import numpy as np

import matplotlib.pyplot as plt

import copy

class mode:

def \_\_init\_\_(self, x0=0, y0=0, phi0=0, v0=0, dt=0.1):

self.x = x0

self.y = y0

self.phi = phi0

self.v = v0

self.dt = dt

def update(self, v):

self.v = v

self.x += self.v\*np.cos(self.phi)\*self.dt

self.y += self.v\*np.sin(self.phi)\*self.dt

desired\_ds = 5

class Pid():

def \_\_init\_\_(self, kp, ki, kd, lim=None):

self.kp = kp

self.ki = ki

self.kd = kd

self.lim = lim

self.e1 = 0.0

self.e2 = 0.0

self.dt = 0.1

def set\_reference(self, r):

self.r = r

def get\_u(self, y):

e = y - self.r

P = self.kp\*(e)/self.dt

I = self.ki\*(e + self.e2)

D = self.kd\*(e - self.e1)/(self.dt\*\*2)

self.e2 += e

self.e1 = copy.deepcopy(e)

PID = P+I+D

return PID

class Pid():

def \_\_init\_\_(self, kp, ki, kd, lim=None):

self.kp = kp

self.ki = ki

self.kd = kd

self.lim = lim

self.e1 = 0.0

self.e2 = 0.0

self.dt = 0.1

self.ut = 0.0

def set\_reference(self, r):

self.r = r

def get\_u(self, y):

e = y - self.r

P = self.kp\*(e - self.e1)/self.dt

I = self.ki\*e

D = self.kd\*(e + self.e2 - 2\*self.e1)/(self.dt\*\*2)

self.e2 = copy.deepcopy(self.e1)

self.e1 = e

PID = P+I+D

self.ut += PID

return self.ut

# def PID\_v(vs, ds):

# return 10\*(ds-desired\_ds)

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

%matplotlib inline

# set up matplotlib

is\_ipython = 'inline' in matplotlib.get\_backend()

if is\_ipython:

from IPython import display

plt.ion()

plt.figure(figsize=(18, 3))

ugv0 = mode(x0=0, y0=0, phi0=0, v0=0, dt=0.1)

ugv1 = mode(x0=10, y0=0, phi0=0, v0=0, dt=0.1)

# set reference trajectory

refer\_path = np.zeros((100, 2))

refer\_path[:,0] = np.linspace(0, 100, 100)

plt.plot(refer\_path[:,0], refer\_path[:,1], '-.b', linewidth=5.0)

ugv\_pid = Pid(kp=1, ki=0.1, kd=0)

ugv\_pid.set\_reference(desired\_ds)

for i in range(1000):

ds = ugv1.x - ugv0.x

v0 = min(ugv\_pid.get\_u(ds), 10)

# v0 = min(PID\_v(ugv0.v, ds), 10)

v1 = abs(5\*np.sin(0.1\*i/(2\*np.pi)))

ugv1.update(v1)

ugv0.update(v0)

plt.subplot(211)

plt.scatter(ugv0.x, ugv0.y, color='r')

plt.scatter(ugv1.x, ugv1.y, color='g')

plt.axis([0, 100, -3, 3])

plt.subplot(212)

plt.scatter(i, ds)

plt.axis([0, 1000, 0, 10])

if is\_ipython:

display.clear\_output(wait=True)

display.display(plt.gcf())