of the Pure Pursuit algorithm

Pure Pursuit : Geometric Interpretation

But the arc is not unique

How do we make it unique?



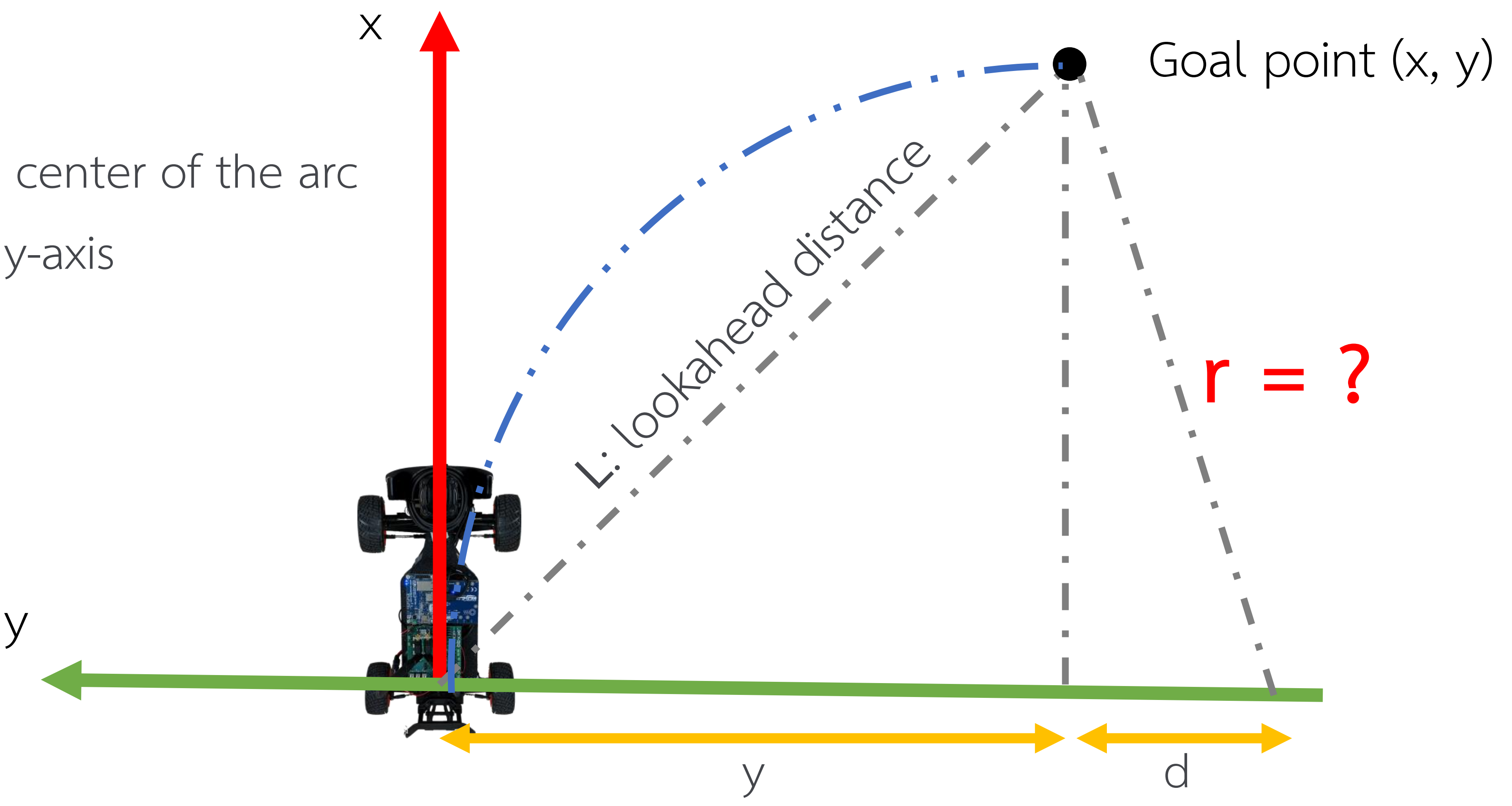
L: lookahead distance

A parameter of the Pure Pursuit algorithm

Pure Pursuit : Geometric Interpretation

Constrain the center of the arc

to be on the y-axis



Pure Pursuit : Geometric Equation

$$r = |y| + d$$

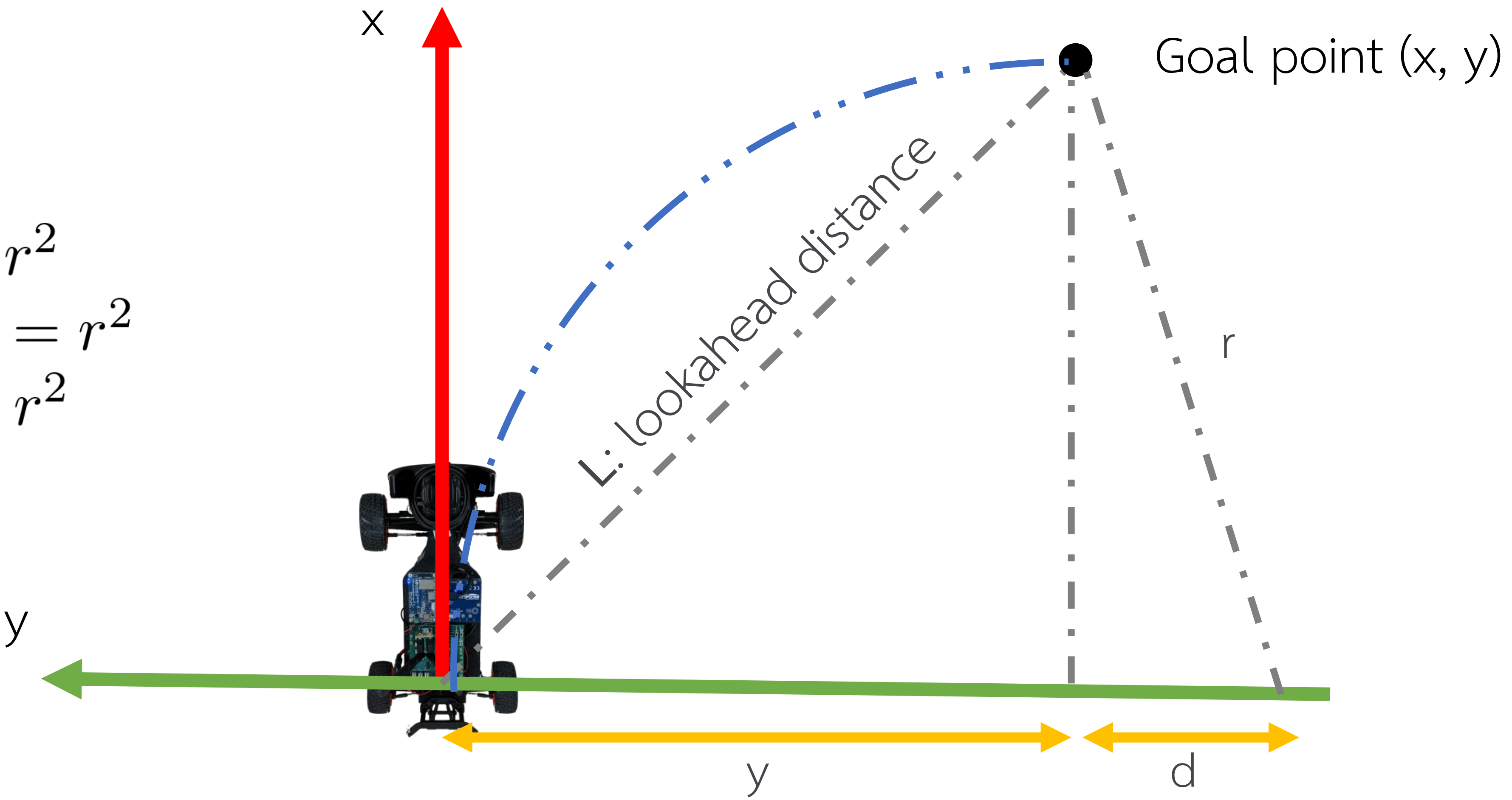
$$d^2 + x^2 = r^2$$

$$(r - |y|)^2 + x^2 = r^2$$

$$r^2 + y^2 - 2r|y| + x^2 = r^2$$

$$r^2 + L^2 - 2r|y| = r^2$$

$$r = \frac{L^2}{2|y|}$$



Pure Pursuit : How do we get steering angle?

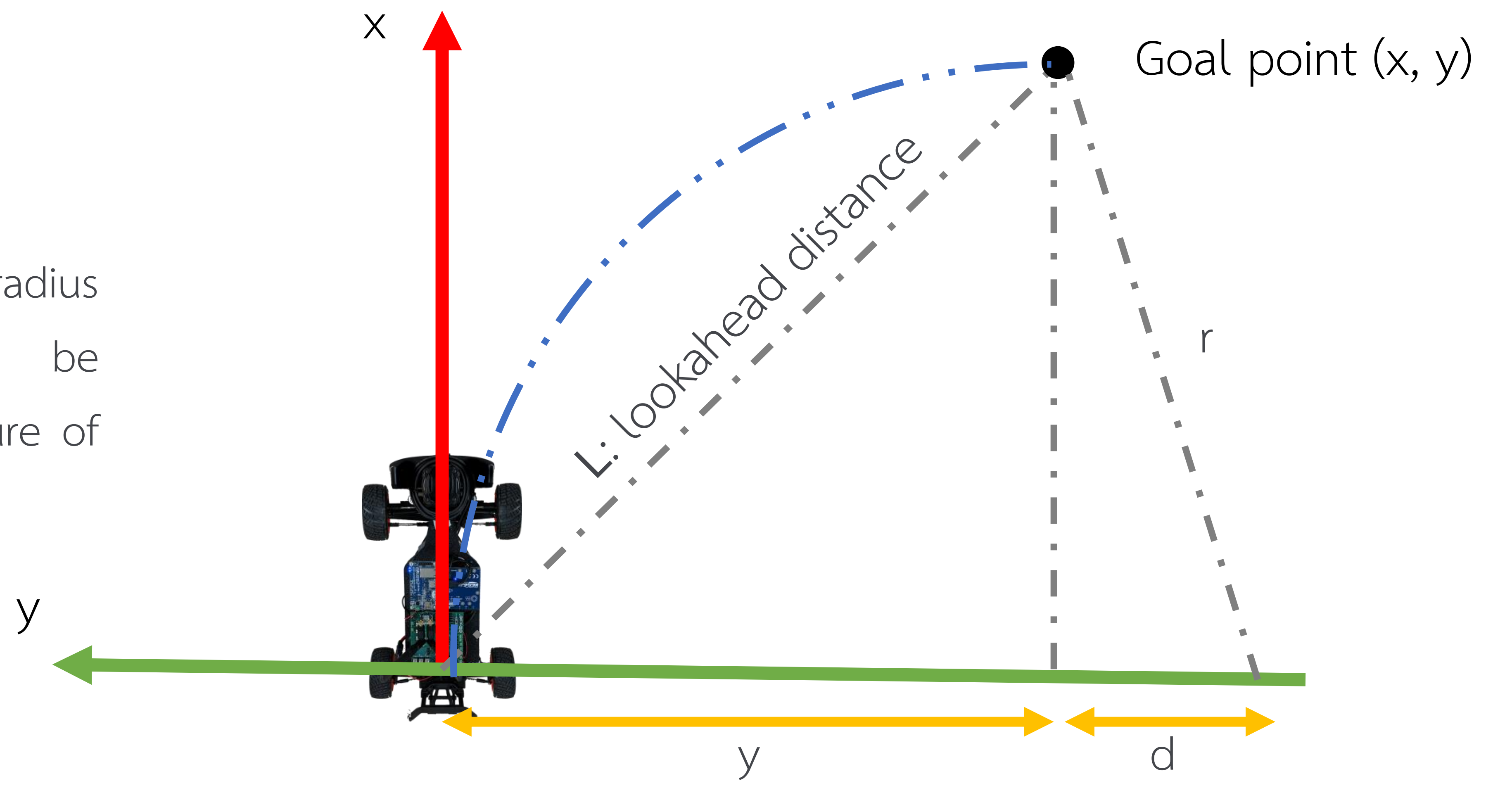
$$r = \frac{L^2}{2|y|}$$

Curvature is the inverse of radius

Steering angle should be **Proportional** to the curvature of the arc

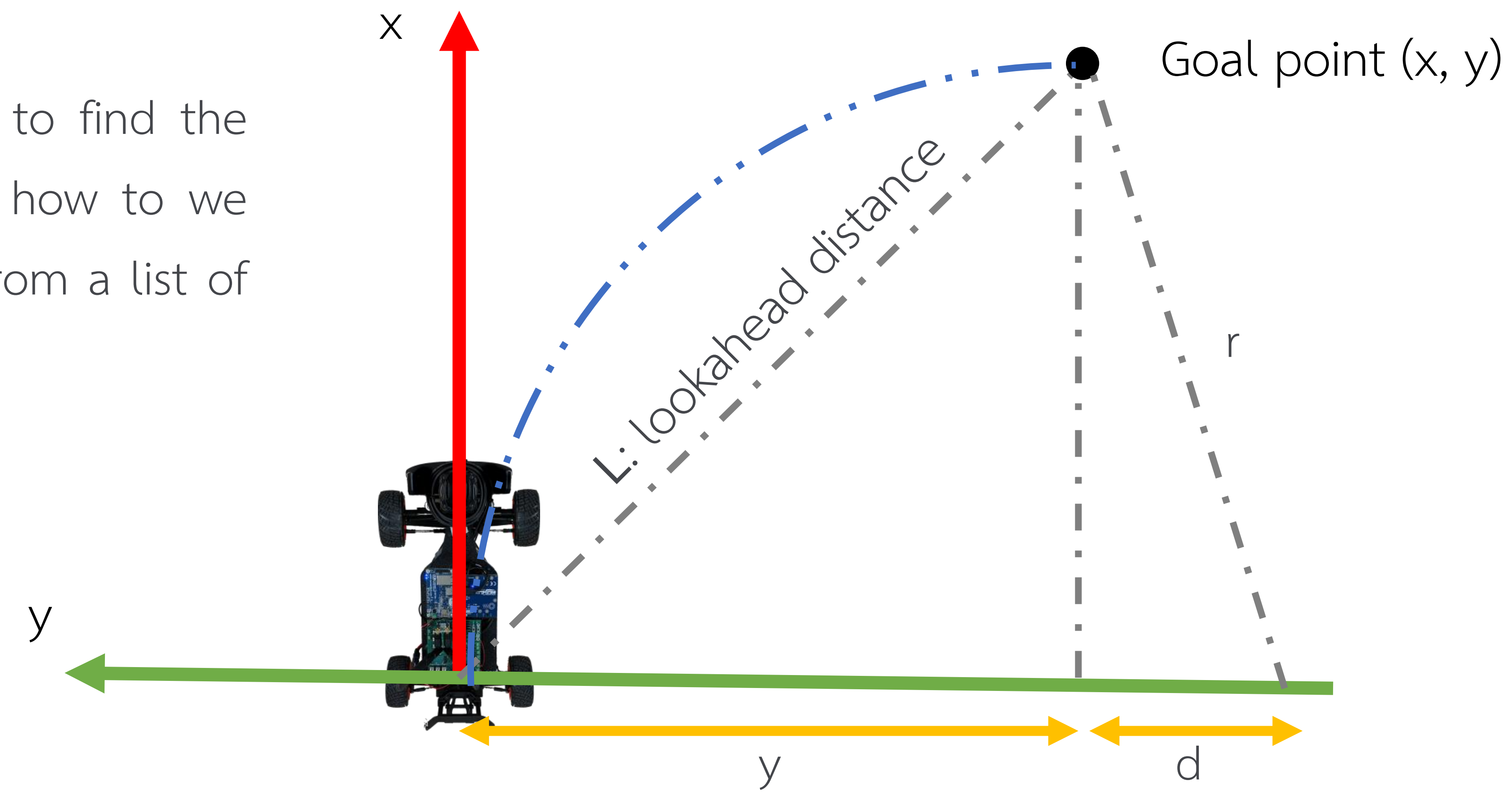
$$\gamma = \frac{1}{r} = \frac{2|y|}{L^2}$$

Look like P-Control

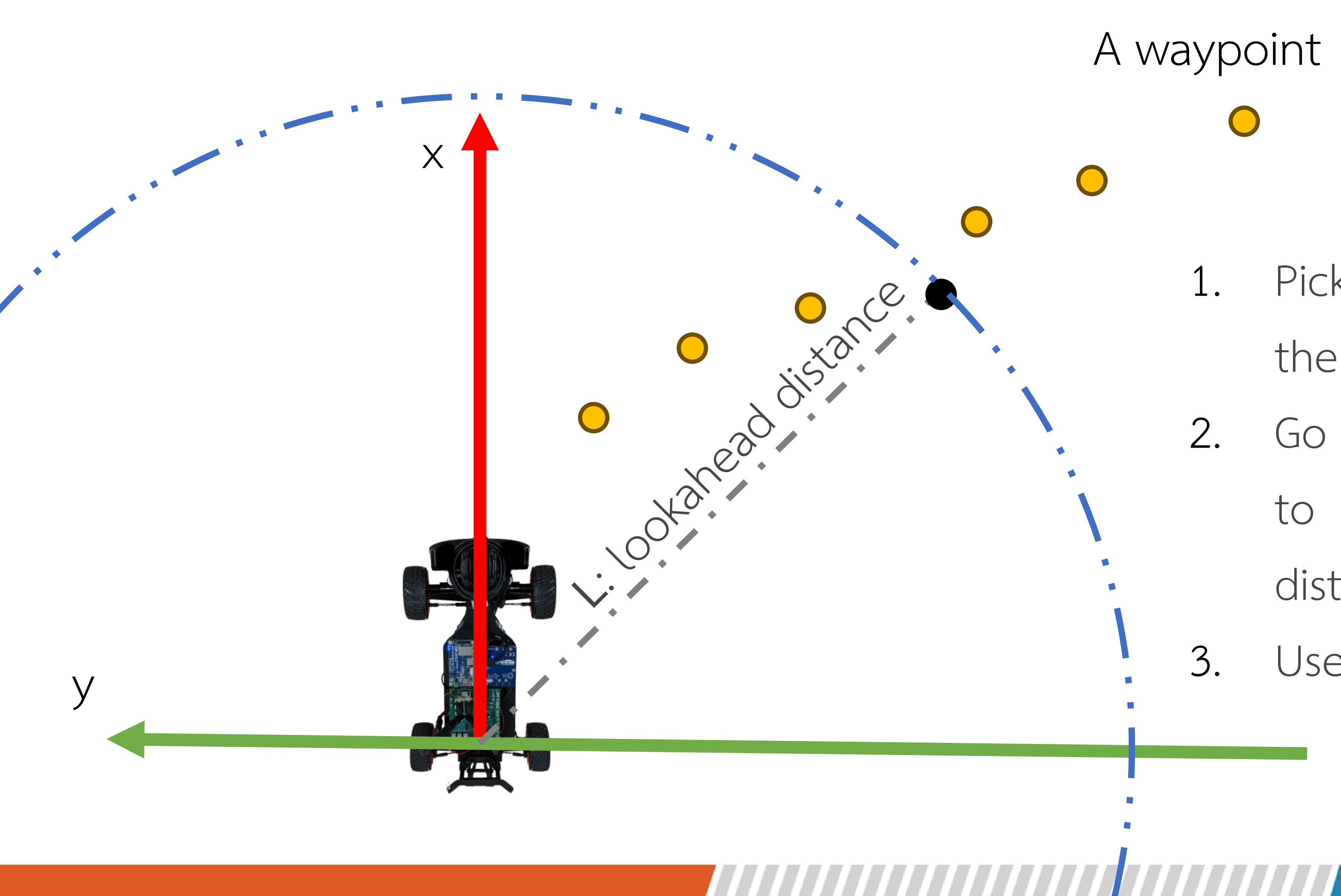


Pure Pursuit : Picking a goal point

Now that we know how to find the arc to a given waypoint, how to we pick a current waypoint from a list of waypoints?

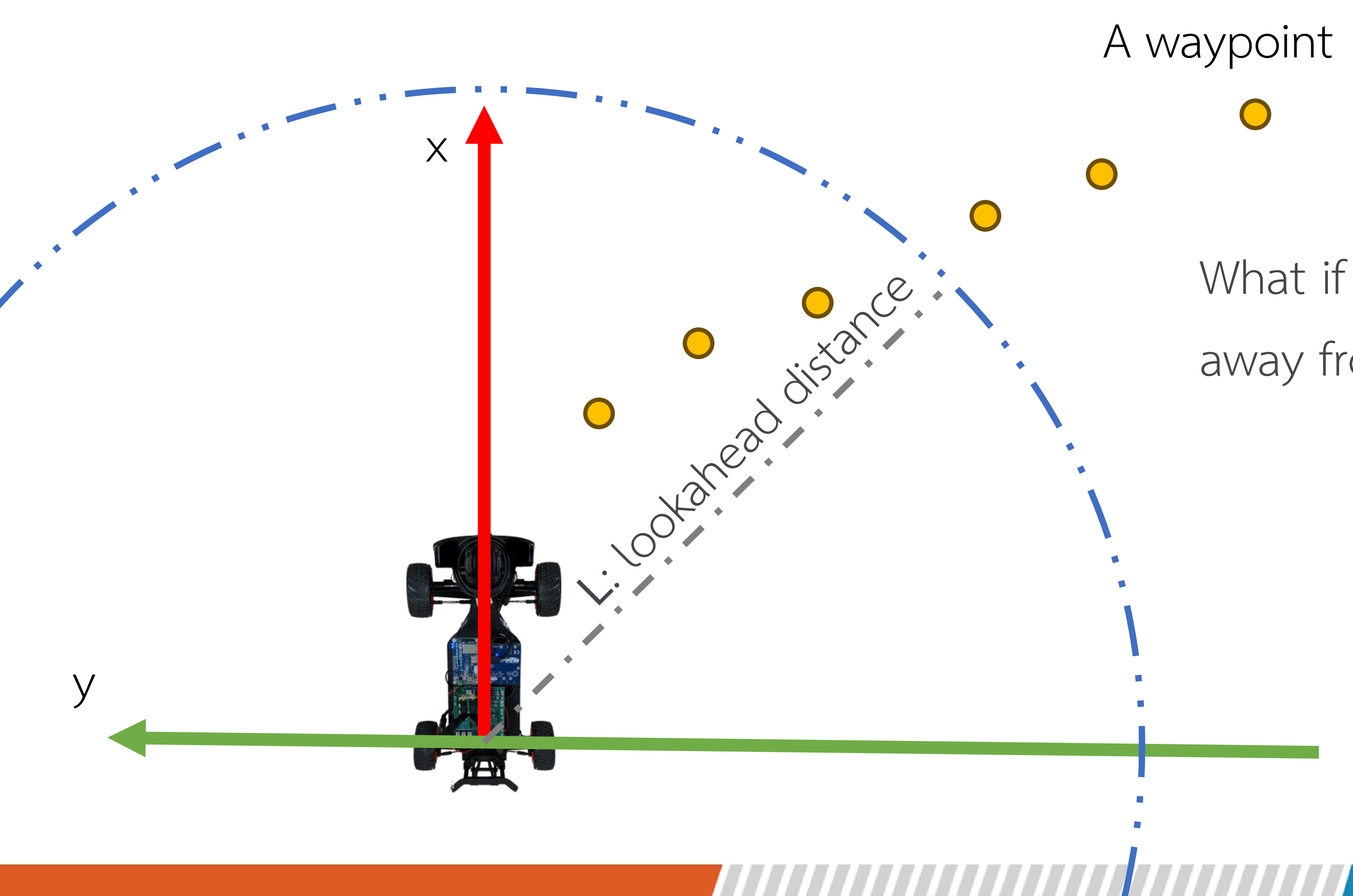


Pure Pursuit : Picking a goal point



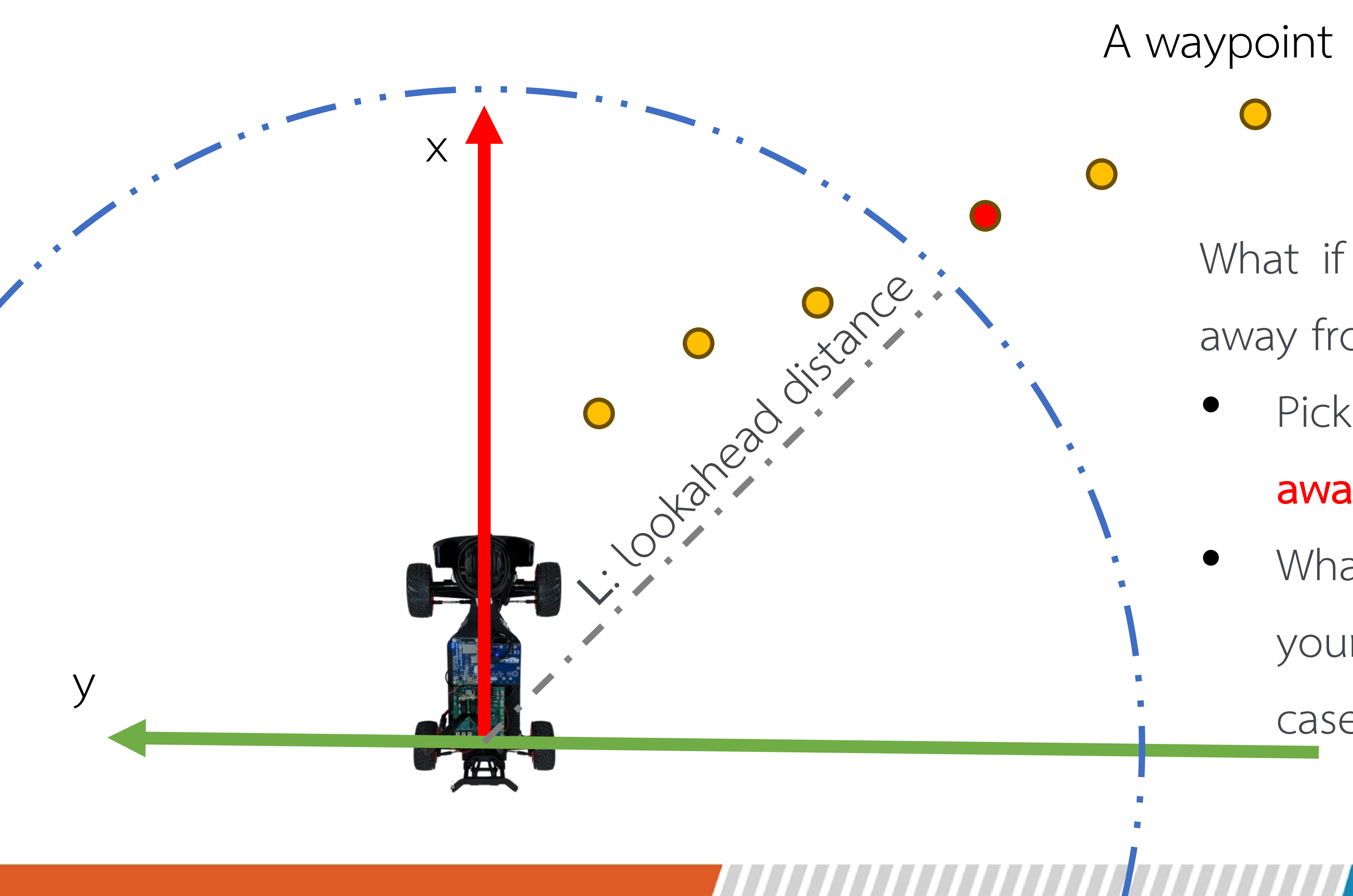
1. Pick the waypoint that is closest to the vehicle
2. Go up to the waypoint until you get to one that is one lookahead distance away from the car
3. Use that as the current waypoint

Pure Pursuit : Picking a goal point



What if there’s no waypoints exactly L away from the car?

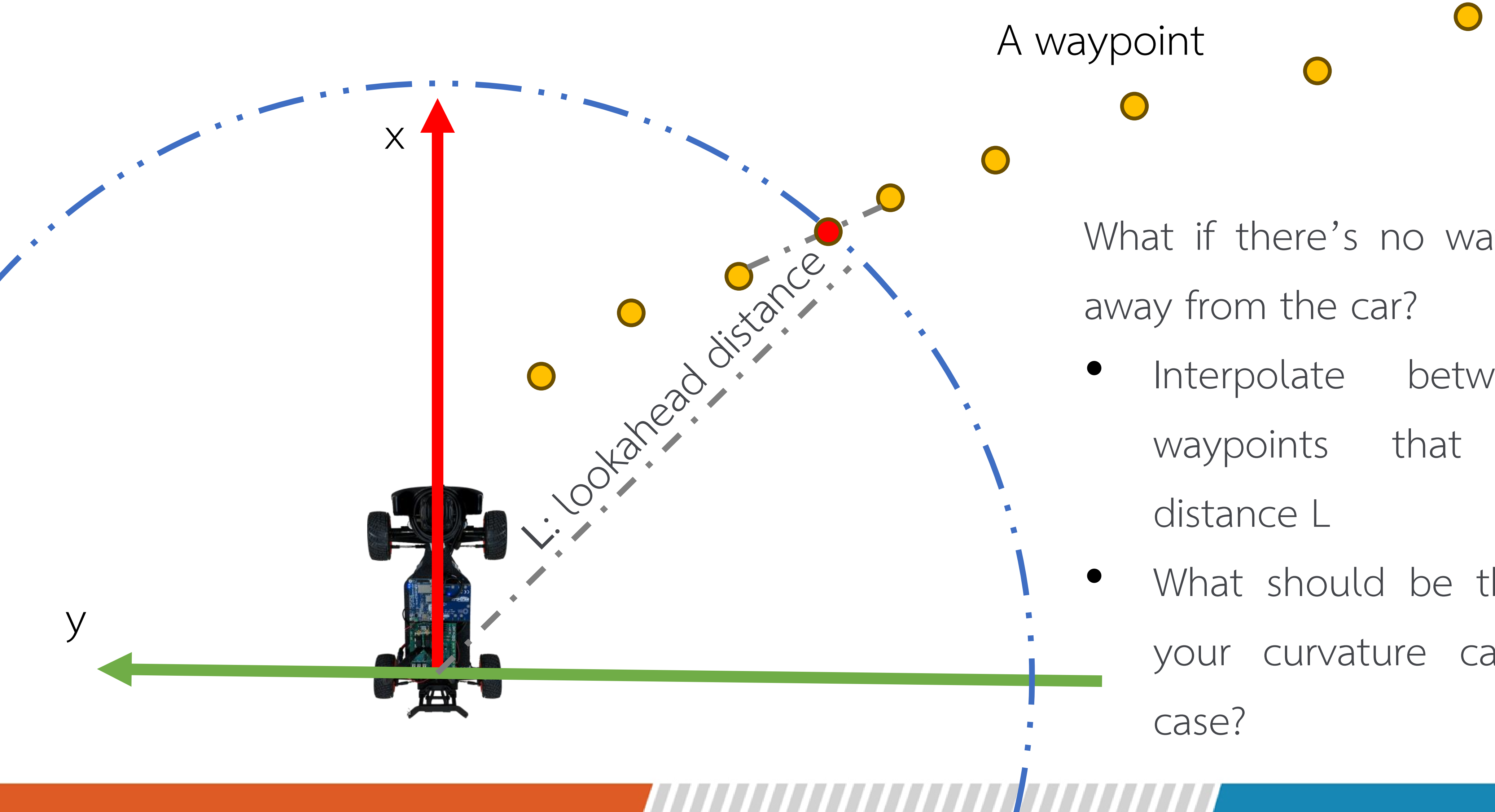
Pure Pursuit : Picking a goal point



What if there's no waypoints exactly L away from the car?

- Pick the next best one **(closest to L away)**
- What should be the value of L in your curvature calculation in this case?

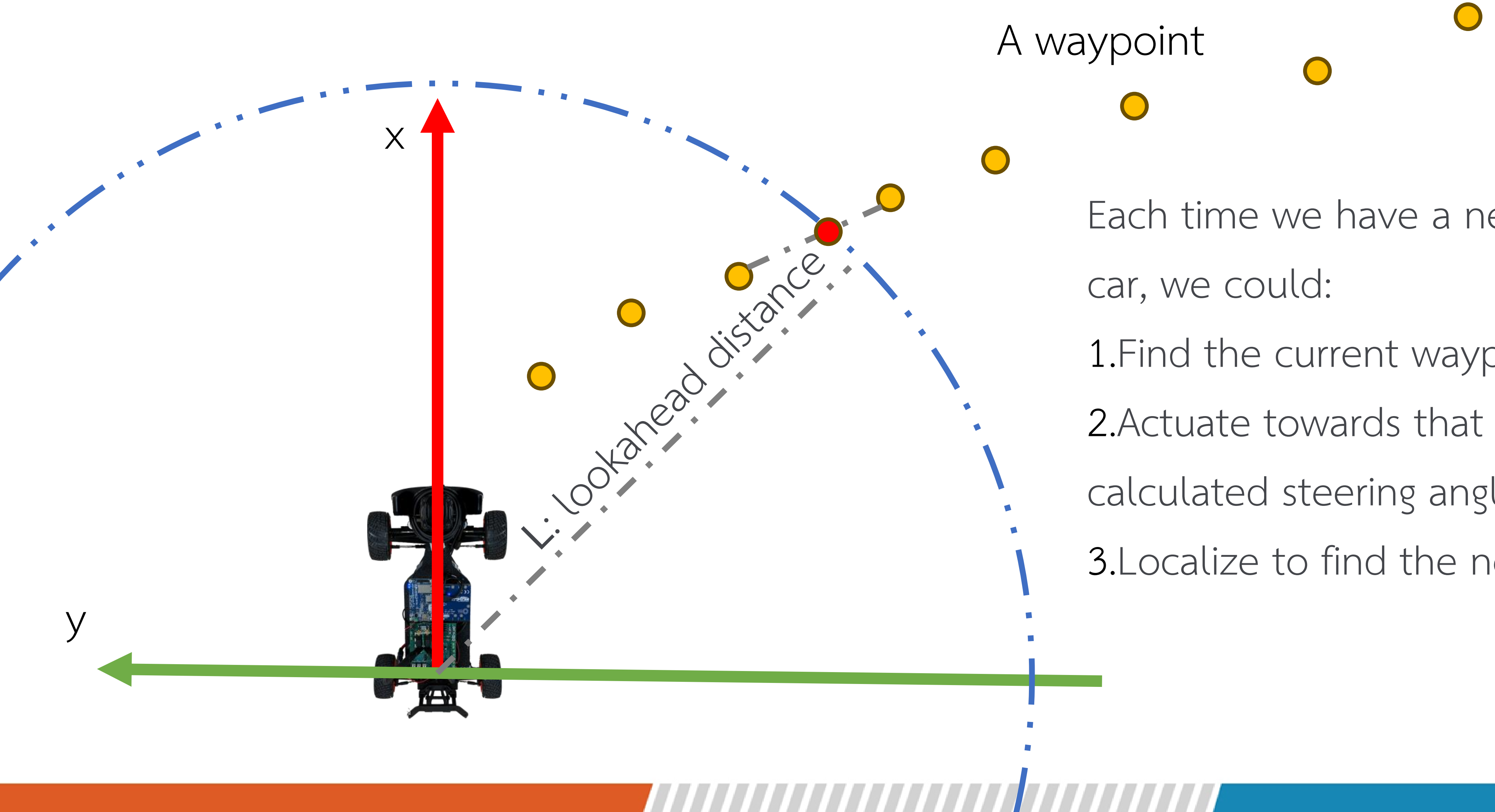
Pure Pursuit : Picking a goal point



What if there's no waypoints exactly L away from the car?

- Interpolate between the two waypoints that sandwich the distance L
- What should be the value of L in your curvature calculation in this case?

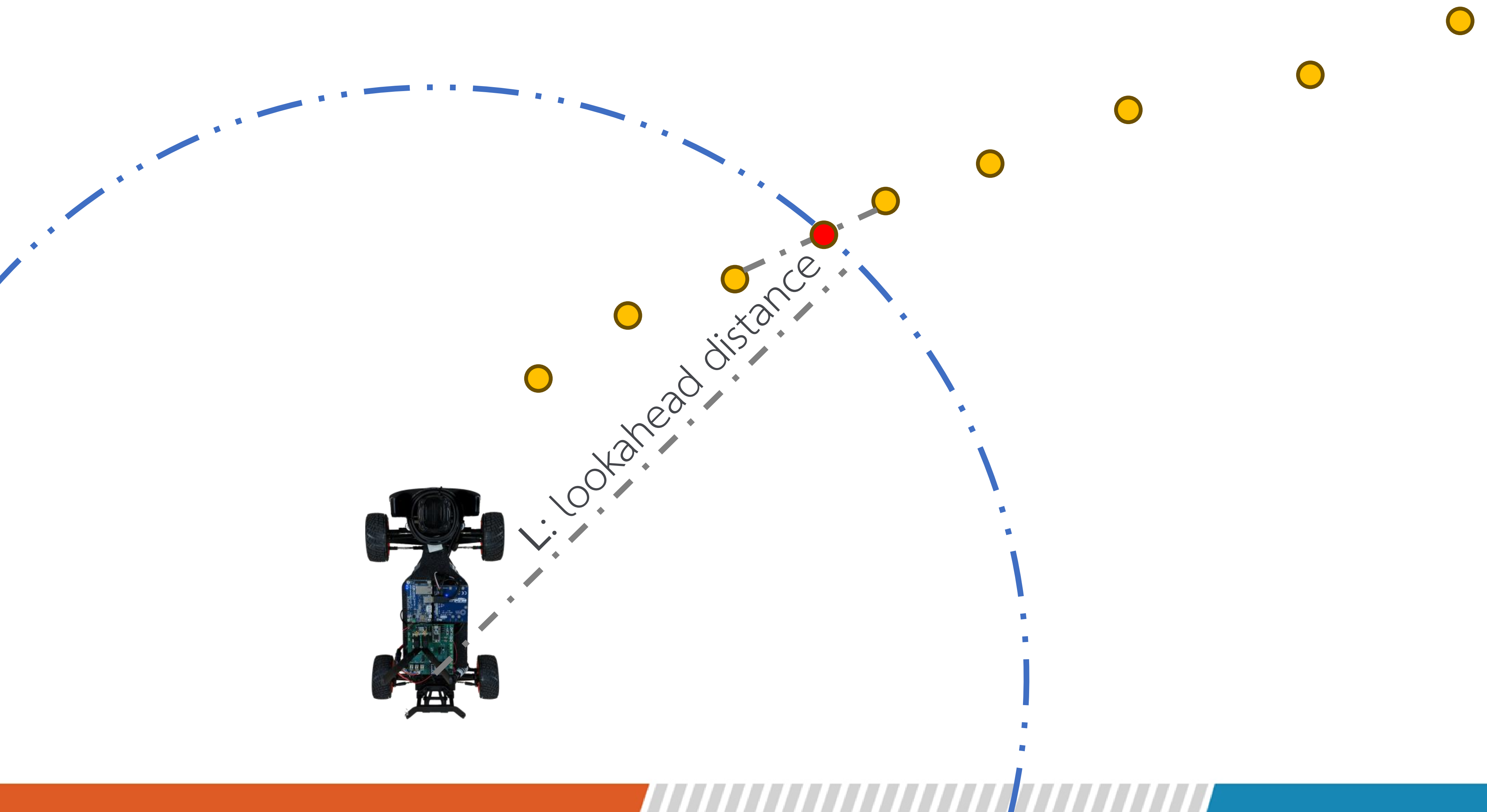
Pure Pursuit : Updating the goal point



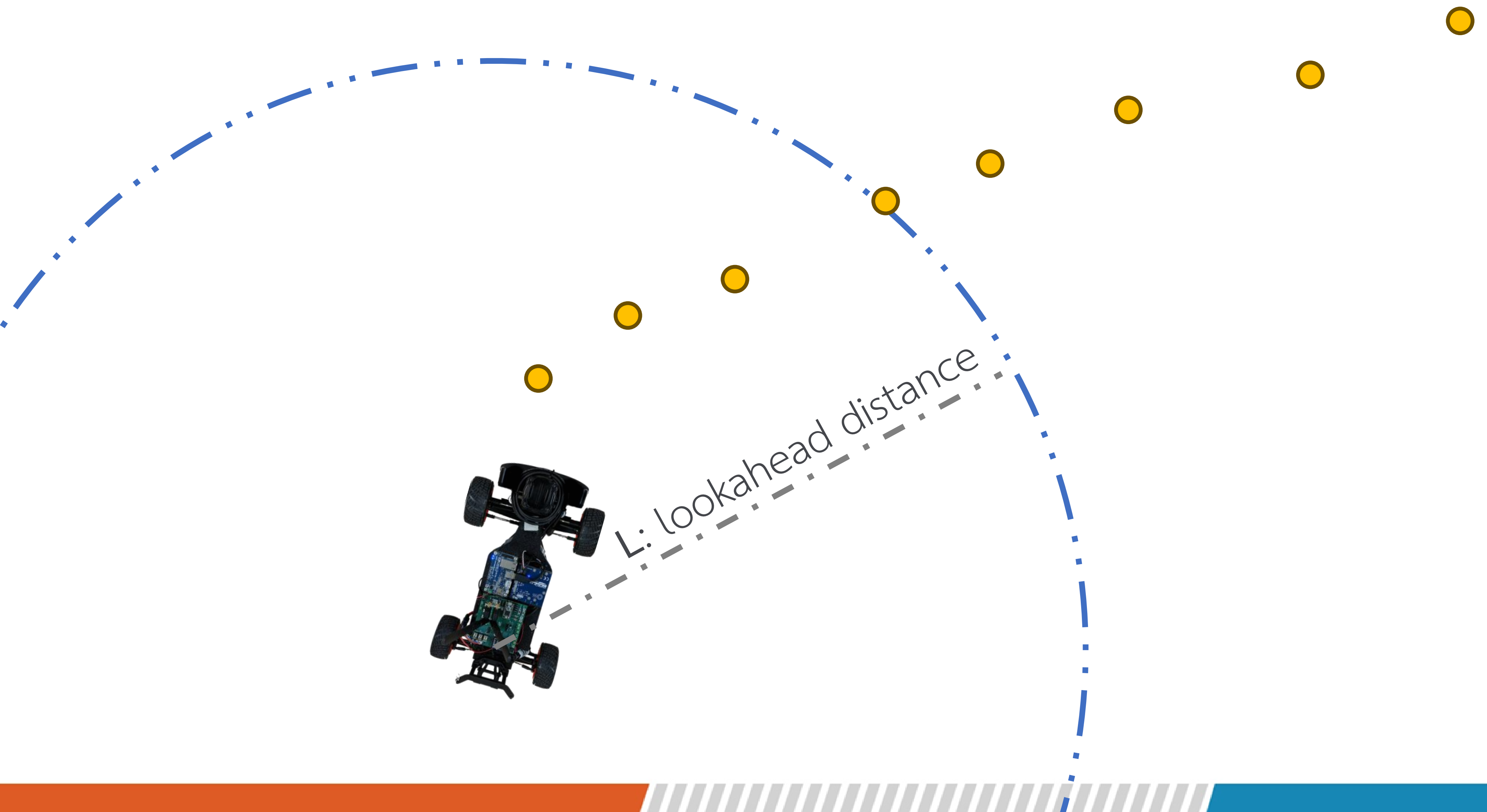
Each time we have a new pose of the car, we could:

1. Find the current waypoint
2. Actuate towards that waypoint with calculated steering angle
3. Localize to find the new pose, repeat

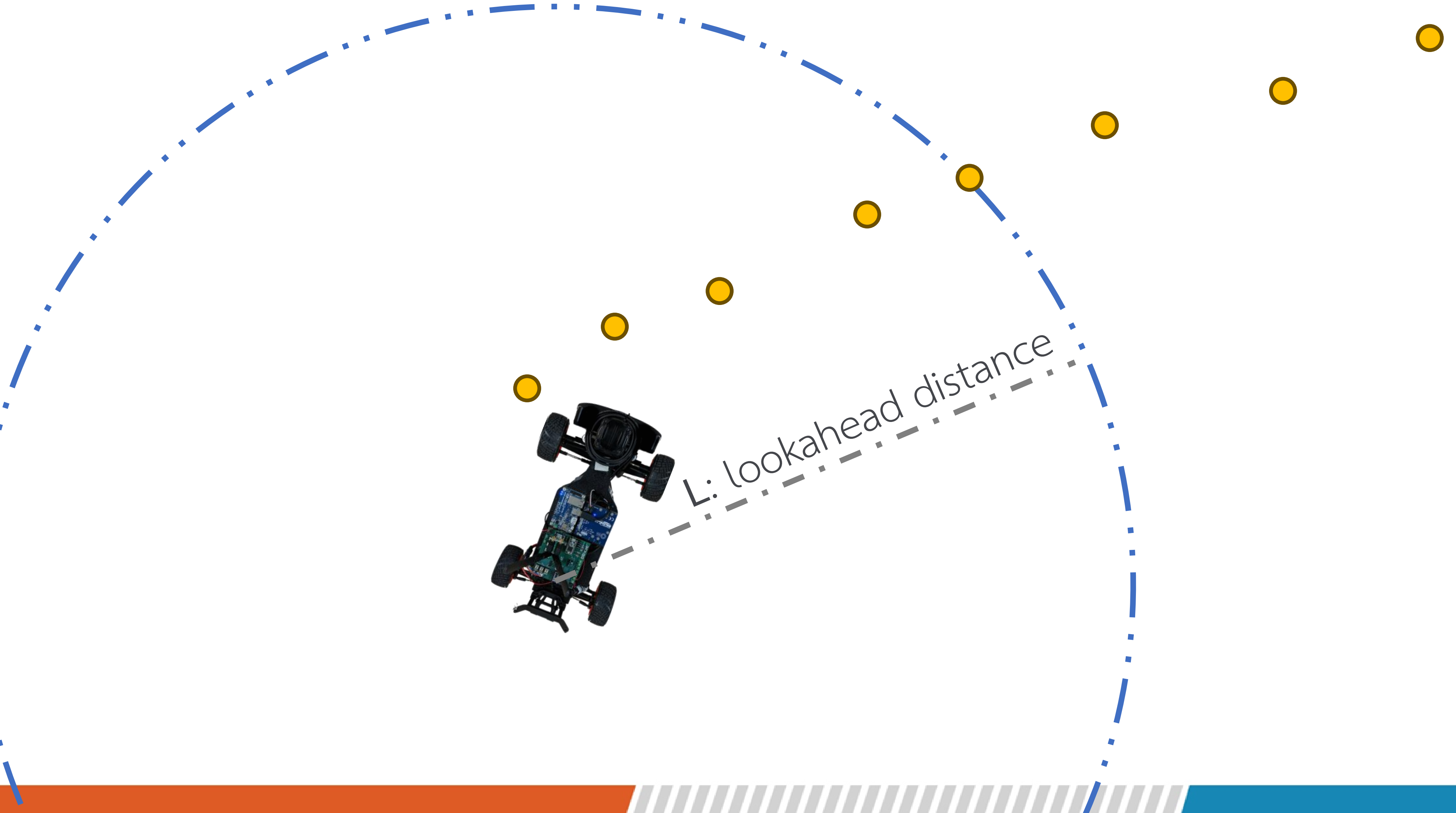
Pure Pursuit : Updating the goal point



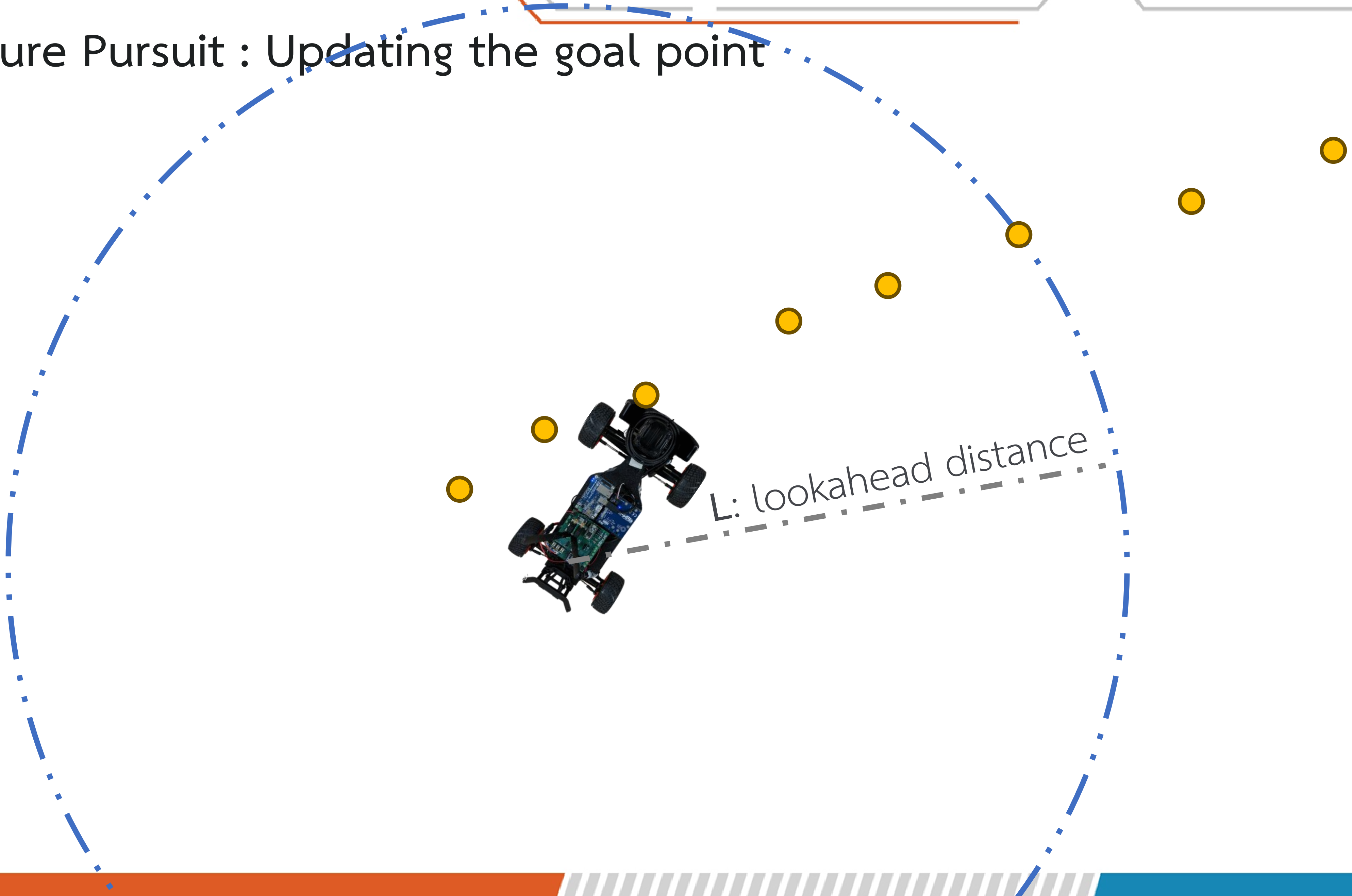
Pure Pursuit : Updating the goal point



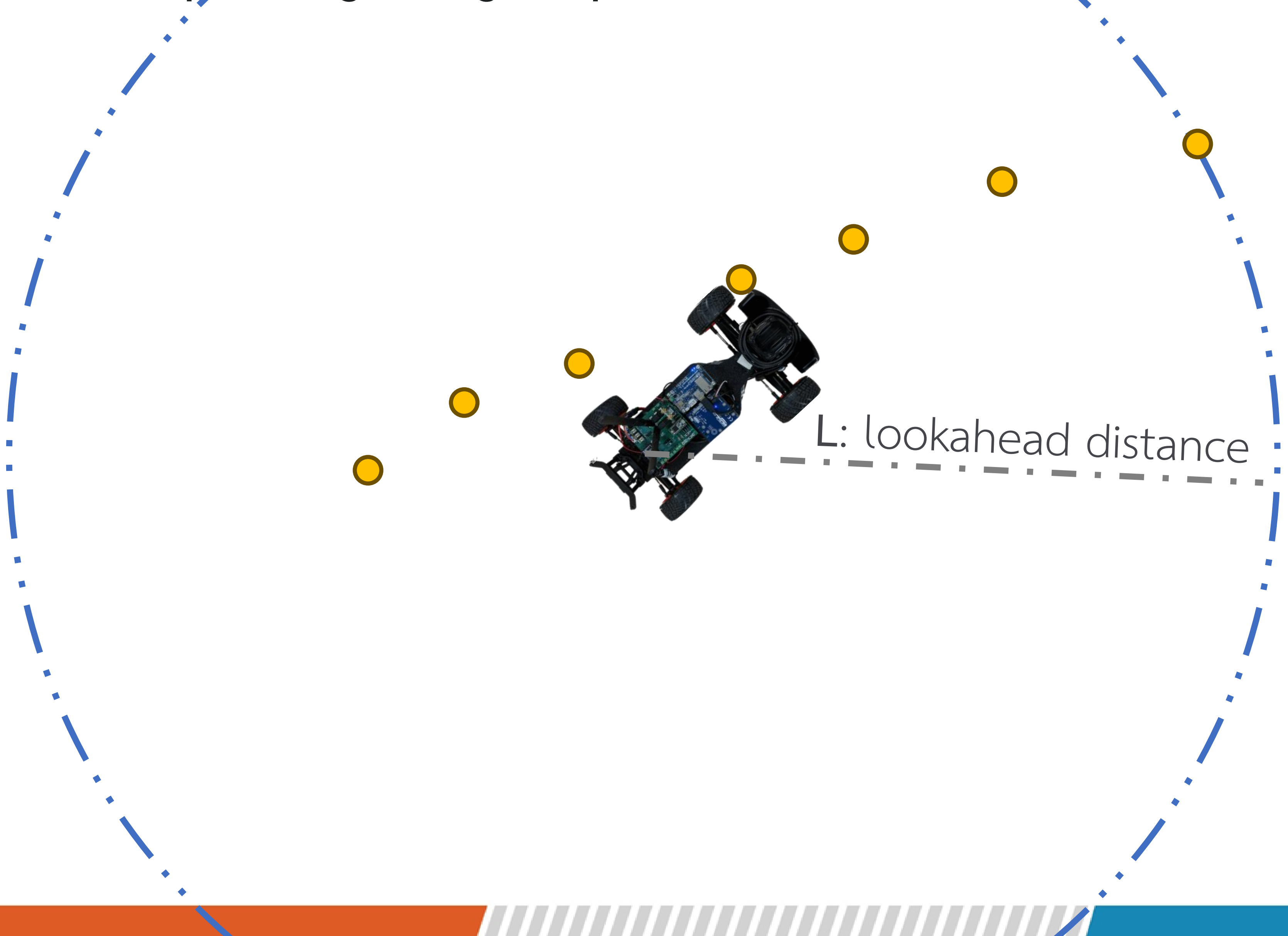
Pure Pursuit : Updating the goal point



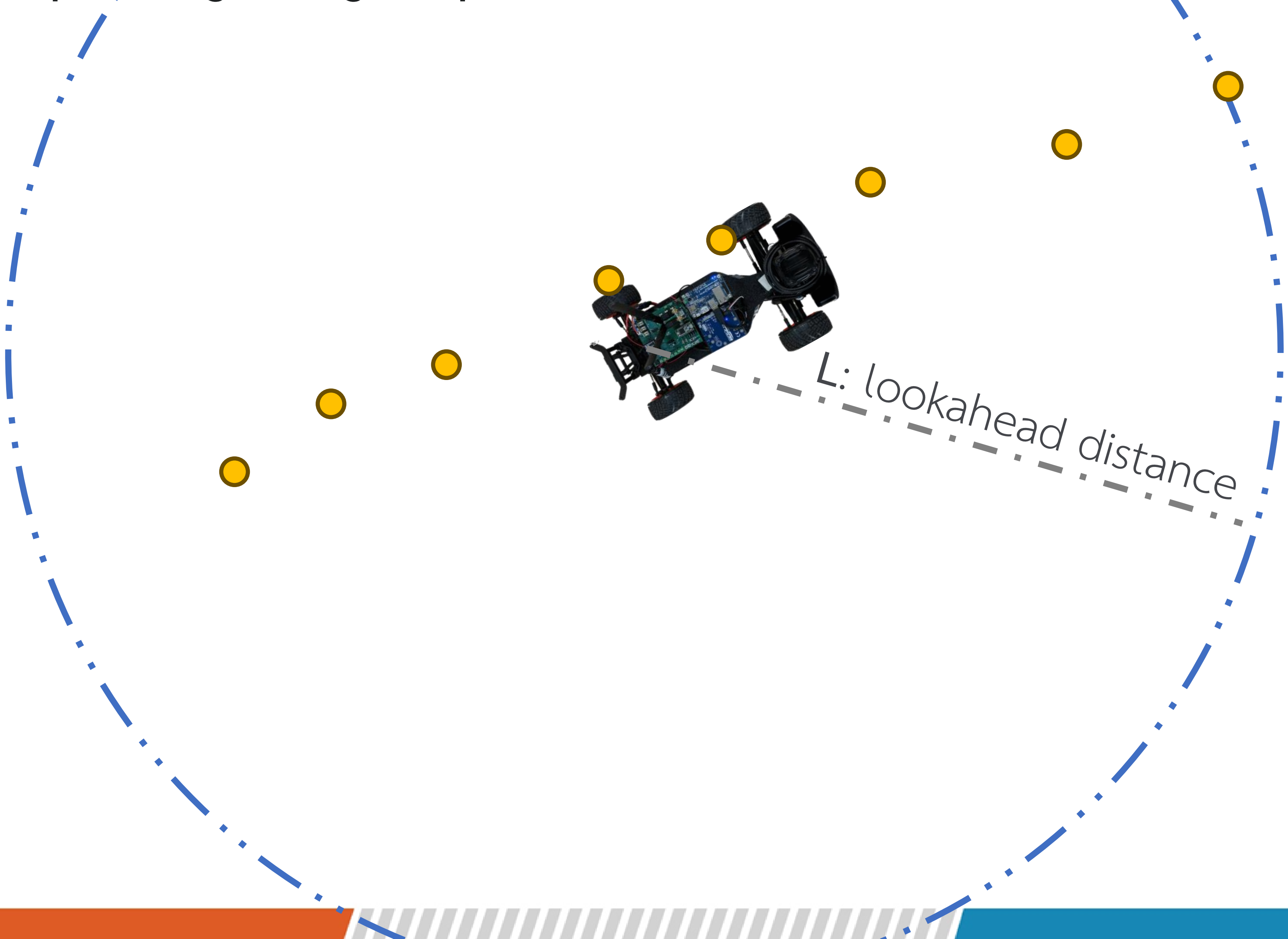
Pure Pursuit : Updating the goal point



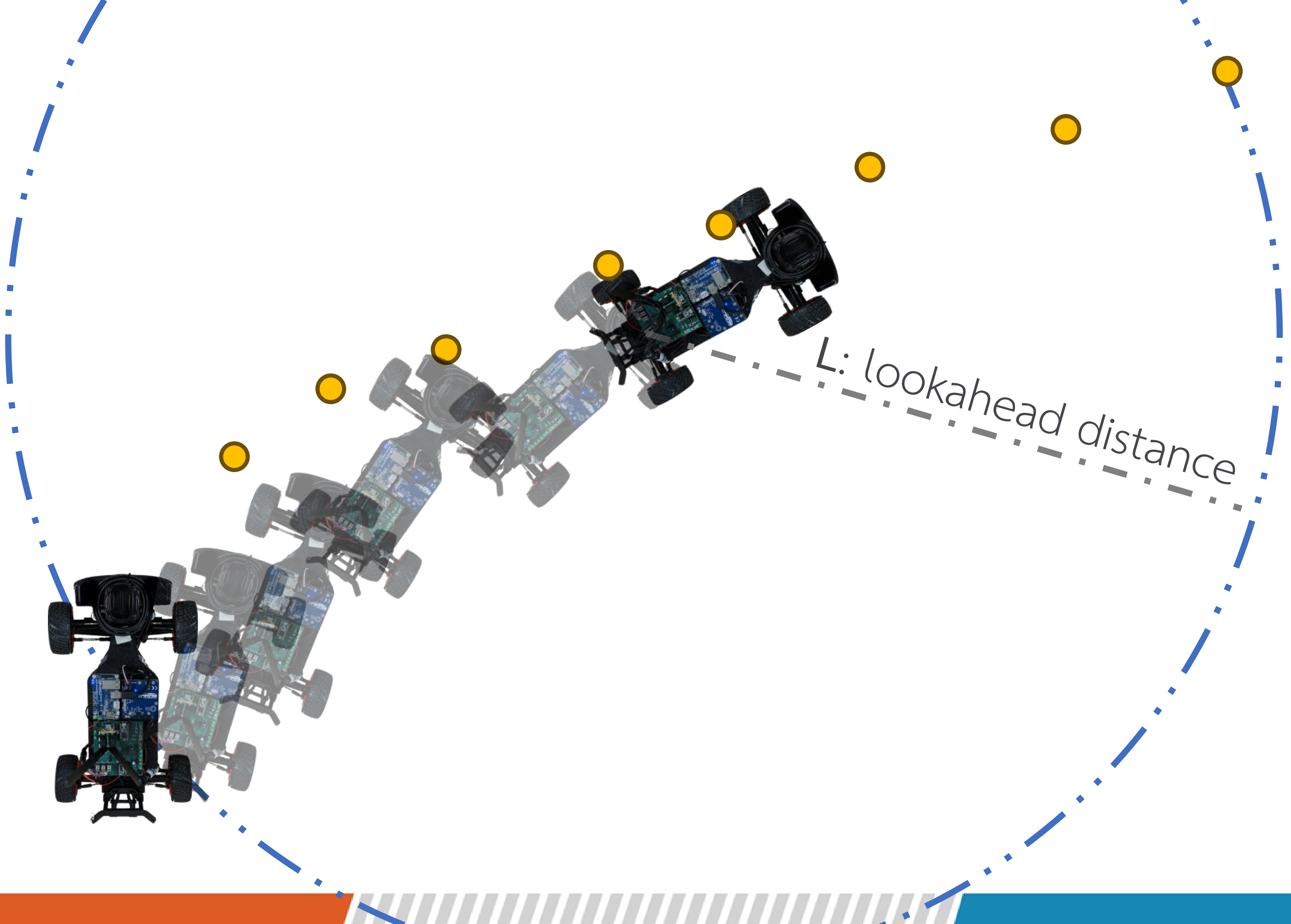
Pure Pursuit : Updating the goal point



Pure Pursuit : Updating the goal point

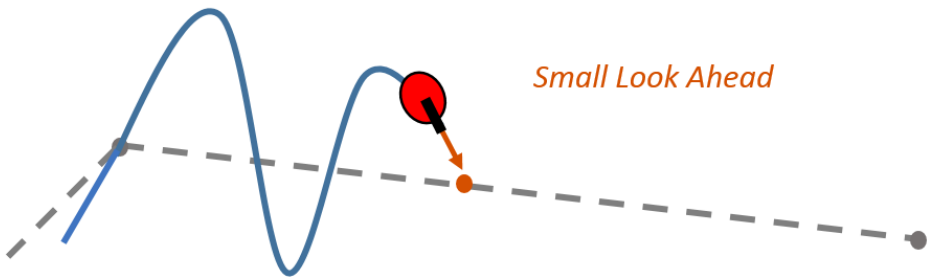


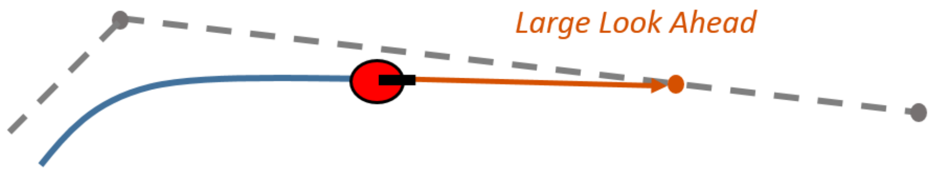
Pure Pursuit : Updating the goal point



Pure Pursuit : Tuning

- The parameter **L** (lookahead distance) is a parameter of pure pursuit.
- Smaller **L** leads to more aggressive maneuvering to track tighter arc, and the tighter arcs might be against dynamical limits of the car.
- Larger **L** leads to smoother trajectory but larger tracking errors, might lead to close calls with obstacles.





Pure Pursuit : Note

- Tuning **L** will change the behavior of pure pursuit the most.
- The waypoints are a sequence of positions, and could also have a velocity component at positions.
- Pure pursuit doesn't take dynamics into account, thus it might produce dynamically infeasible arcs

Attractive / Repulsive Potential Field

The General Idea

- Both the bowl and the spring analogies are ways of storing potential *energy*

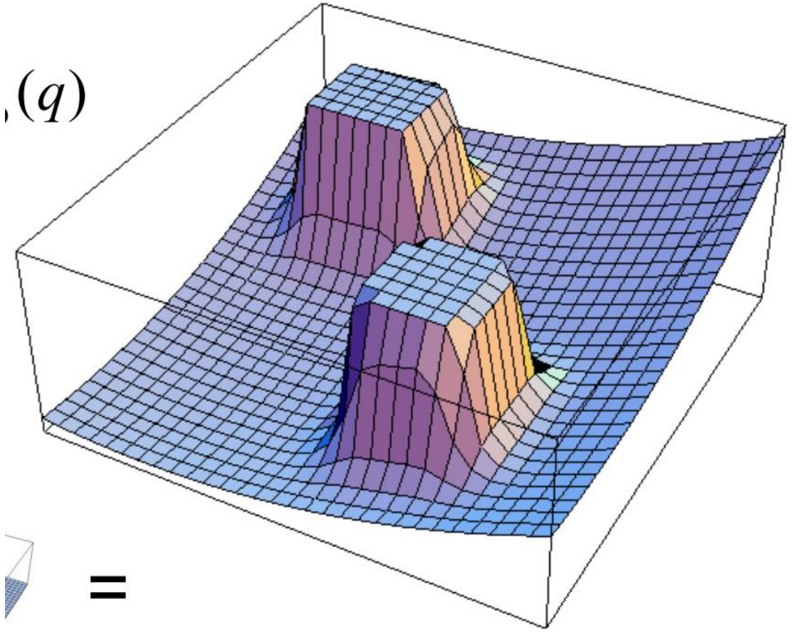
- The robot moves to a lower energy configuration

- A *potential function* is a function $U : \Re^m \rightarrow \Re$

- Energy is minimized by following the negative *gradient* of the potential energy function:

$$\nabla U(q) = DU(q)^T = [\frac{\partial U}{\partial q_1}(q), \ldots, \frac{\partial U}{\partial q_m}(q)]^T$$

- We can now think of a *vector field* over the space of all q's ...
  - at every point in time, the robot looks at the vector at the point and goes in that direction



Attractive / Repulsive Potential Field

$$U(q) = U_{att}(q) + U_{rep}(q)$$

$U_{att}(q)$ is the “attractive” potential --- move to the goal

$U_{rep}(q)$ is the “repulsive” potential --- avoid obstacles

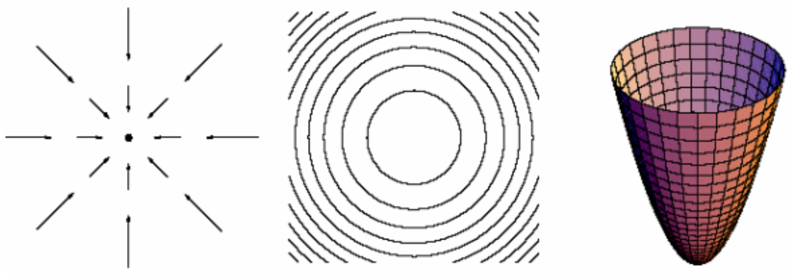
Artificial Potential Field Methods: Attractive Potential

Conical Potential

$$U(q) = \zeta d(q, q_{\text{goal}}).$$

$$\nabla U(q) = \frac{\zeta}{d(q, q_{\text{goal}})}(q - q_{\text{goal}}).$$

Quadratic Potential

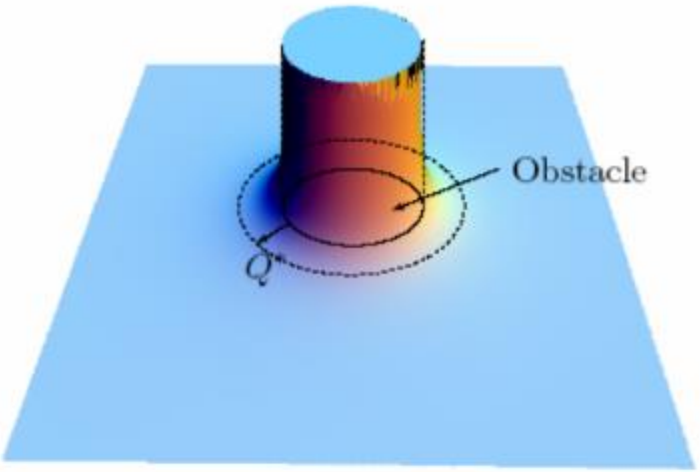


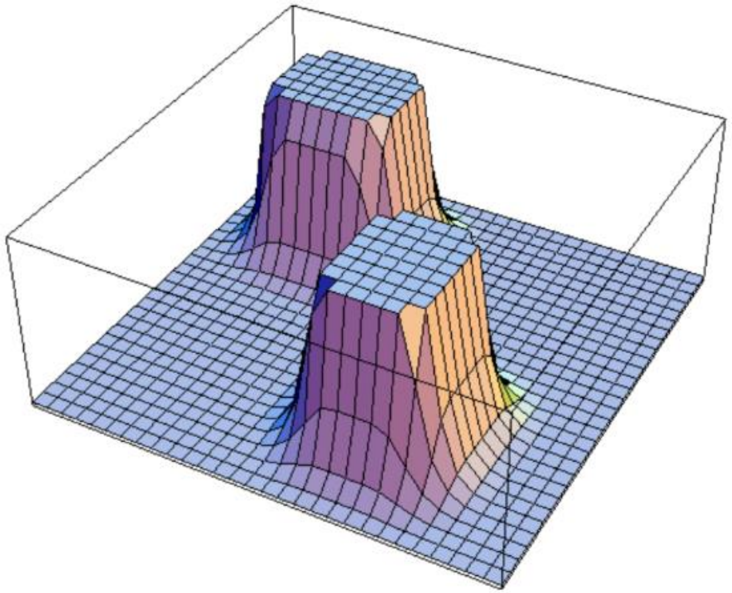
(a) (b) (c)

$$U_{\text{att}}(q) = \frac{1}{2}\zeta d^2(q, q_{\text{goal}}),$$

$$\begin{aligned} F_{\text{att}}(q) = \nabla U_{\text{att}}(q) &= \nabla\left(\frac{1}{2}\zeta d^2(q, q_{\text{goal}})\right), \\ &= \frac{1}{2}\zeta \nabla d^2(q, q_{\text{goal}}), \\ &= \zeta(q - q_{\text{goal}}), \end{aligned}$$

Artificial Potential Field Methods: Repulsive Potential





$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{D(q)} - \frac{1}{Q^*}\right)^2, & D(q) \le Q^*, \\ 0, & D(q) > Q^*, \end{cases}$$

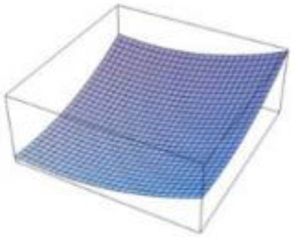
whose gradient is

$$\nabla U_{\text{rep}}(q) = \begin{cases} \eta\left(\frac{1}{Q^*} - \frac{1}{D(q)}\right)\frac{1}{D^2(q)}\nabla D(q), & D(q) \le Q^*, \\ 0, & D(q) > Q^*, \end{cases}$$

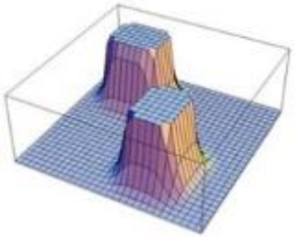
Artificial Potential Field Methods: Total Potential Function

$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q)$$

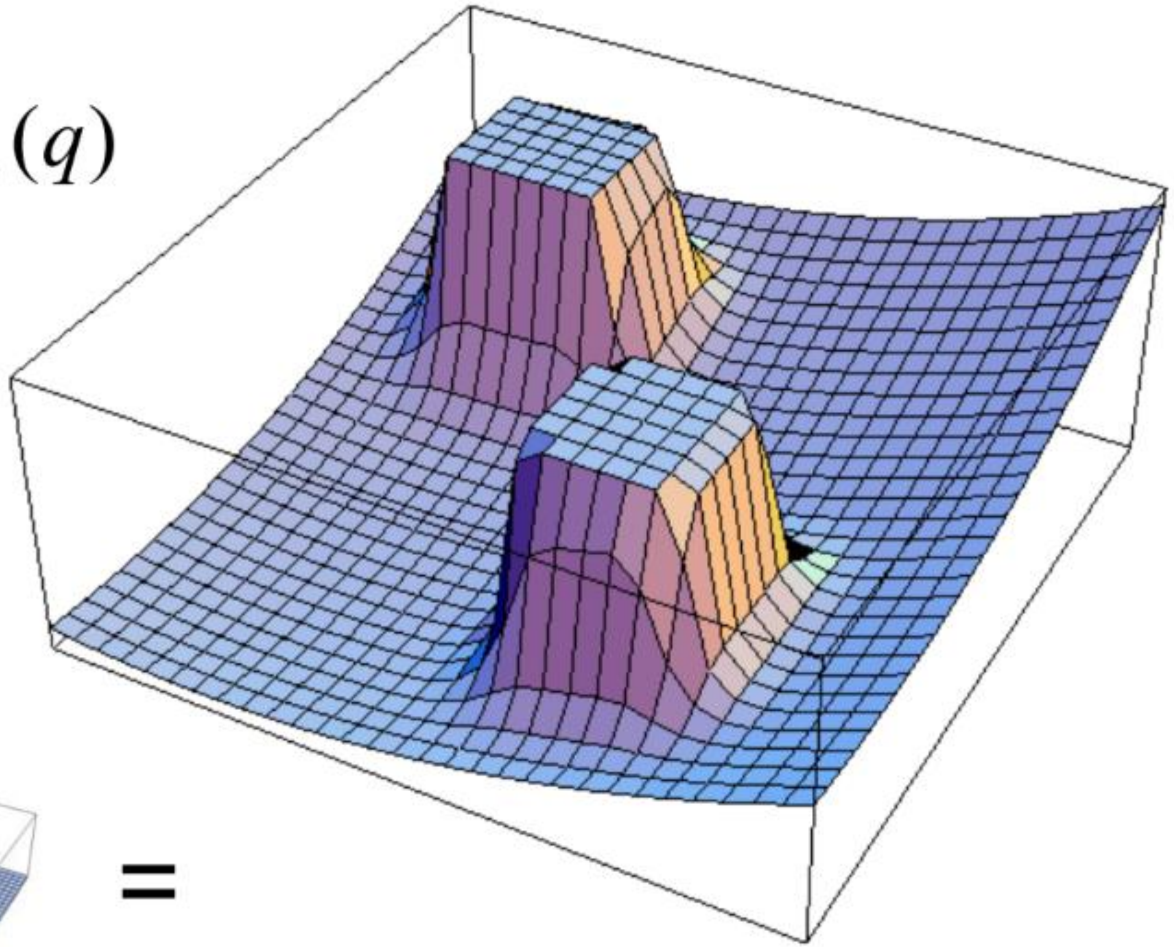
$$F(q) = -\nabla U(q)$$



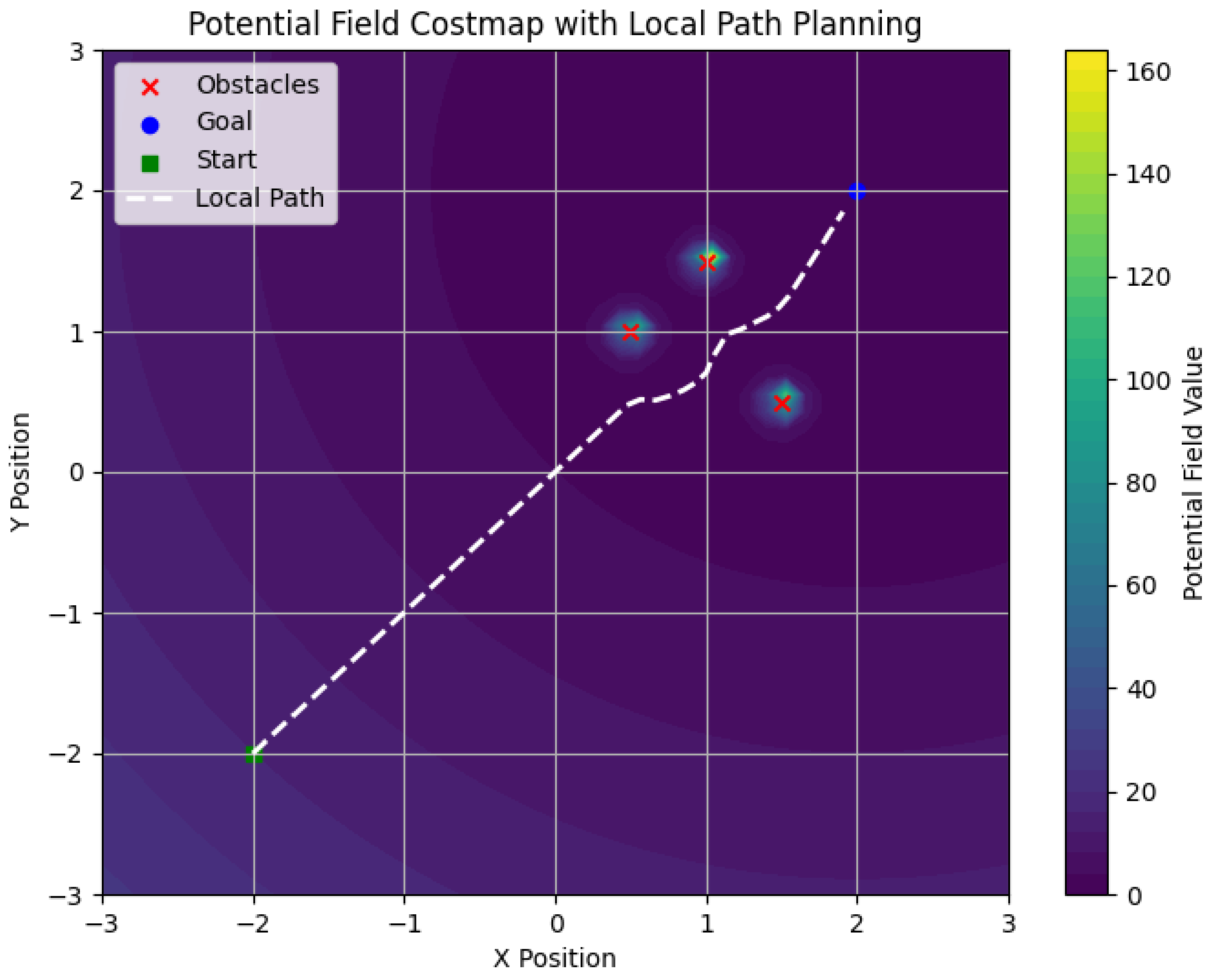
+



=



Artificial Potential Field Methods: Total Potential Function



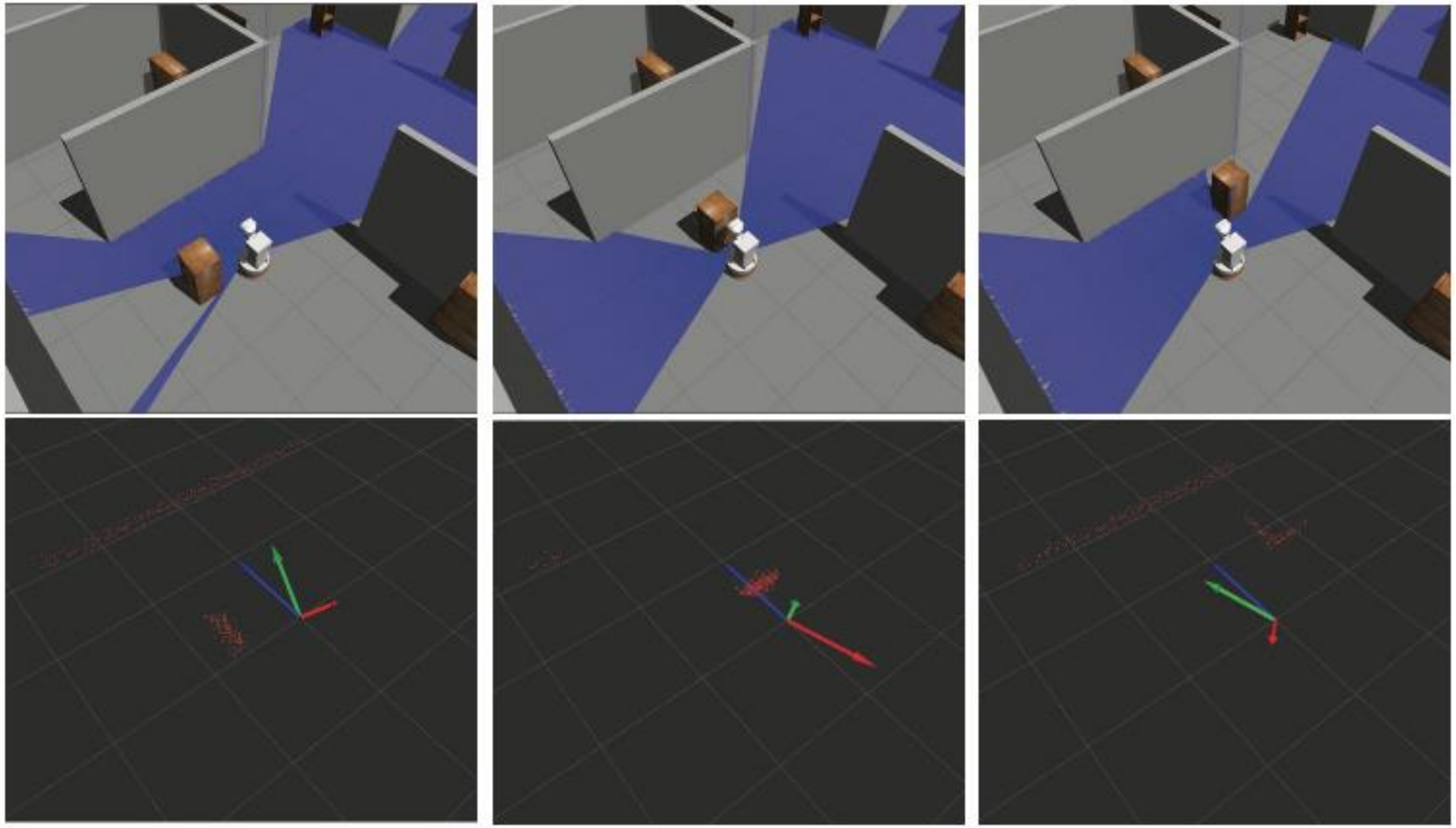
Virtual Force Field

Avoiding Obstacles with VFF

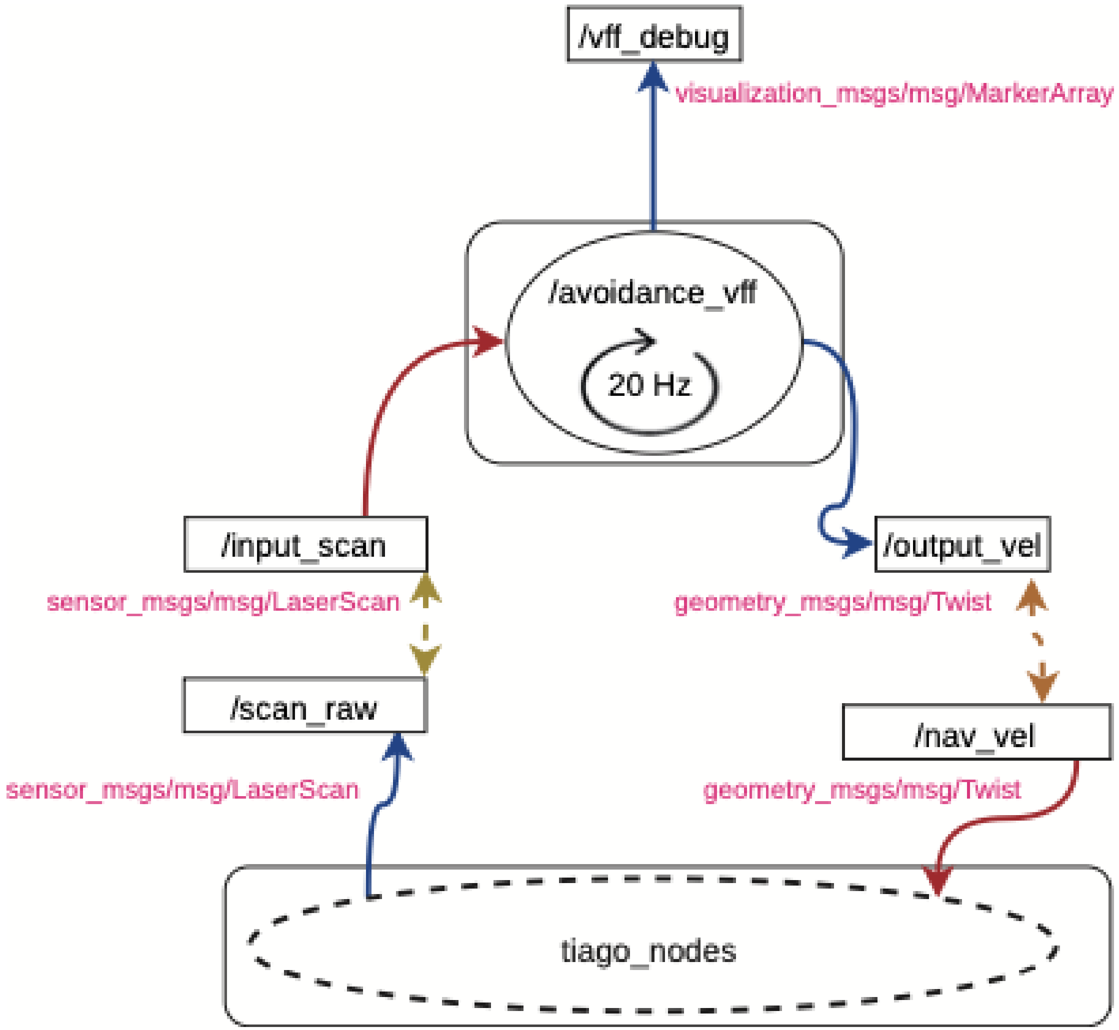
- Use VFF to make the robot go
- forward avoiding obstacles

New concepts:

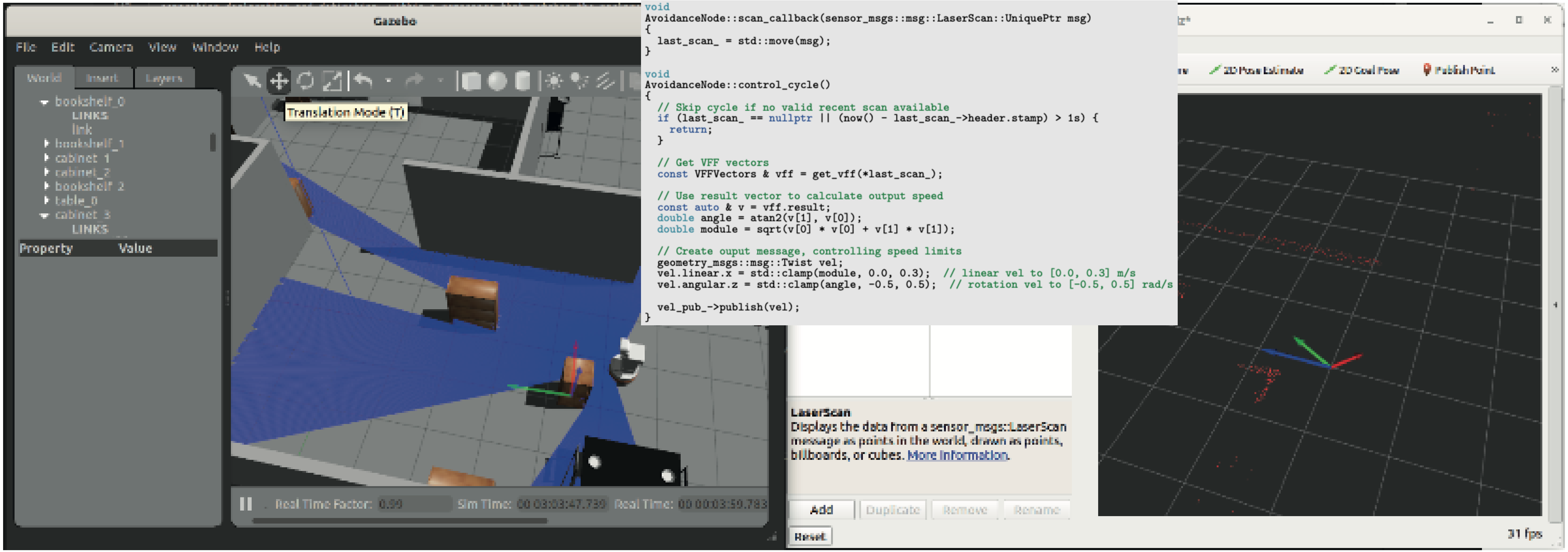
- Laser processing
- Control at joint level
- Testing



Avoiding Obstacles with VFF : The Computation Graph



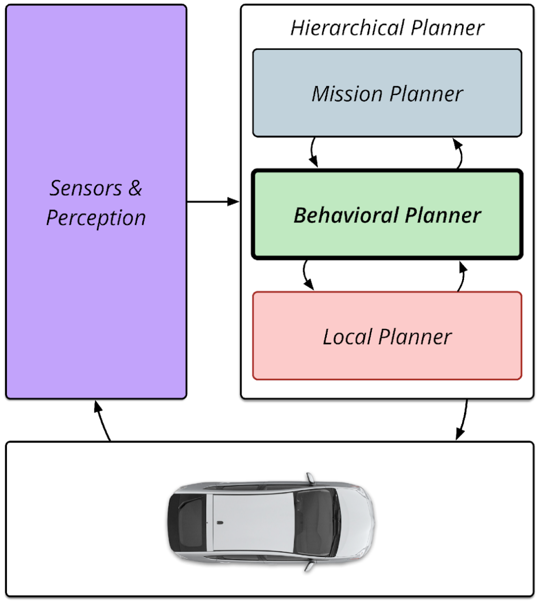
Avoiding Obstacles with VFF : Control Logic

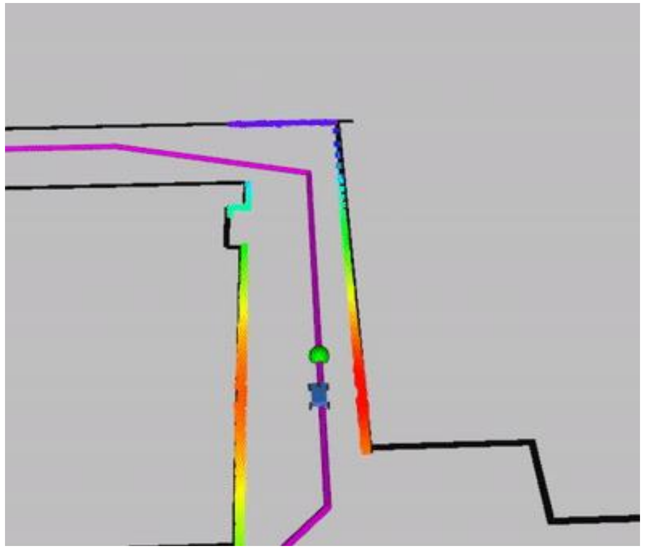


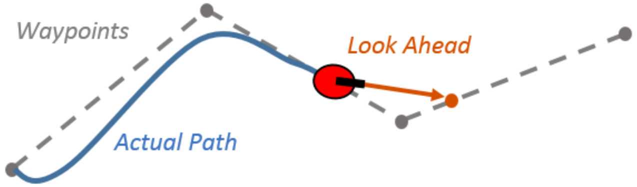

```cpp
void
AvoidanceNode::scan_callback(sensor_msgs::msg::LaserScan::UniquePtr msg)
{
  last_scan_ = std::move(msg);
}

void
AvoidanceNode::control_cycle()
{
  // Skip cycle if no valid recent scan available
  if (last_scan_ == nullptr || (now() - last_scan_->header.stamp) > 1s) {
    return;
  }

  // Get VFF vectors
  const VFFVectors & vff = get_vff(*last_scan_);

  // Use result vector to calculate output speed
  const auto & v = vff.result;
  double angle = atan2(v[1], v[0]);
  double module = sqrt(v[0] * v[0] + v[1] * v[1]);

  // Create ouput message, controlling speed limits
  geometry_msgs::msg::Twist vel;
  vel.linear.x = std::clamp(module, 0.0, 0.3);  // linear vel to [0.0, 0.3] m/s
  vel.angular.z = std::clamp(angle, -0.5, 0.5);  // rotation vel to [-0.5, 0.5] rad/s

  vel_pub_->publish(vel);
}
```

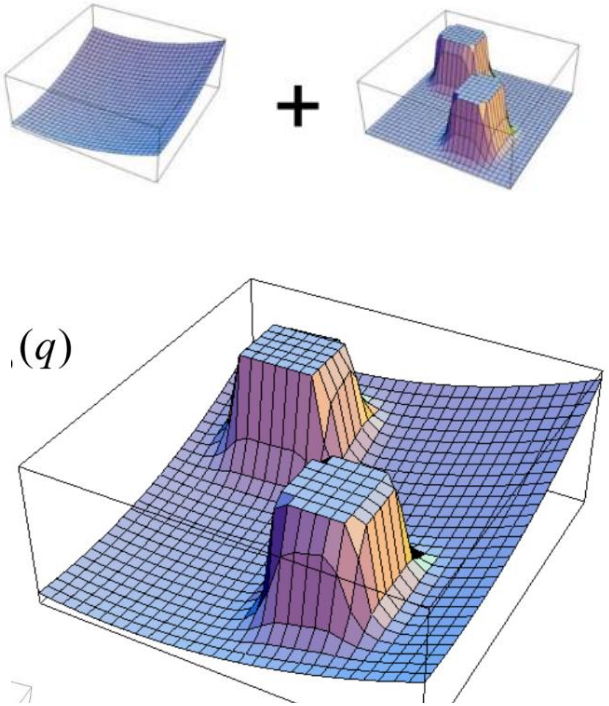
Summary

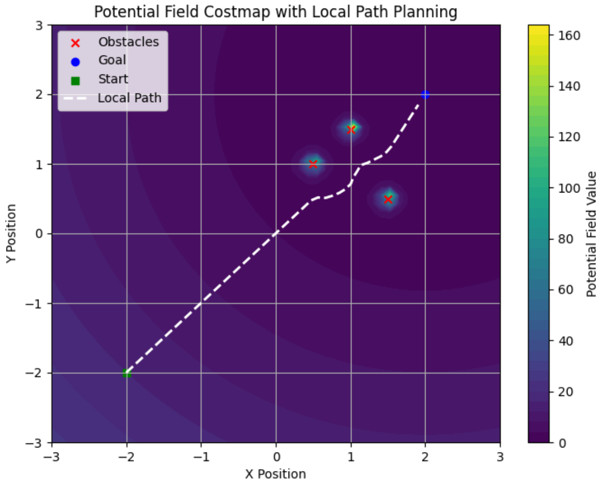
- Mobile Robot Framework
- Pure Pursuit Algorithm
- Potential Field
- Virtual Force Field

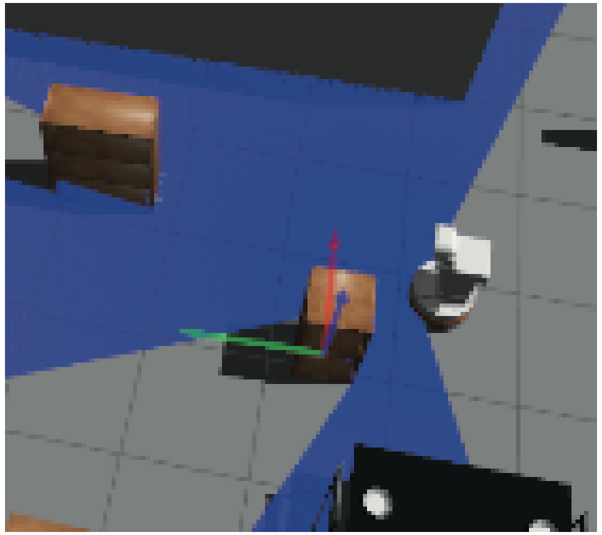


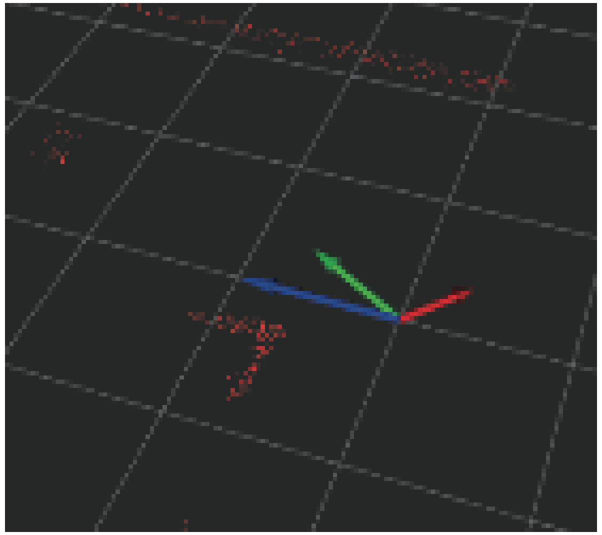












Reference

[1] F1TENTH Autonomous Racing, Pure Pursuit , Hongrui Zheng, Houssam Abbas, Matthew O’Kelly and the F1TENTH Team

[2] A Concise Introduction to Robot Programming with ROS2 - Code Repository

[3] Robotic Motion Planning: Potential Functions, Robotics Institute 16-735, Howie Choset

Q & A

A Cradle of Future Leaders in Robotics