RoboCup

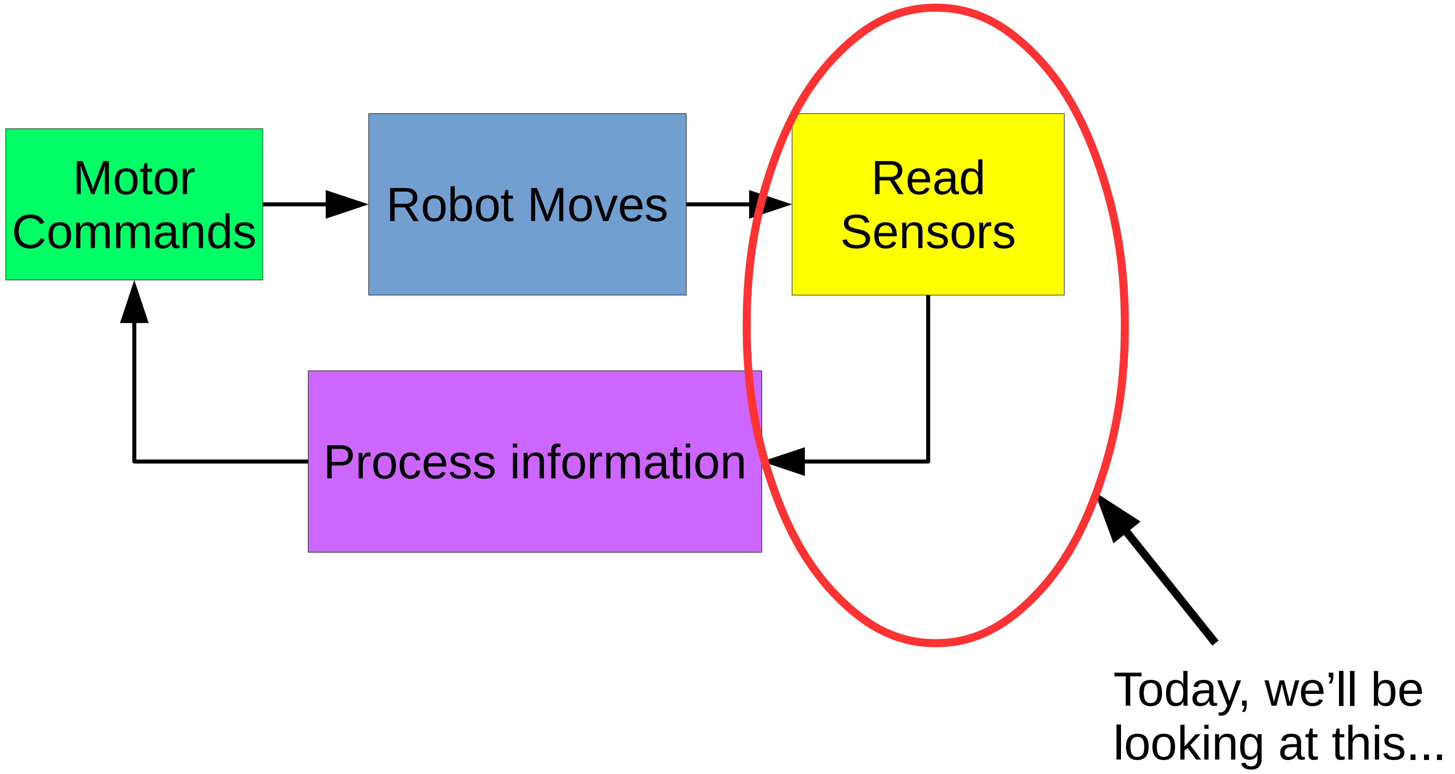
RCAP CoSpace Autonomous Driving (Line Following Intermediate)

A POSTERIORI

Play · Experience · Learn

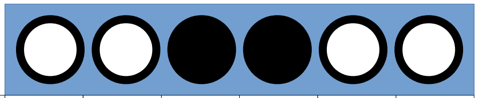
Basics of Line Following

- Feedback loop



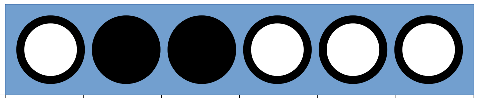
IR Sensor Array

- 6 Sensors that can read:
  - 1 : White
  - 0 : Black
- What are the possible combinations of values?



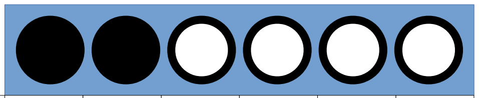
Values:	1	1	0	0	1	1

Line is in the center; go straight



Values:	1	0	0	1	1	1

Line is slightly left; turn slight left



Values:	0	0	1	1	1	1

Line is very left; sharp turn left

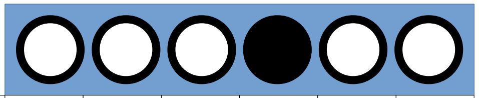
Question
How many different combinations are there?

Answer

- With two blacks next to each other...
  - 5 combinations

Sensor values					
0	0	1	1	1	1
1	0	0	1	1	1
1	1	0	0	1	1
1	1	1	0	0	1
1	1	1	1	0	0

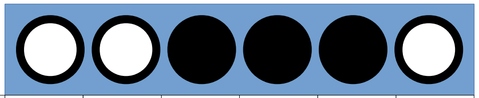
- ...but what about...



Values:	1	1	1	0	1	1

...another 6 more combinations

- ...and...



Values:	1	1	0	0	0	1

...another 4 more combinations

<u>Answer</u>
At least 16 combinations!
(...did I count wrongly?)

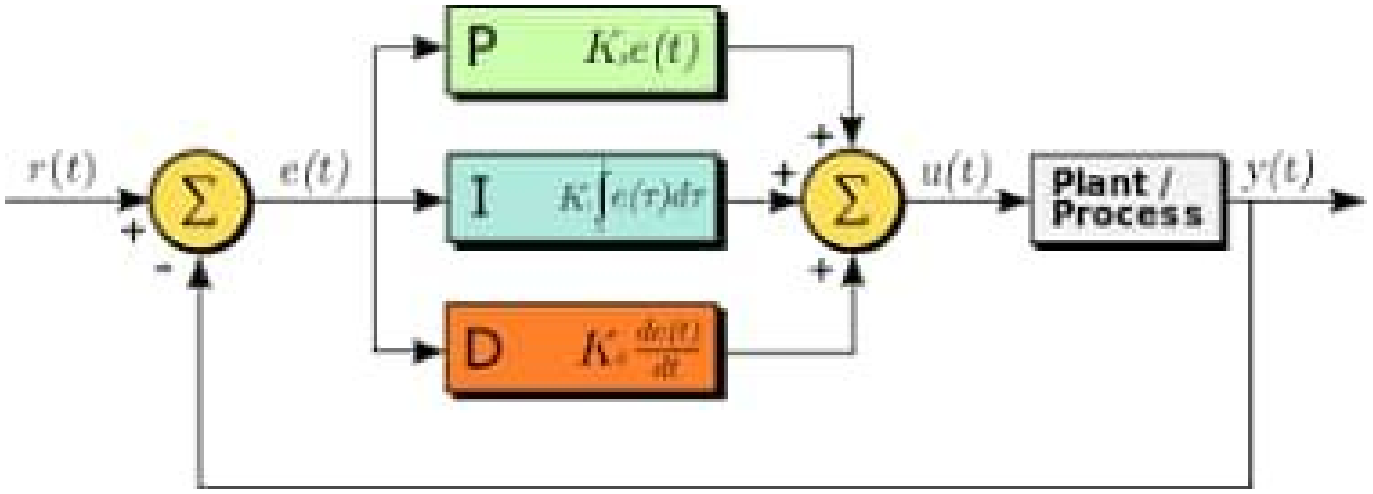
Objective

- Turn the multiple sensor values into a single number that represents the line position
- Example:

Sensors	**Single value**	**Meaning**
○○●●○○	0	Line is in the center
○○○●○○ ○○●●●○	1	Line is slightly to the right
○●●○○○	-2	Line is slightly to the left
○○○○○●	5	Line is far to the right

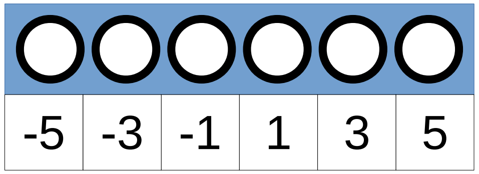
Why?

- Works for any number of black
- Accurate detection of line position
- Easy to program
- Advanced algorithms expects a single input value



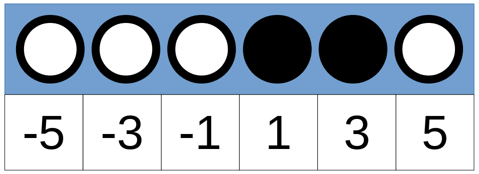
How?

- Let every sensor be represented by a number
  - Example:



...numbers should be equally spaced

- Add up the numbers where a line is detected
  - Example:

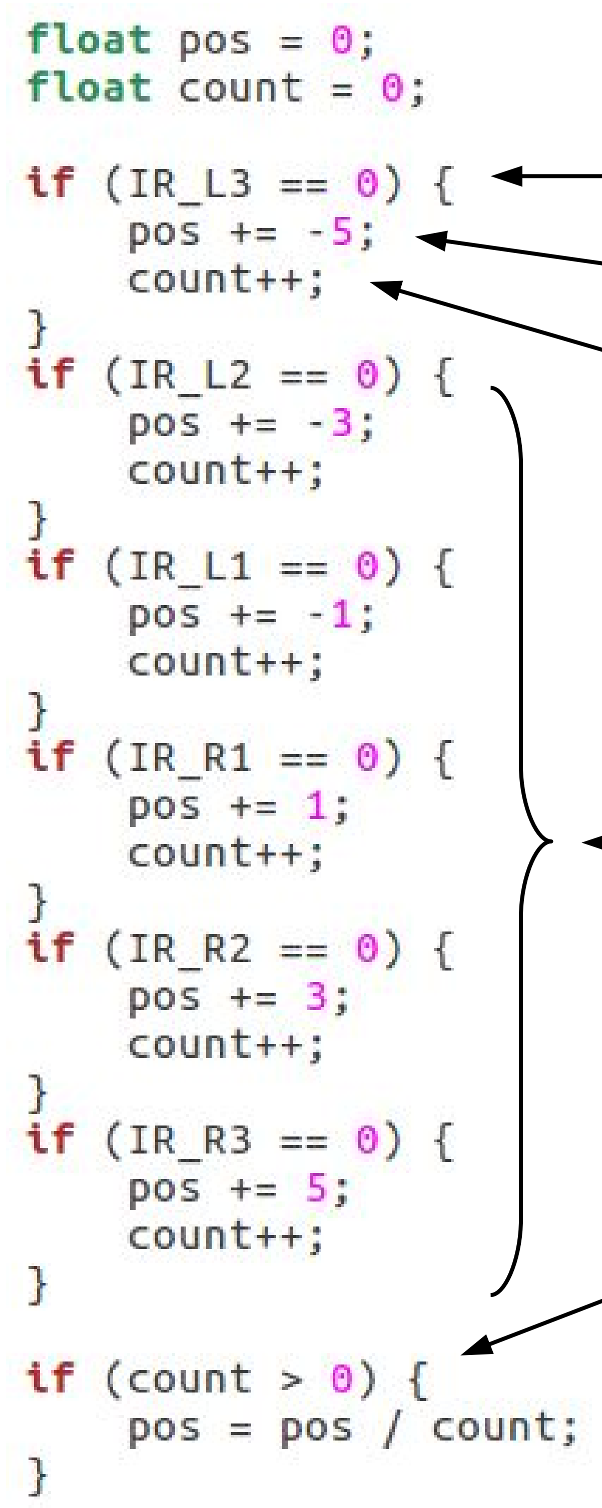


We'll add up 1 and 3
Total: 4

- Calculate the average
  - Example: Average = 4 / 2 = 2

This is the number that represents the center of the line

Code Template



```
float pos = 0;
float count = 0;

if (IR_L3 == 0) {
    pos += -5;
    count++;
}
if (IR_L2 == 0) {
    pos += -3;
    count++;
}
if (IR_L1 == 0) {
    pos += -1;
    count++;
}
if (IR_R1 == 0) {
    pos += 1;
    count++;
}
if (IR_R2 == 0) {
    pos += 3;
    count++;
}
if (IR_R3 == 0) {
    pos += 5;
    count++;
}

if (count > 0) {
    pos = pos / count;
}
```

If IR_L3 is black...

...add -5 to the total

...increase the count by one

Repeat for every sensor (...but with a different value to add)

If there is at least one item (count)...

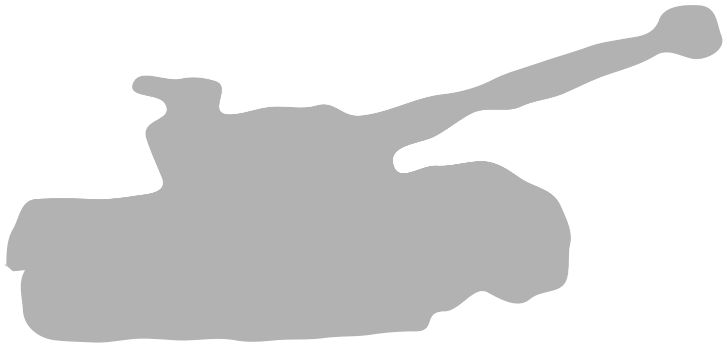
...calculate the average:
Total (pos) divide by number of items (count)

The code can be further shortened with a loop (...important if there are more sensors), but it’s not done here to keep it easy for you to understand.

Other Applications

- Use with a camera image for line following
  - Camera images have hundreds or thousands of pixels; try writing conditions for every possible combinations of that!

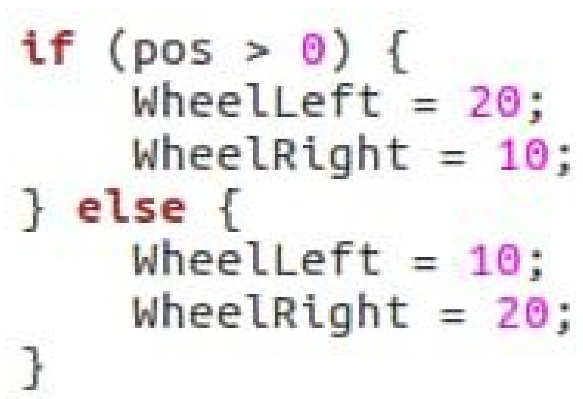
- For finding center of a shape
  - Useful for identifying position of objects in a photo



Silhouette of a tank
How would you find the center?

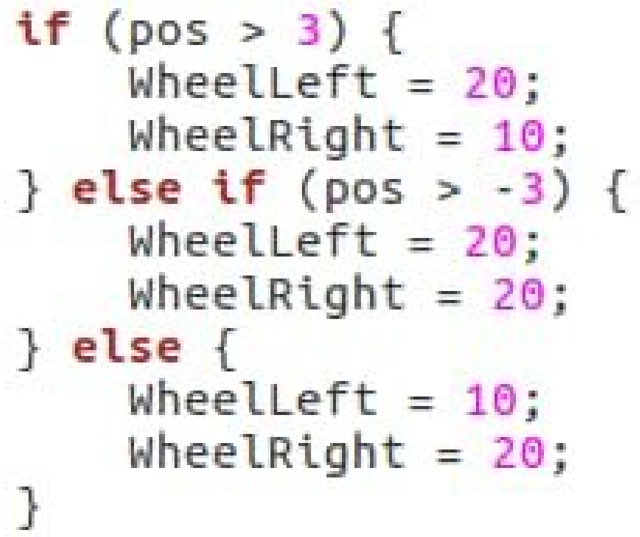
What's next?

- Now that we know the position of the line (pos), we can apply any number of methods for line following, such as...



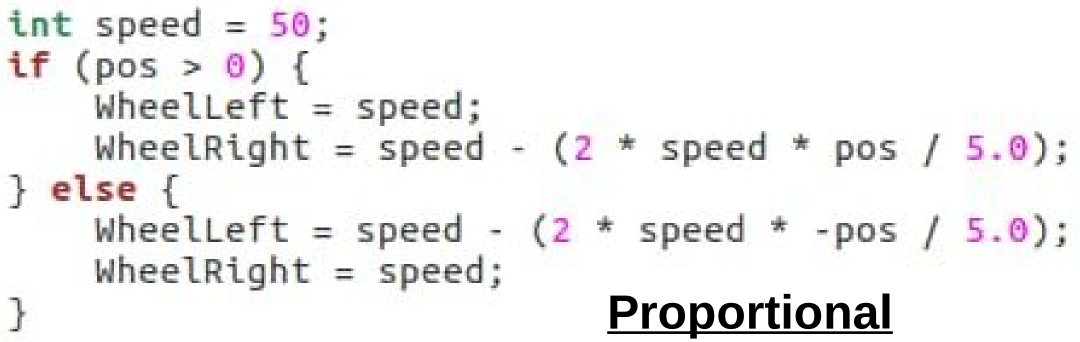
```
if (pos > 0) {
    WheelLeft = 20;
    WheelRight = 10;
} else {
    WheelLeft = 10;
    WheelRight = 20;
}
```

2 States



```
if (pos > 3) {
    WheelLeft = 20;
    WheelRight = 10;
} else if (pos > -3) {
    WheelLeft = 20;
    WheelRight = 20;
} else {
    WheelLeft = 10;
    WheelRight = 20;
}
```

3 States



```
int speed = 50;
if (pos > 0) {
    WheelLeft = speed;
    WheelRight = speed - (2 * speed * pos / 5.0);
} else {
    WheelLeft = speed - (2 * speed * -pos / 5.0);
    WheelRight = speed;
}
```

Proportional

If you want to improve...

- Program in a structured manner
- Build useful functions that you can reuse
- Move out of your comfort zone, don’t just keep doing the same thing

Copyright