ANSWERS

U11.2 Intrinsics and calibration

Vision · University · about 35 min

What this lesson is about

Focal length in pixels, the principal point, and measuring them from known geometry.

- The intrinsic matrix
- Focal length in pixels, not millimetres
- What calibration actually measures
- Distortion
- Calibrating from what you have
- Why one point is not enough

Questions

7 marks in all

1. A 3.6 mm lens sits on a sensor 4.8 mm wide that produces images 640 pixels wide. What is the focal length in pixels? [1 mark]

> **Answer:** 480 (accept within 0.5). f_pixels = f_mm x image width / sensor width = 3.6 x 640 / 4.8 = 480 px.

2. The 320 pixel wide BugBot image, with fx = 92.4, is downsampled to 160 pixels wide. What is fx for the small image, in pixels? [1 mark]

> **Answer:** 46.2 (accept within 0.05). Intrinsics scale with the image: halve the width and fx and cx both halve, to 46.2 and 80.

3. What does a camera calibration actually compute? [1 mark]

- ☐ **A** A least squares fit of the intrinsics, distortion and target poses that minimises the reprojection error
- ☐ **B** The focal length printed on the lens, converted to pixels
- ☐ **C** The average colour of a chessboard under the room's lighting
- ☐ **D** The exact centre of the image in pixels

> **Answer:** A. Calibration fits the model to observations of known geometry, and the leftover reprojection error tells you how good the fit is.

4. Every calibration image shows the chessboard square on to the camera. What goes wrong? [1 mark]

- ☐ **A** The focal length and the target's distance trade off against each other, so the fit is degenerate
- ☐ **B** The corners cannot be found to sub-pixel accuracy
- ☐ **C** The principal point comes out at zero
- ☐ **D** Nothing, square on is the most accurate view

> **Answer:** A. A board further away with a longer focal length looks identical to a nearer board with a shorter one. Tilting the target breaks that tie.

5. Four tags give tan(bearing) and the observed u - 160. The model u - 160 = f tan(bearing) is a line through the origin, fitted by least squares. What does this print? [1 mark]

```
tans = [0.0, 30 / 70, -40 / 50, 40 / 30]
offsets = [0.0, 40.0, -74.0, 123.0]
f = sum(t * o for t, o in zip(tans, offsets)) / sum(t * t for t in tans)
residuals = [round(o - f * t, 1) for t, o in zip(tans, offsets)]
print(round(f, 1))
print(residuals)
```

Answer:

```
92.4
[0.0, 0.4, -0.1, -0.2]
```

The least squares gradient through the origin is sum(t o) / sum(t t) = 92.4 px, and every residual is under a pixel, so the pinhole model describes this camera.

6. Why does a tag straight ahead of the robot carry no information about f? [1 mark]

- ☐ **A** Its tan(bearing) is zero, so u - cx is zero whatever f is
- ☐ **B** A tag straight ahead is too close to measure
- ☐ **C** It lies on the principal point, which is unknown
- ☐ **D** Its range is too large for a good fit

Answer: A. u - cx = f tan(bearing), and with a zero bearing the equation is 0 = 0 for any f.

7. A calibration reports a root mean square reprojection error of 1.4 pixels. What should you conclude? [1 mark]

- ☐ **A** Something is wrong, such as blurred images, a target that was not flat, or too few angles
- ☐ **B** It is a good calibration
- ☐ **C** The camera needs a longer focal length
- ☐ **D** The residual is irrelevant once the intrinsics are known

Answer: A. Under about 0.3 pixels is a good calibration. Above a pixel, the fit is telling you the data or the model is at fault.

The task: measure the focal length

Four tags are on the mat at (100, 130), (130, 110), (60, 90) and (140, 70). The robot stands at (100, 40) facing along +y. Print `focal length:`, fitted in pixels, and `field of view:`, the full horizontal field of view in degrees that your focal length implies.

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

ROBOT = (100.0, 40.0)
TAGS = {1: (100.0, 130.0), 2: (130.0, 110.0), 3: (60.0, 90.0), 4: (140.0, 70.0)}
set_cv("apriltag")
wait(0.3)
```

The hint students can ask for: Four tags are stuck on the mat at positions the task tells you, and the robot is at a position it tells you too, facing along +y. So you know the true bearing of every tag, and you can read the column each one lands in. Plot the pixel offset against the tangent of the bearing: it is a straight line through the origin, and the focal length is its gradient. Fit it, do not take one point. The field of view then follows from the half width of the sensor.

A solution

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

# where the calibration target is, and where the robot is standing
ROBOT = (100.0, 40.0)
TAGS = {1: (100.0, 130.0), 2: (130.0, 110.0), 3: (60.0, 90.0), 4: (140.0, 70.0)}

set_cv("apriltag")
wait(0.3)

# one correspondence per tag: the true bearing, and the column it landed in
num = den = 0.0
for tag_id, cx, cy, dist in apriltags():
    tx, ty = TAGS[tag_id]
    truth = math.atan2(tx - ROBOT[0], ty - ROBOT[1])     # heading is 0, so this is the
bearing
    u = cx - 160.0                                       # offset from the principal point
    num += u * math.tan(truth)
    den += math.tan(truth) ** 2

f = num / den                                            # least squares gradient through
the origin
fov = 2 * math.degrees(math.atan(160.0 / f))
print("focal length:", round(f, 2))
print("field of view:", round(fov, 1))
```

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