WORKSHEET

U2.6 Go to a point

Kinematics and frames · University · about 25 min

Name _____ Class _____ Date _____

What this lesson is about

The first useful controller: a vector to the target, straight through the inverse kinematics.

- What is going on
- Where the position came from
- Stopping

Questions

6 marks in all

1. The go-to-point controller sets its speed from the gap to the target. What does this print? [1 mark]

```
for gap in (60.0, 28.0, 10.0, 3.0):
    speed = min(14.0, max(4.0, 0.5 * gap))
    print(gap, speed, "stop" if gap < 4 else "drive")
```

2. The target is 30 cm east and 40 cm north of the robot. What does this print? [1 mark]

```
import math
dx, dy = 30.0, 40.0
gap = math.hypot(dx, dy)
speed = min(14.0, max(4.0, 0.5 * gap))
wx, wy = speed * dx / gap, speed * dy / gap
print(gap, speed, round(wx, 1), round(wy, 1))
```

3. A student writes wx, wy = speed * dx, speed * dy, leaving out the division by gap. What goes wrong? [1 mark]

- ☐ **A** The effective gain depends on how far away the target is, so the robot behaves differently on long and short runs
- ☐ **B** The robot heads in the wrong direction
- ☐ **C** Nothing, because speed is already capped at 14
- ☐ **D** The robot can never get within 4 cm, because the velocity is always zero

4. How should the stopping threshold, gap < 4, be chosen? [1 mark]

- ☐ **A** Larger than the distance the robot coasts from the speed it arrives at, or it sails past and hunts
- ☐ **B** As small as possible, since a smaller threshold is always more accurate
- ☐ **C** Equal to the robot's radius
- ☐ **D** Smaller than the noise on position(), so noise cannot trigger a stop

5. The robot starts at world (60, 60), and position() counts from where it started. How far is a target at world (100, 45) from the start, in cm to one decimal place? [1 mark]

6. This controller reads position(), which U1.3 called the lab's overhead camera. Why is that acceptable here? [1 mark]

- ☐ **A** The subject is kinematics for now, and from U6 an estimate the robot works out for itself replaces it under the same loop
- ☐ **B** position() is what a real robot would use for this controller
- ☐ **C** The inverse kinematics only works with a perfect position
- ☐ **D** odometry() is in the body frame, so it cannot be used with world targets

The task: go to a point

Drive to world `(140, 150)` and stop within 10 cm of it. The robot starts at `(60, 60)` facing the wrong way, and `position()` counts from where it started.

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

V_MAX, V_LAT = 20.0, 15.0
TX, TY = 80.0, 90.0        # the target, in cm from the start
```

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

Do it on the robot

www.bugbotlab.com/learn/u2-6-go-to-a-point/

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

Challenges

1. Add a heading term so the robot also finishes facing a chosen direction, without changing the path.
2. Feed it a list of points and drive them in order, without stopping at each one.
3. Replace `position()` with `odometry()` and run it again. How close does it get now, and how much worse does it get on a longer run?