ANSWERS

U11.5 Colour, and why it breaks

BugBotLab

Vision · University · about 35 min

What this lesson is about

Segmentation by threshold, what the threshold is really measuring, and where it fails.

- What a camera pixel measures
- Brightness is not colour
- The other four ways it fails outside the lab
- What to do instead, in order of effort
- Finding the centre

Questions

6 marks in all

1. Why does the test `r > 150` fail to find a red ball on the nearly white mat, even in a simulator with no lighting model? [1 mark]

- ☐ **A** It tests brightness, and white is red-rich, so the mat passes too
- ☐ **B** The simulator's red is below 150
- ☐ **C** The camera's white balance shifts the ball towards grey
- ☐ **D** The threshold should be in radians

> **Answer:** A. A white or pale surface has a large r. The question to ask is whether red dominates, not whether it is large.

2. Four pixels: white mat, a red ball, the same ball in shadow, and a pale pink wall. What does this print? [1 mark]

```
pixels = [(230, 225, 220), (200, 40, 30), (120, 30, 25), (240, 200, 180)]
bright = sum(1 for (r, g, b) in pixels if r > 150)
dominant = sum(1 for (r, g, b) in pixels if r - max(g, b) > 60)
chroma = sum(1 for (r, g, b) in pixels if r / (r + g + b + 1) > 0.5)
print(bright, dominant, chroma)
```

> **Answer:**
>
> ```
> 3 2 2
> ```
>
> The brightness test passes the mat, the ball and the wall but misses the shadowed ball. Both colour tests pass exactly the two ball pixels, lit and shadowed.

3. In a 64 pixel wide picture from the 320 pixel camera, the red pixels' centroid is at column 40. What is the bearing in degrees to 1 decimal place, using f = 92.4? [1 mark]

> **Answer:** 23.4 (accept within 0.1). Scale to the full frame first: column 40 x 5 = 200. Then atan((200 - 160) / 92.4) = 23.4 degrees. The intrinsics belong to the full frame.

4. In pixel = illumination x reflectance x sensor response, which factor is a property of the object? [1 mark]

- ☐ **A** Reflectance
- ☐ **B** Illumination
- ☐ **C** Sensor response
- ☐ **D** All three

> **Answer:** A. Illumination belongs to the room and sensor response to the camera. A raw RGB threshold is a threshold on all three.

5. Which of these break a colour threshold that worked in the lab? [1 mark]

Tick every answer that is true.

- ☐ **A** Auto exposure changing the gain when a dark object fills the view
- ☐ **B** Auto white balance shifting the channels
- ☐ **C** A shadow or a specular highlight across the object
- ☐ **D** Different light, such as fluorescent instead of daylight
- ☐ **E** Converting the column to a bearing with the pinhole model

> **Answer:** A, B, C, D. Each changes the pixel values without changing the object. The pinhole conversion is geometry and does not touch colour.

6. Why is a specular highlight worse for a chromaticity test than a shadow? [1 mark]

- ☐ **A** A highlight saturates the channels, and a clipped channel has thrown its information away for good
- ☐ **B** A shadow is darker, so it is easier to threshold
- ☐ **C** Chromaticity divides by brightness, which is zero in a highlight
- ☐ **D** Highlights only affect the red channel

> **Answer:** A. A shadow drops all three channels and can partly survive normalisation. Once a channel clips at its maximum the ratio is gone and cannot be recovered.

The task: threshold a picture

A red ball is ahead of the robot. Take a 64 by 48 picture with `camera_image(64, 48)` and print three things: `naive:`, the percentage of the picture that passes a plain `r > 150` test; `red:`, how many pixels pass a test that looks at colour rather than brightness; and `bearing:`, the bearing in degrees of the centroid of those pixels.

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

F = 92.4
W, H = 64, 48
img = camera_image(W, H)
```

The hint students can ask for: Take a 64 by 48 picture and count, twice. First with the test a beginner writes, red above some level, and express the answer as a percentage of the whole picture, because the number itself is the lesson. Then with a test that asks whether red dominates the other two channels rather than whether the pixel is bright. Average the columns of the pixels that survive the second test, scale that column back up to the camera's own 320 wide frame, and turn it into a bearing the way you did in U11.1.

A solution

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

F = 92.4
W, H = 64, 48

img = camera_image(W, H)

bright = 0
cols = []
for row in img:
    for u in range(W):
        r, g, b = row[u]
        if r > 150:
            bright += 1                      # the beginner's test: bright, not red
        if r - max(g, b) > 60:
            cols.append(u)                   # red dominates the other two channels

print("naive:", round(100.0 * bright / (W * H), 1))
print("red:", len(cols))

u = sum(cols) / len(cols)
x = (u + 0.5) * 320.0 / W                    # back to the camera's own 320 wide frame
print("bearing:", round(math.degrees(math.atan((x - 160) / F)), 2))
```

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