WORKSHEET

U11.4 A tag as a fix

Vision · University · about 40 min

Name _____ Class _____ Date _____

What this lesson is about

Range and bearing to a landmark you know the position of, which is a position fix.

- One tag, if you know your heading
- Two tags, if you do not
- Where a tag's pose estimate is weak
- Fixing and driving
- A fix is not a filter

Questions

6 marks in all

1. Tag 20 is at (40, 160). Facing heading 0, the robot sees it at a bearing of -20 degrees and a range of 50 cm. What does this print? [1 mark]

```
import math
Tx, Ty = 40.0, 160.0
h, b, d = 0.0, -20.0, 50.0
x = Tx - d * math.sin(math.radians(h + b))
y = Ty - d * math.cos(math.radians(h + b))
print(round(x, 1), round(y, 1))
```

2. A one tag fix is taken at a range of 100 cm with a heading that is 1 degree wrong. How far sideways does the fix move, in cm to 2 decimal places? [1 mark]

3. Why do two tags make the heading observable when one tag does not? [1 mark]

- ☐ **A** Two tags give four measurements, two ranges and two bearings, for three unknowns x, y and h
- ☐ **B** Two tags always form a right angle with the robot
- ☐ **C** The second tag's bearing is measured in the mat frame
- ☐ **D** Averaging two ranges removes the heading error

4. Put the parts of an AprilTag pose estimate in order from most to least reliable. [1 mark]

Number the lines 1 to 3 to put them in the right order.

___ `Range, from apparent size`

___ `Out-of-plane rotation, from the corner geometry`

___ `Bearing, from the centroid`

5. A robot steering on a tag's reported yaw sees the estimate jump by 15 degrees between frames while nothing moves. What is the best explanation? [1 mark]

- ☐ **A** Near square on, tilting the tag a few degrees either way gives almost the same image, so the solver flips between two poses
- ☐ **B** The bearing is noisy because the centroid is quantised
- ☐ **C** The camera's focal length is wrong
- ☐ **D** The tag is too close for its range to be measured

6. Why is a tag fix not a substitute for a filter? [1 mark]

- ☐ **A** A fix exists only while a tag is in view, so it has to be combined with dead reckoning, weighted by how much each is worth
- ☐ **B** A fix is always less accurate than odometry
- ☐ **C** A filter cannot use range and bearing measurements
- ☐ **D** A fix gives heading but never position

The task: fix your position from two tags

Tag 20 is at (40, 160) on the mat and tag 21 at (150, 140). Work out where the robot is, print it as `my x:` and `my y:`, and then drive to (130, 100) on the mat and stop there, refixing as you go. Neither `position()` nor `heading()` is allowed: both are the lab's truth, and the point is that two tags are enough.

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

F = 92.4
TAGS = {20: (40.0, 160.0), 21: (150.0, 140.0)}
TX, TY = 130.0, 100.0
set_cv("apriltag")
wait(0.3)
```

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

Do it on the robot

www.bugbotlab.com/learn/u11-4-a-tag-as-a-fix/

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

Challenges

1. Start the iteration from a heading guess of 90 degrees. Where does it end up, and what check would catch that answer before the robot drove on it?
2. Use only tag 20 and hold the heading at whatever your two-tag solve last said. How far has the fix drifted by the end of the run?
3. Corrupt one range by 20 percent before solving. Which of `x`, `y` and `h` suffers most, and does that match what the lesson said about where the strength lies?