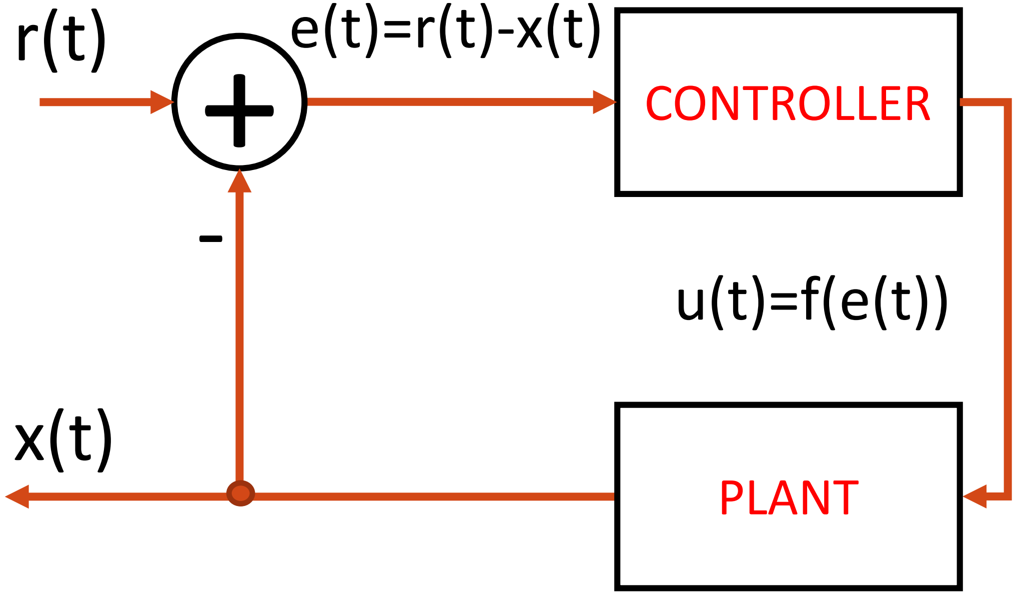
Module B: Reactive Methods

Part 1. Automatic Emergency Braking

Part 2. PID Control and Wall Following

Part 3. Gap-Following Method

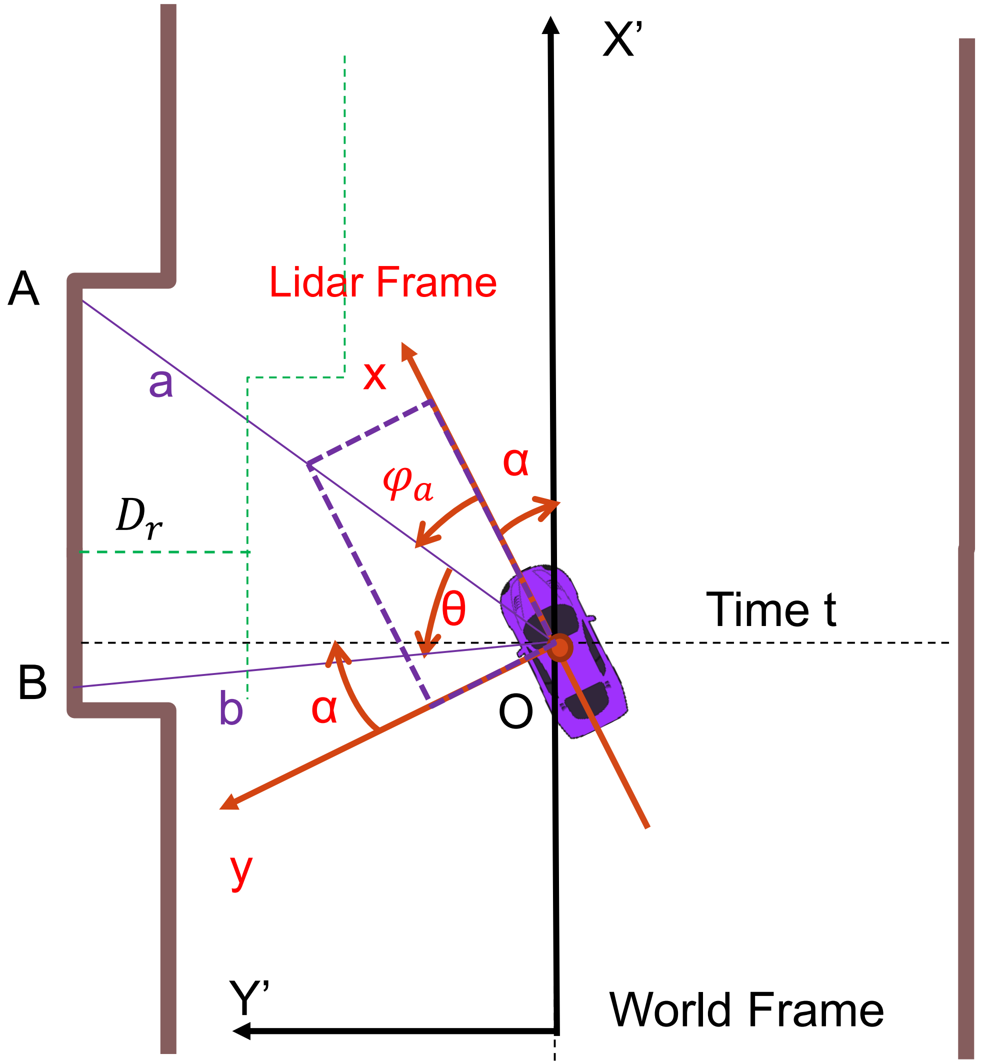


Part B2. Wall Following: coordinate frames

- Coordinate frames
  - The world frame: X'-Y'-Z'
  - The Lidar frame: x-y-z
  - Desired distance between the car and left wall = D_r
  - Assume the car starts at a random pose. The car will view the points A-B as a straight wall.
  - LidarScan ranges: choose two angles at any time t. Range a at angle φ_a Range b at angle φ_b

Note: angles are expressed in degrees here, but shall be in radians in Python. Exact 90° may not be available.

  - Angle difference between ranges a and b is θ;
  - Offset angle between Lidar frame (x-y) and the wall is α (clockwise)
  - What are the good choices of angles for ranges a and b?



Part B2. Following the Left Wall

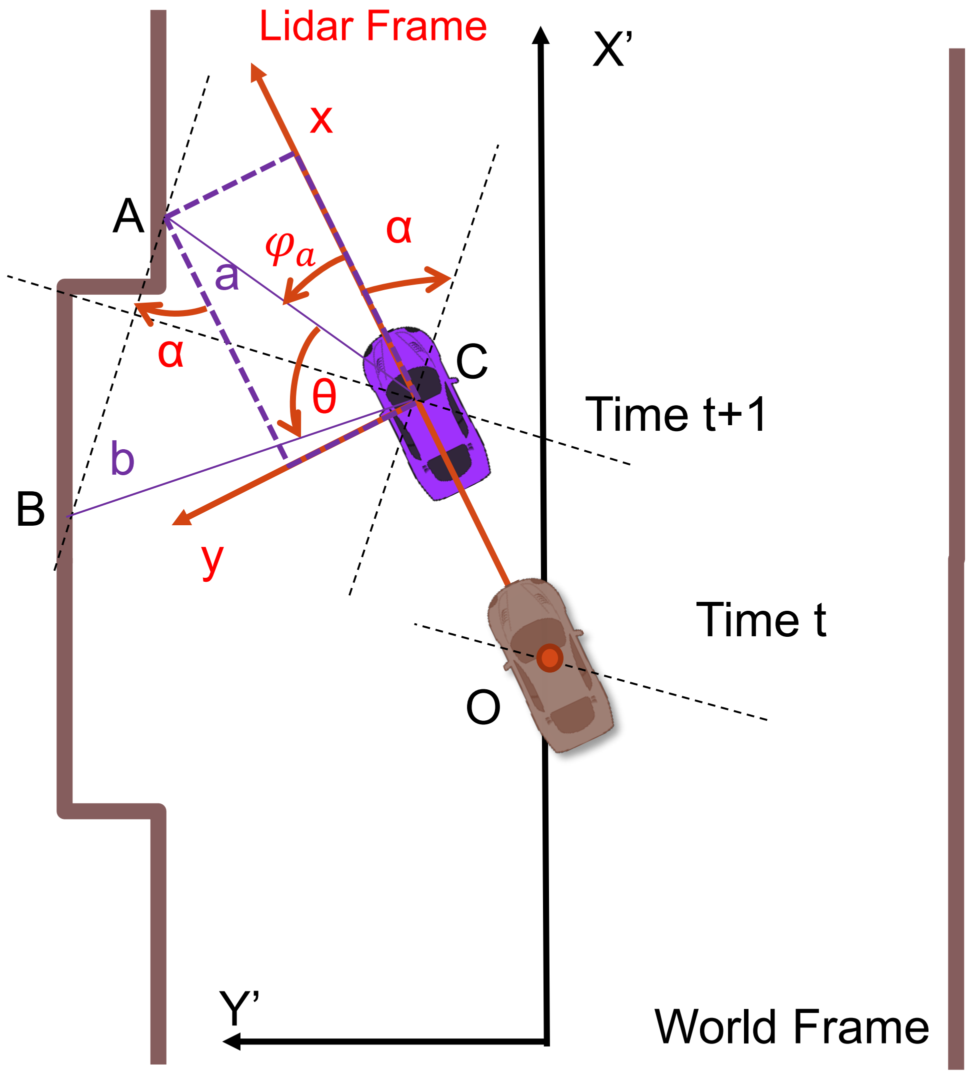
At position C, the new points A and B are the new wall perceived by the car. The offset angle α is the angle between the x axis and the line parallel to AB

To compute α, set range b coincide with the y axis,
then $\cos(\alpha) \propto a\sin(\theta)$,
and $\sin(\alpha) \propto b - a\cos(\theta)$.

Therefore, when $\varphi_b \cong 90°$,

$$\alpha = \tan^{-1}\left(\frac{b - a\cos\theta}{a\sin\theta}\right)$$

This is not to be confused with the steering angle, denoted as $u_\alpha(t)$, which controls the turning angle of the Ackermann steering wheels



Part B2. Following the Left Wall

LEHIGH UNIVERSITY

Offset angle

$$\alpha = \tan^{-1}\left(\frac{b - a\cos\theta}{a\sin\theta}\right) = \tan^{-1}\left(-\frac{a\cos\theta - b}{a\sin\theta}\right)$$

PID error: $e_\alpha \propto e_t = D_r - D_{t+1}$

Insert e_t into PID eqn to compute control input for the steering angle:

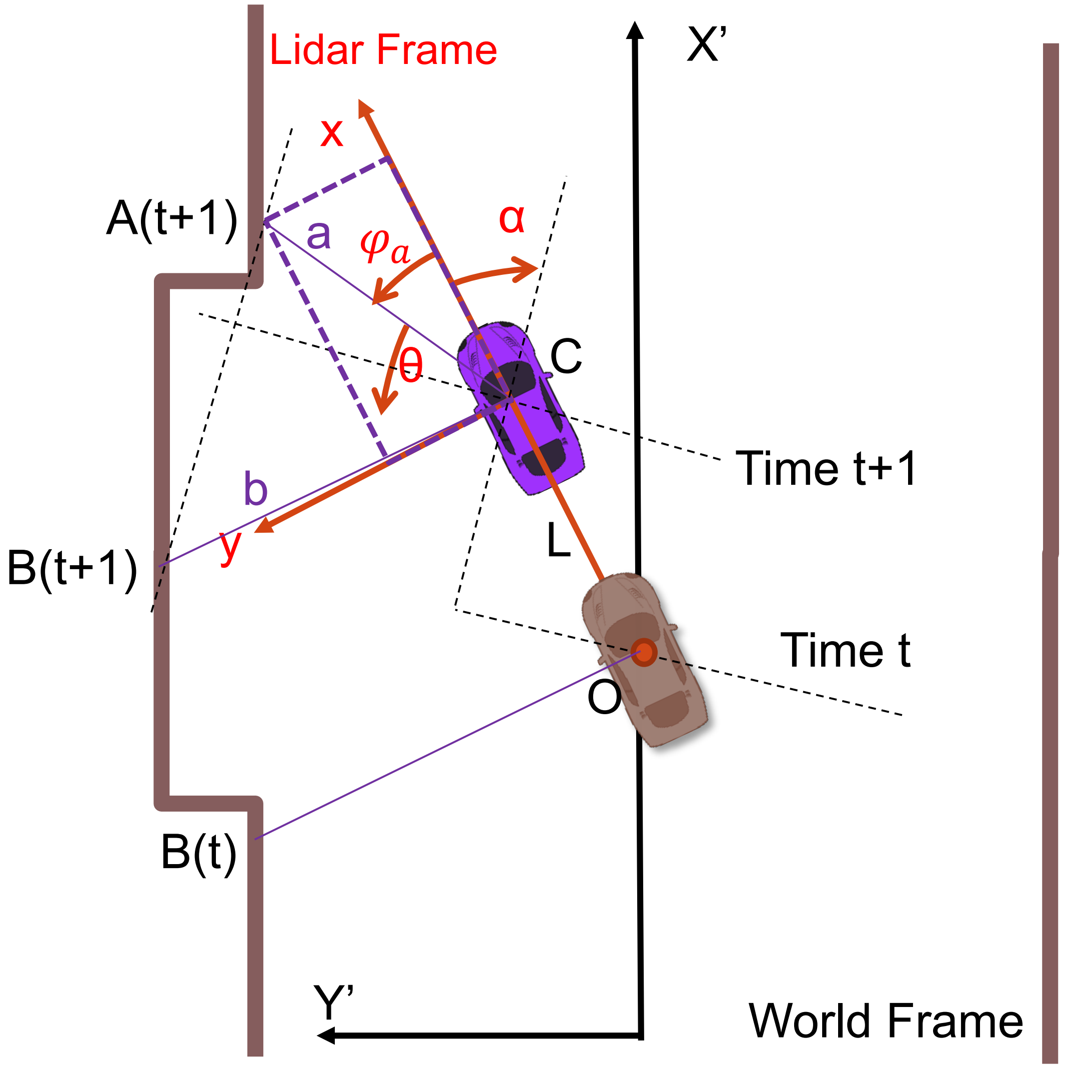
$$u_\alpha(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t)$$

Predicted distance: $D_{t+1} = b\cos\alpha - L\sin\alpha$

Look-ahead distance

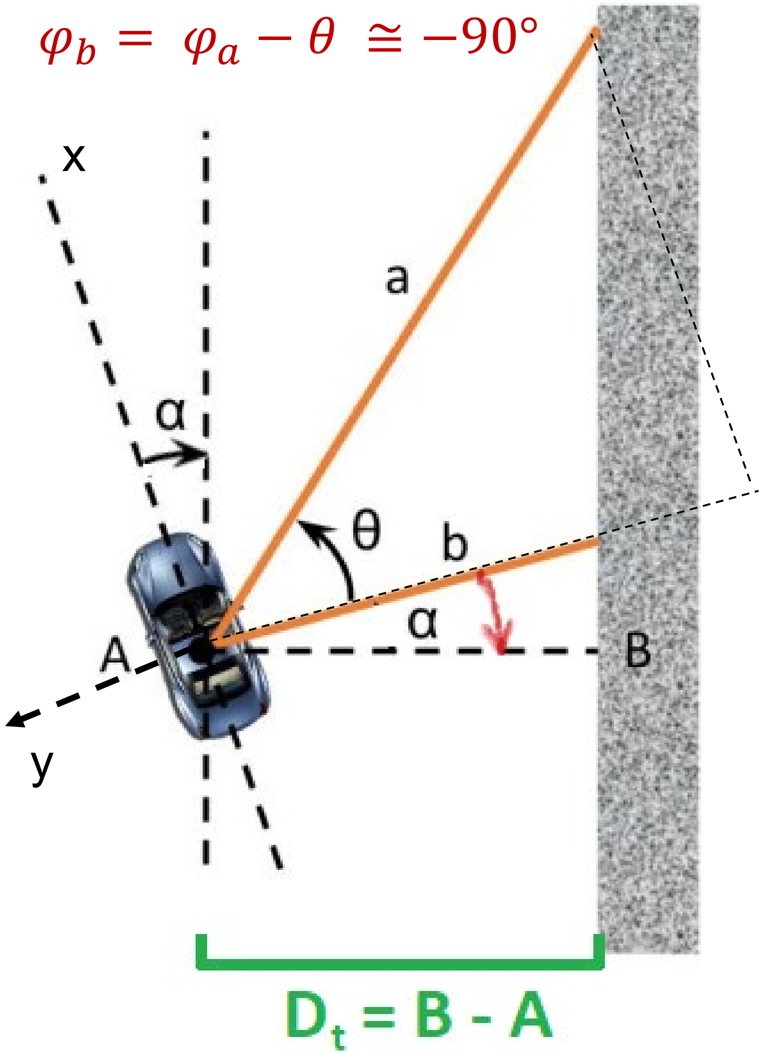
$$L = velocity * time\_increment$$

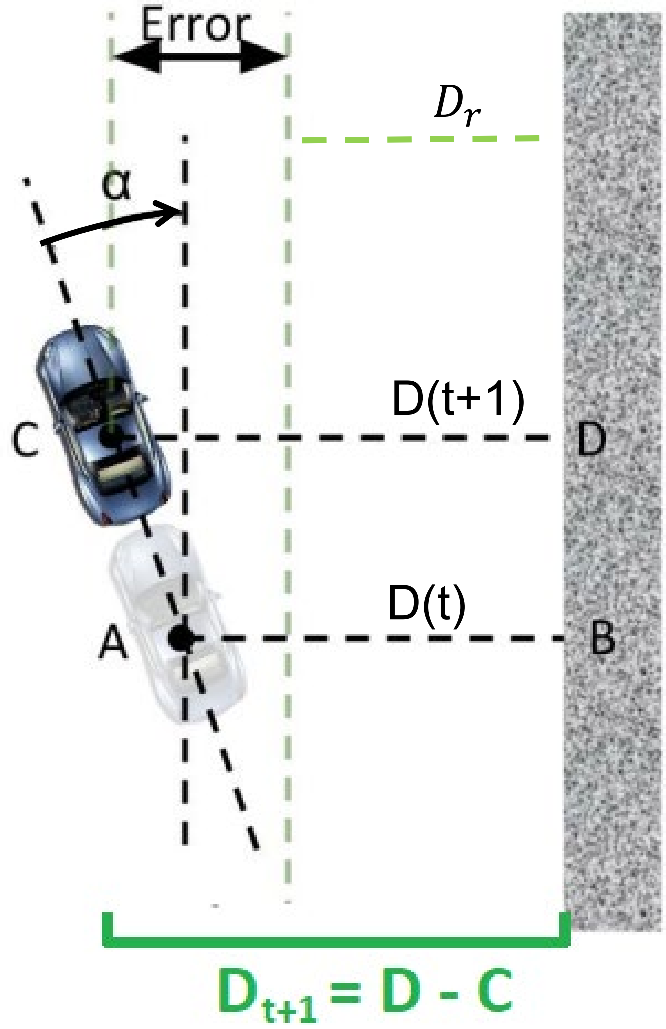
For left-wall following, select a and b such that $\varphi_b = \varphi_a + \theta \cong 90°$ and $0° < \theta < 70°$



Part B2. Following the Right Wall

$\varphi_b = \varphi_a - \theta \cong -90°$





Similarly, offset angle

$$\alpha = \tan^{-1}\left(\frac{a\cos\theta - b}{a\sin\theta}\right)$$

PID error: $e_\alpha \propto e_t = D_r - D_{t+1}$

Predicted distance:

$$D_{t+1} = b\cos\alpha + L\sin\alpha$$

If change α in the left wall-following to $-\alpha$, then both algorithms are the same, except different φ_b, φ_a.

Velocity control

$0° < |u_\alpha(t)| < 10°, v = 1.5\ m/s,$

$10° < |u_\alpha(t)| < 20°, v = 1.0\ m/s,$

$|u_\alpha(t)| > 20°, v = 0.5\ m/s$

Figures are modified from UPenn course CS680 Lab 2 Instructions

Part B2. Wall Following Python

```
class WallFollow:
        def __init__(self):        # Topics & Subs, Pubs
            ……
        def getRange(self, data, angle):  # data: single message from topic /scan
            # angle: lidar scan angles for a or/and b    # Output: return length in meters to object
            return 0.0  #(range at a or/and b)
        def pid_control(self, error, velocity): #Use kp, ki & kd to implement a PID controller
            ……
            drive_msg.drive.steering_angle = angle #Steering_angle u_alpha, not the angles of a or b
            self.drive_pub.publish(drive_msg)
        def followLeft(self, data, leftDist): #Follow left wall  or def followRight(self, data, rightDist): #Follow right wall
            return 0.0 #replace by something important
        def lidar_callback(self, data):  #LidarScan callback, compute error and send to pid_control
            …….              self.pid_control(error, VELOCITY) #initial velocity
```