ANSWERS

U11.3 Range from apparent size

Vision · University · about 35 min

What this lesson is about

A known object gives a range, and the range gets worse as the square of itself.

- The relation
- How bad it gets, and how fast
- Detection versus measurement
- Where this is the right tool anyway

Questions

6 marks in all

1. An 8 cm ball appears 20 pixels wide on a camera with f = 92.4 px. How far away is it, in cm to 1 decimal place? [1 mark]

> **Answer:** 37.0 (accept within 0.1). d = f S / w = 92.4 x 8 / 20 = 36.96 cm.

2. How many centimetres of range is one pixel of width worth for the 8 cm ball at 50 cm and at 100 cm? [1 mark]

```
F, S = 92.4, 8.0
for d in (50, 100):
    print(d, round(d * d / (F * S), 1))
```

> **Answer:**
>
> ```
> 50 3.4
> 100 13.5
> ```
>
> One pixel is worth d^2 / (f S): 2500 / 739 = 3.4 cm at 50 cm and 10000 / 739 = 13.5 cm at 100 cm. Doubling the range quadruples the error.

3. Why does range from apparent size get so much worse with distance? [1 mark]

- ☐ **A** The apparent width falls as 1 / d, so one pixel is a larger fraction of it, and the range error grows as d squared
- ☐ **B** The detector adds more noise to far objects
- ☐ **C** The lens distorts distant objects
- ☐ **D** Far objects are nearer the edge of the frame

> **Answer:** A. dd/dw = -d^2 / (f S). Nothing about the algorithm changes with range; the geometry does.

4. Beyond what range is one pixel of width worth more than 10 percent of the range itself, for the 8 cm ball and f = 92.4? Give the range in cm to 1 decimal place. [1 mark]

> **Answer:** 73.9 (accept within 0.2). d^2 / (f S) > 0.1 d when d > 0.1 f S = 0.1 x 92.4 x 8 = 73.9 cm.

5. A flat tag 40 cm away is seen at 60 degrees rather than square on, and its range is worked out from the width of its bounding box. What range is reported, in cm? [1 mark]

> **Answer:** 80 (accept within 0.5). At 60 degrees the tag's width is cos 60 = half its true width, so d = f S / w reads double: 80 cm.

6. The box width and the area of the far ball give ranges about 3 cm apart (123.2 and 126.1 cm), and repeated frames give the same numbers. Which statements are right? [1 mark]

Tick every answer that is true.

- ☐ **A** This is quantisation of the box edges, not noise
- ☐ **B** Averaging repeated frames will not remove it
- ☐ **C** The area is a sum over many pixels, so it is quantised far more finely than an edge
- ☐ **D** Averaging a few dozen frames will remove the disagreement
- ☐ **E** The pinhole model does not apply to a far ball

> **Answer:** A, B, C. A box edge is an integer, and the same error comes back every frame. Area is a finer measurement of the same size.

The task: range from apparent size

Two red balls are on the mat, both 8 cm across. Standing still, print `near:` and `far:`, the range to each in centimetres worked out from how wide it looks, and `per pixel:`, how many centimetres of range one pixel of width is worth at the further ball.

```
from bugbot import *
connect()

F = 92.4
R = 4.0
set_cv("blob", "red")
wait(0.3)
```

The hint students can ask for: Both balls are 8 cm across. A sphere of radius r at range d covers 2*r*f/d pixels, with f the focal length you measured in U11.2, so the range falls out of the width of the detector's box. Then differentiate that relation with respect to the width: the sensitivity is the thing worth printing, because it is what tells you when the measurement has stopped being a measurement.

A solution

```
from bugbot import *
connect()

F = 92.4
R = 4.0                       # the balls are 8 cm across

set_cv("blob", "red")
wait(0.3)
seen = blobs()

ranges = []
for cx, cy, area, x0, y0, x1, y1, aspect in seen:
    w = x1 - x0
    ranges.append(2 * R * F / w)

print("near:", round(ranges[0], 1))
print("far:", round(ranges[1], 1))

# d = 2*R*F/w, so dd/dw = -d*d / (2*R*F): one pixel is worth this many cm out there
print("per pixel:", round(ranges[1] ** 2 / (2 * R * F), 1))
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.