WORKSHEET

U10.8 Time scaling a path

Following a trajectory · University · about 40 min

Name _____ Class _____ Date _____

What this lesson is about

Cubic and quintic time laws and what smoothness costs, the speed limit that depends on direction (this robot is fastest on the diagonal), and the time-optimal speed along a path.

- A time law as a function
- What the robot makes of them
- The speed limit depends on direction
- Time-optimal along a path
- Driving the plan

The task: dash to the corner

The far corner of the mat is about two metres away, on the diagonal. Be in it by 8.8 seconds and stay there until the clock passes 11, then let the program end; the run is cut off at 12. Facing the corner and driving flat out gets there after about 11 seconds on this robot, which is too slow. Use the corner of the velocity box instead: work out which way to face so that the robot's fastest direction points at the corner, turn to face it, and go. Even a perfect aim will not stay perfect, because the vibration drive turns the robot slowly as it goes, and leaks a little of its forward push sideways. So steer as you drive: compare the direction the robot is actually moving with the direction to the target, and turn a little to close the gap. Once it steers, the steering also soaks up a few degrees of error in the aim, so the data sheet's speeds will do nearly as well as your own. They matter if you drive open loop, without steering: they put the corner of the box at 37 degrees rather than this robot's 41, and a few degrees of aim is 10 to 15 cm at two metres. The figure shows the two runs against the clock.

```
from bugbot import *
import math
connect()

V_FWD, V_SIDE = 20.0, 15.0                  # replace with the speeds you measured
START, TARGET = (30.0, 30.0), (170.0, 170.0)

def here():
    px, py = position()
    return START[0] + px, START[1] + py
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/u10-8-time-scaling-a-path/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Add the last cell's acceleration limit to the dash, so the robot follows a trapezoid along the diagonal instead of going flat out at once. How much time does 15 cm/s² cost, and does it still make 8.8 s?
2. Instead of turning, hold the heading at zero and mix forward and sideways so that the robot moves at 45 degrees, straight at the corner. That is the single-heading idea of the third cell: no turn, but the direction is off the corner of the box, so the sideways axis runs out first. What is the top speed that way, and does it make 8.8 s as reliably as turning does?
3. Drive the whole route from the third cell with the heading held at zero, following the time-optimal speed leg by leg as the last cell did for one. Does the time the robot takes match the prediction?

4. On the third cell's route, turning to a fast diagonal at every corner only tied with never turning, and one heading chosen once beat them both. Make the legs longer, or the acceleration limit higher, in the third cell: at what point does turning at every corner start to beat choosing one heading once?