

G54MRT

Python and Sensors 3: Algorithm Development

Joe Marshall, Stuart Reeves

Sensor processing for the coursework

- What you need to do for the coursework is covered in these four technical lectures:
- Lecture 1: Get data from sensors
- Lecture 2: Filter the raw data so that it is usable
- **Lecture 3: Combine two or more data streams to make some kind of inference, such as:**
 - Has an event happened (e.g. someone has opened a door, someone has knocked on the door)
 - The value of an unknown quantity (e.g. how many people do we think are in a room, how 'busy' is the room, what shape is the room)
 - What is happening to a device (e.g. is it being thrown around, has it been dropped, is it outside or inside)
- Lecture 4: Test it

Do something clever with it

Your project must use at least **two sensors**.

And do some meaningful processing that analyses or responds to the data from both of them. 

You can't just display sensor data.

You can (for example):

- Use sensor data to make inferences about things, e.g.
 - What is the state of the room / environment?
 - What is happening to the system (or an object to which it is attached)?
 - Who is using a system?
- Use sensor data as an input device, e.g.
 - Where is something being touched?
 - How hard is something hit?

In the report, you need to demonstrate:

- What you based your sensor algorithms on (real world data, algorithms from research literature ...)
- What your sensor algorithms do?
- Do they work? (demonstrate by running tests and reporting numerical results & statistics)

What is a sensor (again)

- What is a sensor?
- ‘A device that receives a stimulus and responds with an electrical signal’
- Real world → electronics (input not output)
 - LED screen, buzzer, LED lights are not sensors
- **Responds** is not the same as **equals**
- **What a sensor senses may not be what you are actually interested in**

What we get from sensors

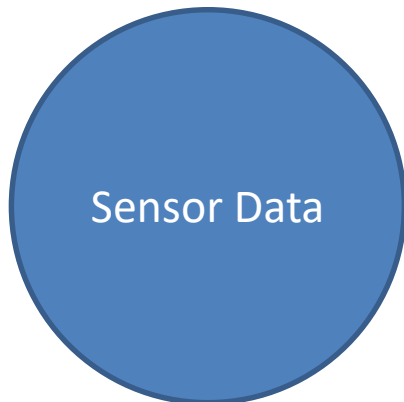
- Estimates of physical quantities
- e.g. PIR motion sensor
 - Measures whether it sees changes in emitted heat (infrared) levels from one part of the sensor to another
 - Outputs 1 for a second or so if it sees a difference.
 - Outputs 0 otherwise
 - Doesn't know anything about people or animals.
- Accelerometer
 - Measures the acceleration forces on the accelerometer chip.
 - Doesn't inherently know anything about tilting, vibrations or anything else.

The kind of things we are typically interested in

- Human (or animal) behaviour
 - What is a person doing?
 - Why are people behaving in the way they are?
 - What is someone doing to an object?
- The state of the world
 - What is happening in a room?
 - What is happening in the world around us?
 - What is happening to an object?

Sensor processing

We have



processing



We get



Why use multiple sensors?

- Sensors only respond to part of a situation
- Different sensors may have different noise characteristics
 - Combining the two may lead to far lower noise levels.
- Sensors may respond to the same thing but have different ranges (physical or sensor range)
 - Combining the two can allow a wider range of events to be sensed
- Sensors may sense different aspects of the same situation
 - Combining the two can give us a greater understanding of the whole situation.

Noise characteristics

- Example: Detection of motion in a room
- Sensor 1: PIR
- Sensor 2: Loudness sensor
- PIR
 - Very low noise, false positives are rare
 - On/off response
- Loudness sensor
 - Quite a lot of high frequency noise
 - Needs a lot of filtering to be usable without false positives.
 - Gives a nuanced idea of how much sound someone is making

Algorithm example:

if PIR == 1:

 movement level = filtered loudness

else:

 movement level = 0

Different ranges

- Same example as before, detection of motion in a room
- Sensor 1: PIR
- Sensor 2: Loudness
- Covers full room
- More accuracy in the PIR zone



Algorithm example:

```
if PIR==1:
```

```
    move_probability=1
```

```
else:
```

```
    move_probability = loudness / max_loudness
```

Different aspects of the same situation

- I have shamelessly stolen some interesting situations from previous years' projects to talk about here
 - Because they had some better ideas than me
 - Because it forces me to think about situations I hadn't previously considered.

Different aspects of the same situation 1

- Example: How is someone riding a mountain bike?
 - How fast are they riding?
 - GPS, magnetic switch and wheel magnet
 - How bumpy is the terrain?
 - Accelerometer
 - How tightly do they turn?
 - Accelerometer, gyroscope, magnetometer
 - How long do they stay in the air on a jump?
 - Accelerometer

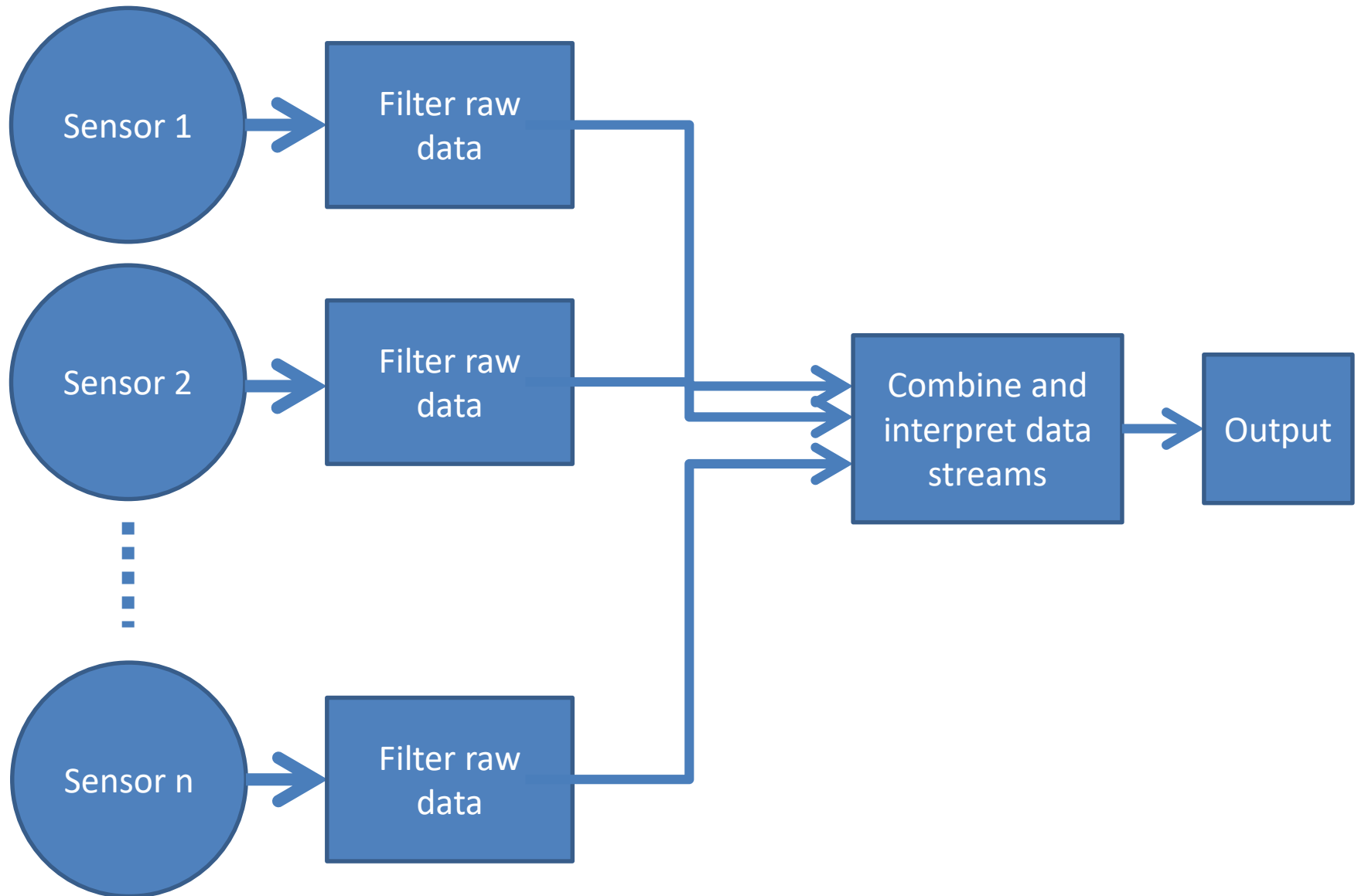
Different aspects of the same situation 2

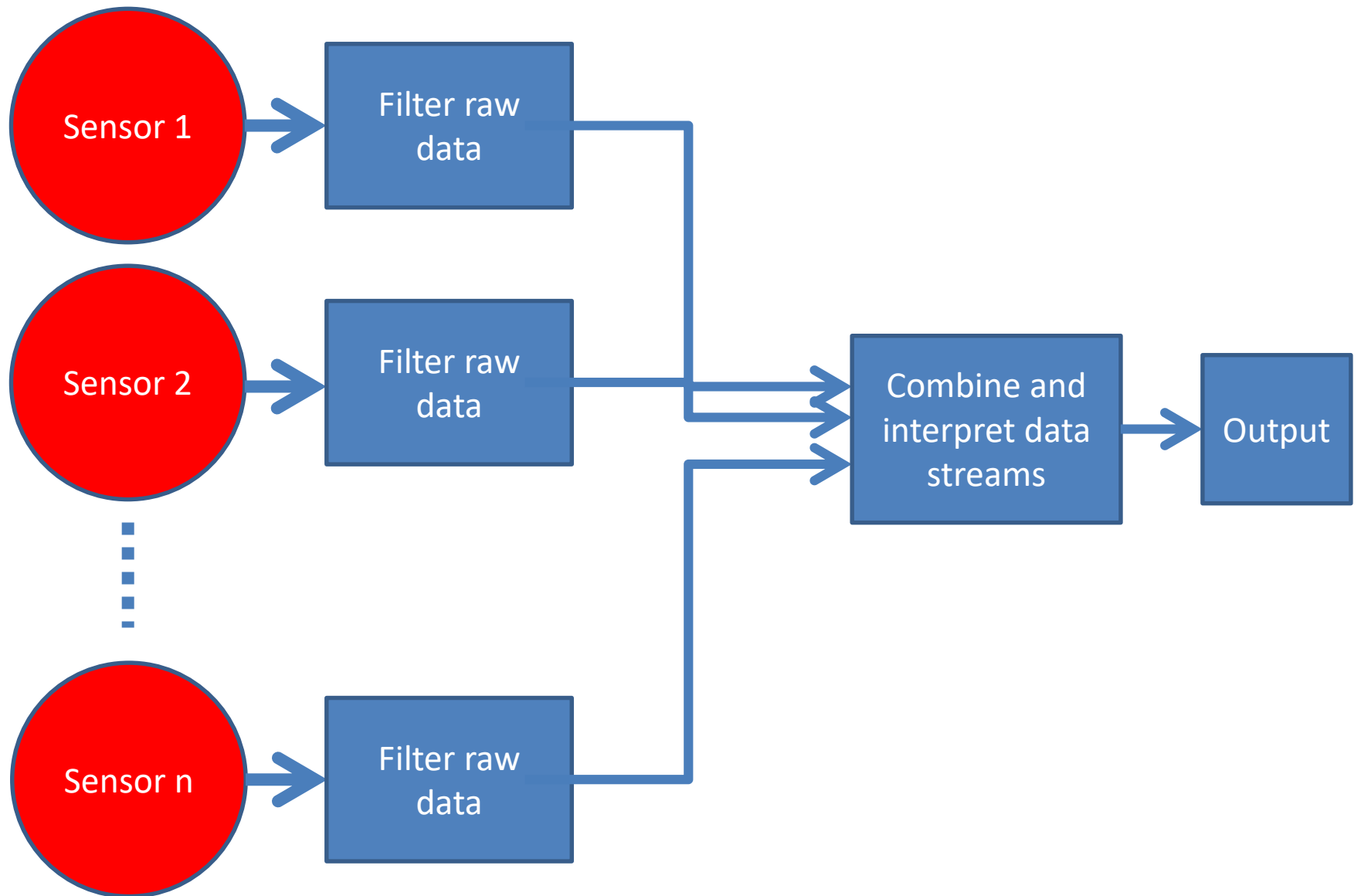
- If you have remotely accessible motion sensors in all toilet cubicles in a bathroom, what are these useful for?
 - Remote engaged / vacant detection
 - So you can check before you leave your office
 - Building management
 - They can tell how much the toilet is used. Do they need to fit more toilets?
 - They can tell which toilets are used most – are there any problems with the layout of the bathroom?
 - If people can check vacancy, they can tell how often people want to go, but find it engaged. Does this mean they need to fit more toilets?
 - Maintenance and cleaning
 - If the usage pattern of one toilet changes a lot, what does it mean?
 - What do we expect people to do after they check the toilet vacancy?
 - Particular usage patterns that imply a broken or unusably dirty toilet.
- What is the situation as a whole that we are monitoring?
 - Usage patterns of the bathroom.

Basic algorithm development recipe

- Not the only way to develop an algorithm
 - In particular, I say filter individual signals and then combine them. Some cunning engineering algorithms combine signals and then filter them – called Sensor Fusion Algorithms
- But good enough for most coursework projects

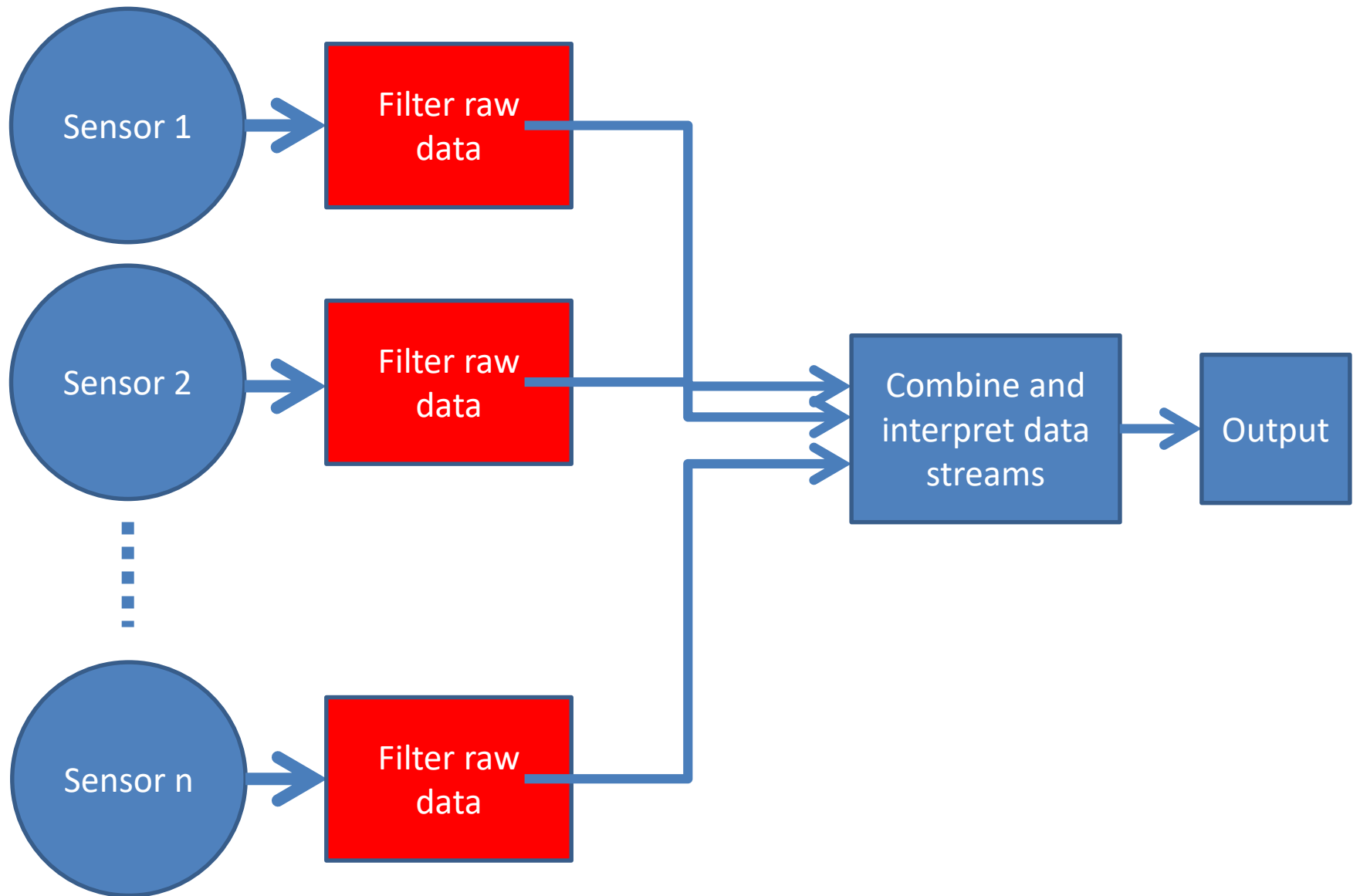
An algorithm structure

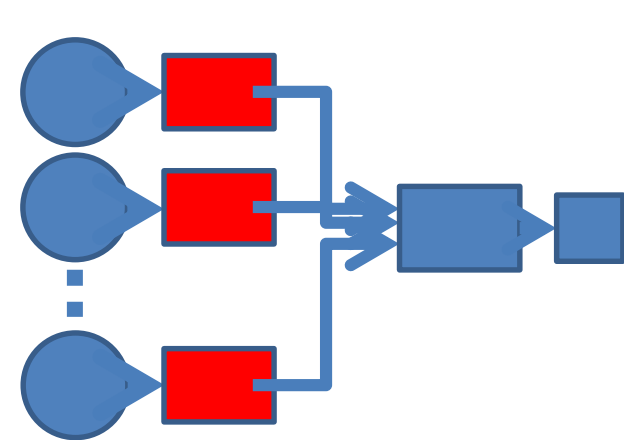




Step 1: Look at your data

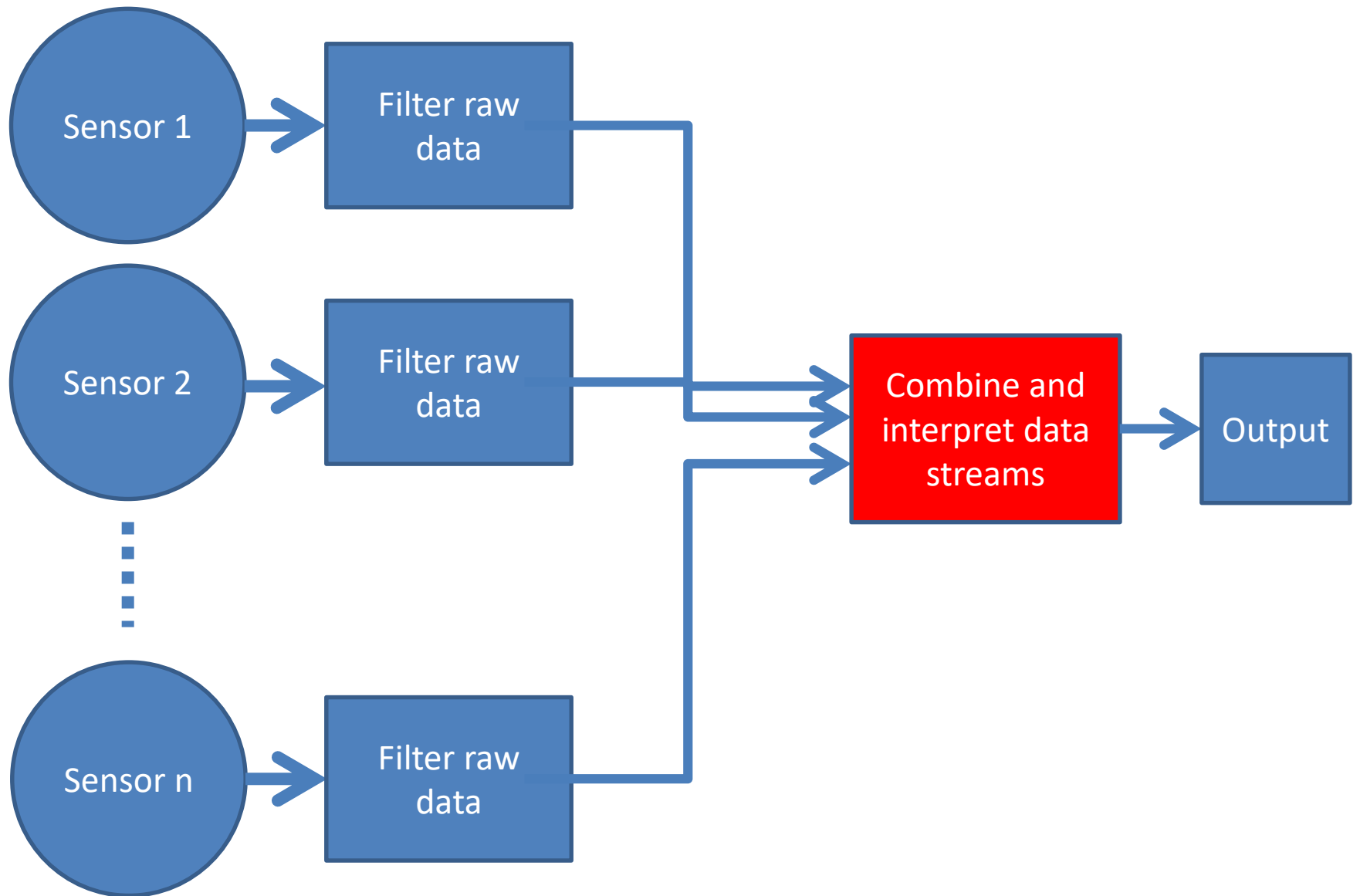
- Look at your sensor data (in Excel, in the program output, in text files, see lecture 5)
- Look at how the sensors respond to things of interest (performing actions, noise level in the room....)
- Basically do the stuff in sensors 1 lecture for all your sensors, while keeping track of what they are responding to.





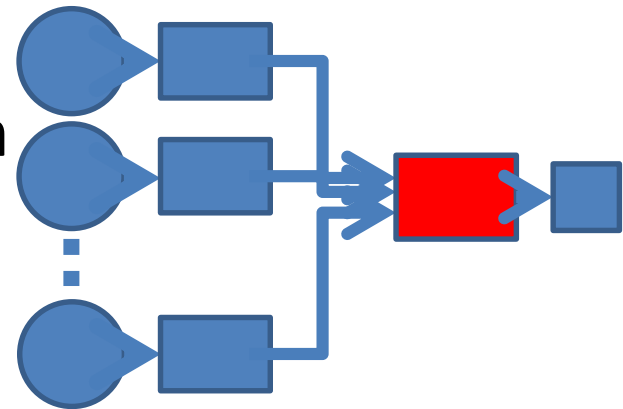
Step 2: Filter raw sensor data

- Often you will need to filter some sensors first to get a usable signal
- For various purposes
 - Noise reduction
 - Counting numbers of or durations of sensed events
 - Generating statistics about multiple sensor values (e.g. maximum over a time period, mean over a time period)
- Methods for doing this are described in sensor lecture 2.
- Be aware that filtering may induce a time delay relative to other sensors.
 - E.g. exponential filter, delays by time constant, median filter, delays by half the length of the buffer.



Step 3: Combine sensor data

- Take multiple sensor streams and combine them
- Output is some kind of interpretation of the likely state of whatever we are observing e.g.
 - How many people are in a room?
 - Is someone moving in a room?
 - Did someone just leave the room
- Lots of ways to do this



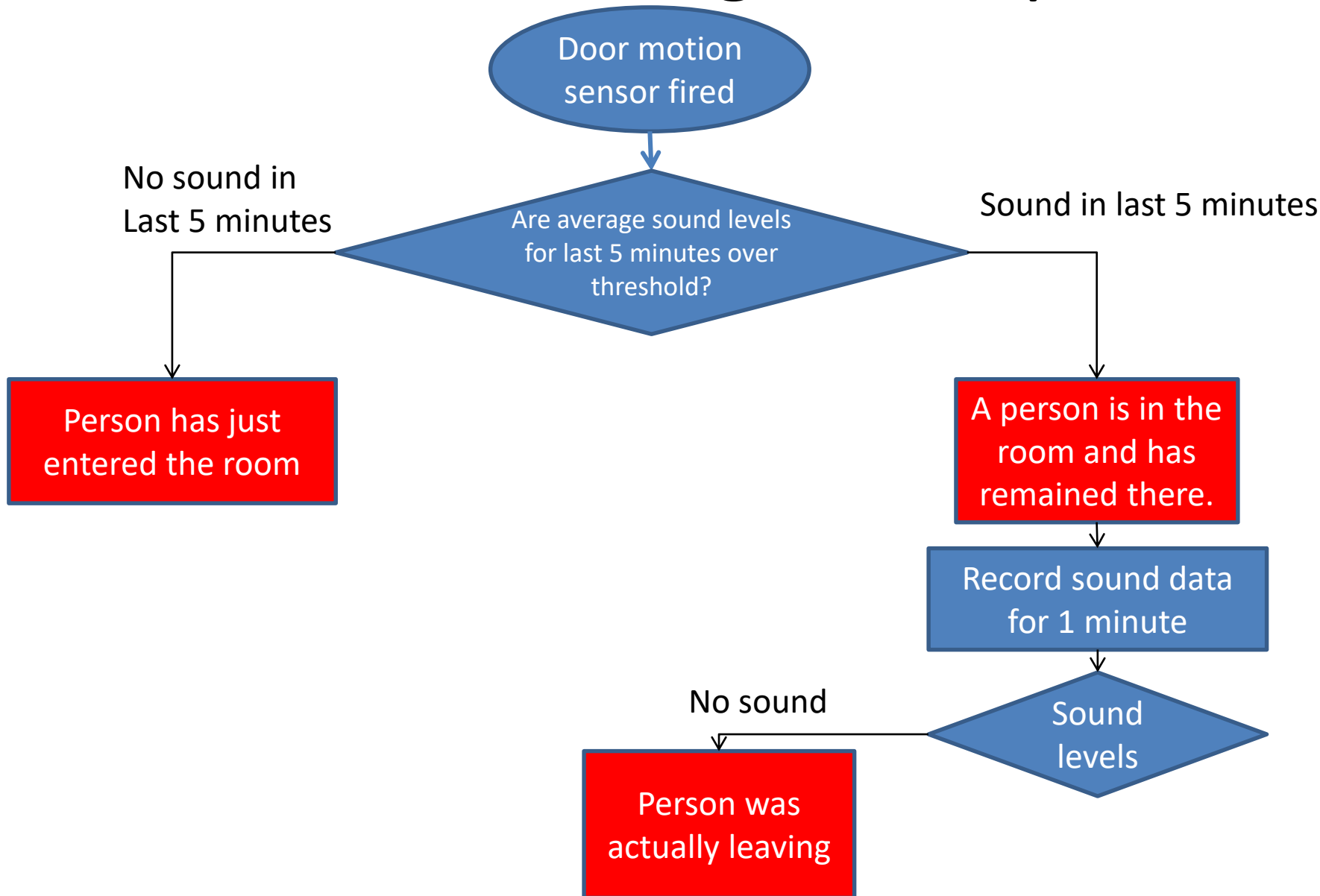
How to combine multiple sensors in an algorithm

- If this then that
- (Combined) Thresholds
- Probabilistic modelling
- Sensor fusion

Combining sensors 1: If this then that

- Simplest way to combine sensors to make inferences
- Use basic logical statements to say when an event is true.
- E.g. if motion sensor fires then no sound or motion sensed in the minute after, then a person has left the room.
- Can make logical statements about thresholds, counters of events over time, sensor value statistics etc.
- Need to use your knowledge of the situation being sensed, along with an idea of the likely response of the sensors gathered from preliminary testing.

If this then that logic example



Combining sensors 2:

Multiple thresholding

- You may threshold sensors and say that a greater than threshold value is an event (see sensors 2 lecture)
- If you have multiple sensors, you can combine this with if this then that logic.
- Example 1: (using different thresholds depending on the value of each sensor)

```
if sensor1>500: fireEvent()  
elif sensor2>500: fireEvent()  
elif sensor1>250 and sensor2>250: fireEvent()
```
- Example 2: (using weighted sum of two sensors)

```
if sensor1+2.0*sensor2>500: fireEvent()
```
- Don't forget values may be filtered before they get to this thresholding bit too


Combining sensors 3:

Probability and individual sensors

- For each individual sensor, through structured measurement of sensor false positives, negatives, and estimation of event probabilities, we can calculate a probability that a particular sensor value means an event occurs.
- Use Bayes' Theorem:

- $$P(Event|SensorVal) = \frac{P(SensorVal|Event)P(Event)}{P(SensorVal)}$$

- $$= \frac{P(SensorVal|Event)P(Event)}{P(SensorVal|Event)P(Event) + P(SensorVal|\neg Event)P(\neg Event)}$$


True positive rate


Base rate of event happening


False positive rate

Combining sensors 3:

Probability and individual sensors

- Calculating true and false positive rates
 - Measure sensor a lot of times, both when events are or aren't happening.
 - *true positive rate* = $\frac{\text{count of (sensor=1 and event=1)}}{\text{count of (sensor=1)}}$
 - *false positive rate* = $\frac{\text{count of (sensor=1 and event=0)}}{\text{count of (sensor=1)}}$
- Estimating event base rates
 - Either: Collect real data over a period of time to estimate on average how many times event = 1 will be true.
 - Or: Make an estimate based on some factual knowledge of the underlying event

Combining sensors 3:

Probability and multiple sensors

- If sensors are all true/false sensors, we can treat each combination of sensor values as a value of a single combined sensor.
 - E.g. if we have: motion sensor and a thresholded sound level sensor, we have 4 combined values.
 - Probabilities calculated the same as on a couple of slides back, need to calculate $P(\text{Event} \mid \text{FF})$, $P(\text{Event} \mid \text{TF})$, $P(\text{Event} \mid \text{FT})$, $P(\text{Event} \mid \text{TT})$, using the same formula

Motion	Sound	Combined value
False (F)	False (F)	FF
True (T)	False(F)	TF
False(F)	True (T)	FT
True (T)	True (T)	TT

Combining sensors 4:

Numerical sensor fusion

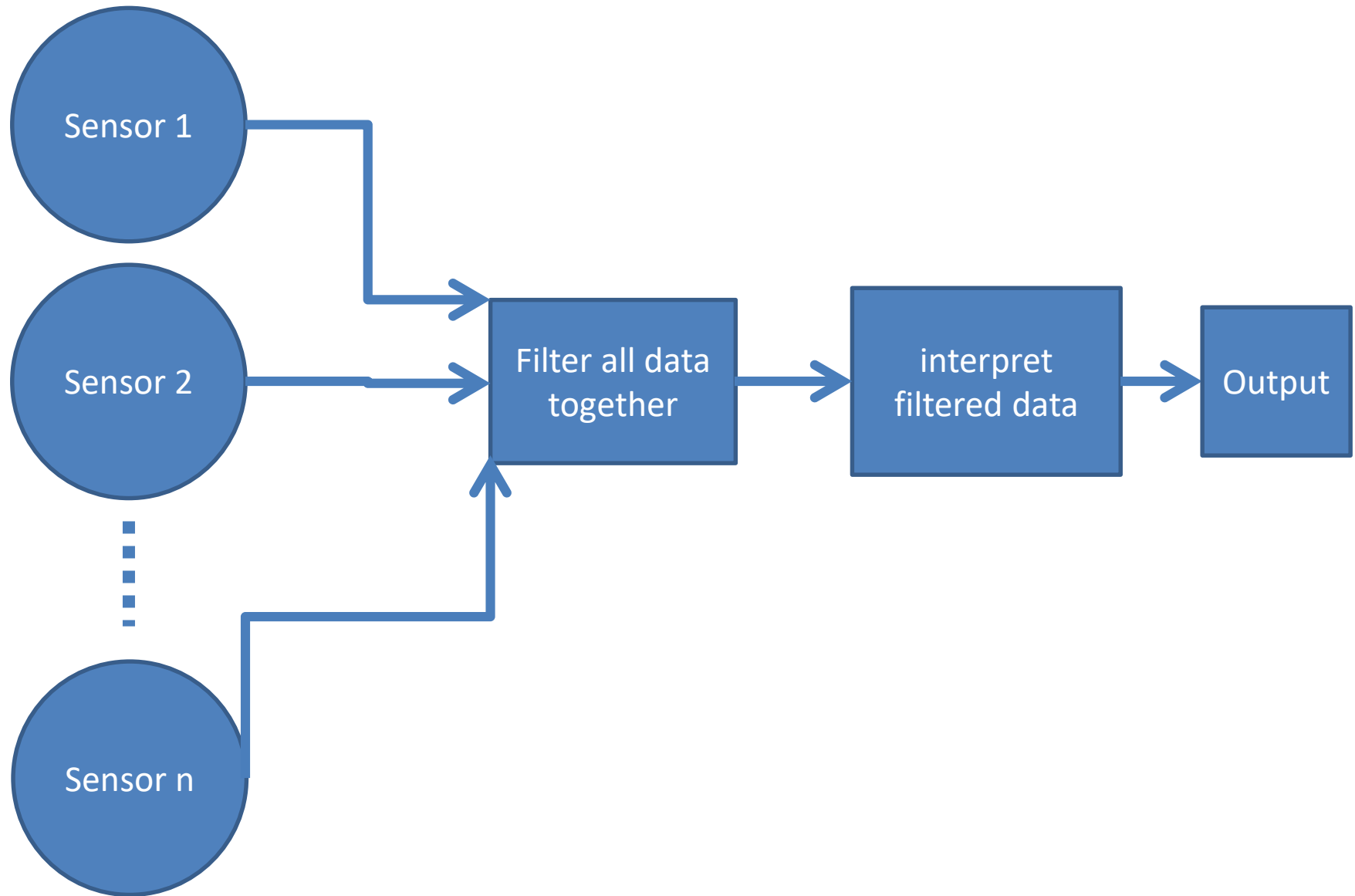
- Mathematical methods for estimating the state of a real system based on sensor readings. They take:
 - A statistical model of multiple sensors, their error characteristics, accuracy etc.
 - A statistical model of how the sensed underlying real value changes
- Useful if you have a bunch of numerical (e.g. analog) sensor values, and want to estimate the value of some process being sensed
 - E.g. Used in mobile phones or radio controlled planes to take accelerometer / gyroscope / magnetometer measurements and estimate the orientation of a device.
 - E.g. Could potentially be used to take a bunch of analog sensors (e.g. sound, light) and estimate the number of people in a lab.
 - You can potentially use things derived from digital sensors, such as number of times sensor fired in the last minute, or amount of time sensor was on in the last hour, but be careful as to whether these fit the criteria of your fusion algorithm (e.g. linear versus non-linear, error distributions)
- Mathematically quite advanced (Engineers or people with mathematical backgrounds might want to use these, but if you don't come from that background, you might well be best avoiding).
- Examples of algorithms:
 - Kalman filter (and extended kalman filter)
 - Particle filter

Combining Sensors 4.5

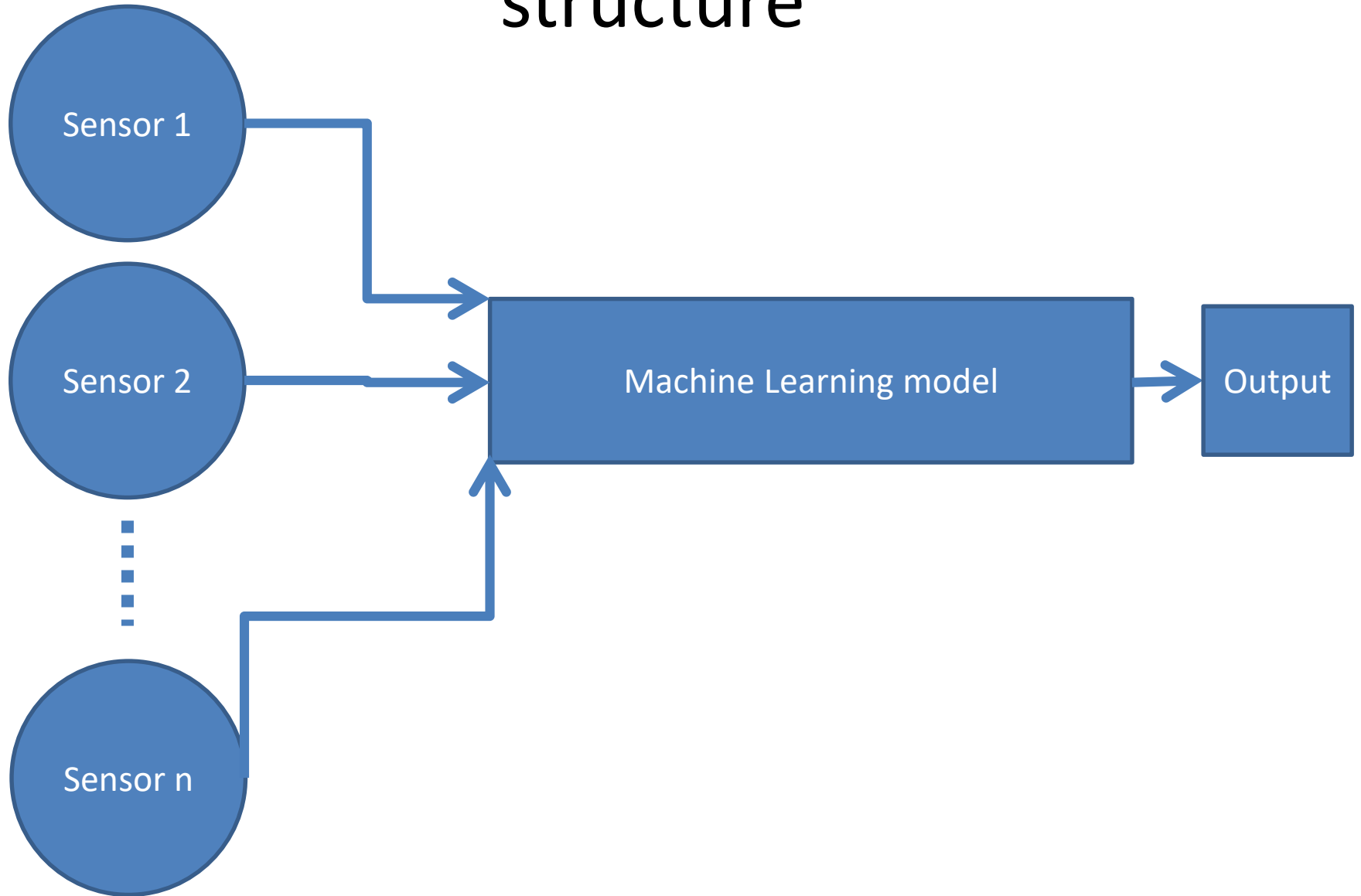
Machine Learning

- Way of doing sensor fusion using automated models
- State of the art in most sensor processing systems now.
- Typically:
 - **Train** neural network or other ML model with training data where ground truth is known.
 - **Test** trained network with known testing dataset
 - **Run** trained network to output predictions in your project.
- Beyond scope of course, but loads of stuff on internet for anyone interested (search 'introduction to deep learning' for example)
- Design and training of models is a bit of an art.
- Raspberry PIs have tensorflow library installed.

Sensor fusion algorithm structure



Machine learning sensor fusion structure



Here is one I made earlier

- A couple of real world examples of sensor processing algorithms from my work
- The overall systems are probably more complex than you are likely to use in the coursework
- But it is all based on the same basic principles

Touch-o-matic: Detect Two People

- Two player game
- Played by detecting players touching each other
- Need to stop people playing it as just one player
- Single player rejection algorithm



Touch-o-matic single player rejection

- Start game requires a high five
- Sensing:
 - Capacitive – is someone touching each pole.
 - Circuit – is there a circuit between the poles
 - (caused by players touching hands, or one player holding both)
 - Both signals thresholded to be active / non active for high-five sensing
- High five sequence is:
 - no circuit **Both capacitive sensors active**
 - yes circuit
 - no circuit **Both capacitive sensors active**
- No circuit plus both capacitive sensors active can't happen if one person is holding both sensor poles

```
1. int twoPersonAccumulator=0;
2. void findTwoPeople()
3. {
4.     // we keep a counter here of how many frames we have
5.     // seen recently where we thought there were two people
6.     twoPersonAccumulator--;
7.     if(twoPersonAccumulator<0)twoPersonAccumulator=0;

8.     // if someone is holding onto both sensors
9.     if(leftVal>100 && rightVal>100) // threshold capacitance
10.    {
11.        // but there is no contact in the middle
12.        if(sqrt(var)<100) // threshold resistance
13.        {
14.            twoPersonAccumulator+=2;
15.            // only report two people if we have seen two people for 8
16.            // frames in a row
17.            if(twoPersonAccumulator>16)
18.            {
19.                Serial.println("Found two people");
20.            }
21.        }
22.    }
23.}
```

Swim stroke sensing

- Senses swimming, using accelerometer, gyroscope and magnetometer mounted on back.
- Counts strokes.
- Uses:
 - Body side to side orientation
 - Body front to back orientation

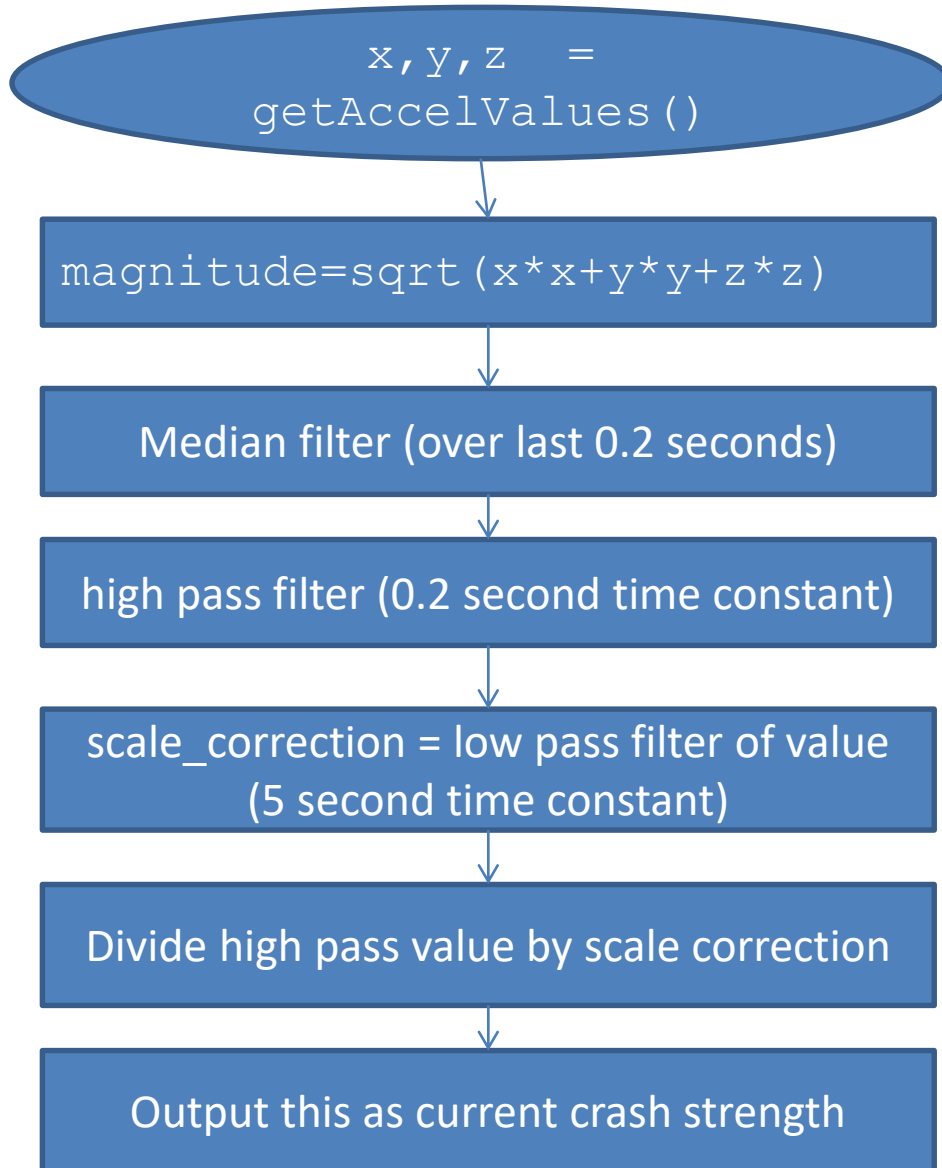
Swim stroke algorithm

```
lastRoll, strokes=(0,0)
while true:
    #Read data from all sensors
    sensorValues=getSensorData()
    #Combine sensors to get body orientation (including front-
back pitch, left-right roll)
    pitch,roll=getOrientation(sensorValues)
    # we aren't swimming at all unless body is flat (pitch<20)
    if abs(pitch)<20:
        # threshold roll to be either left, right, or centre
        if roll<-20:
            currentRoll =-1 # left
        elif roll>20:
            currentRoll=1 # right
        else:
            currentRoll=0 # centre
        if currentRoll!=0:
            if lastRoll!=currentRoll:
                #rolled to the other side - count a stroke
                strokes+=1
                # save the side that the body rolled to last
                lastRoll=currentRoll
```

Accelerometer crash sensing

- Detects how hard a person falls over or is bumped into.
- Done using an accelerometer in their pocket.
- Built for a game involving physical contact.
- Designed to work on a variety of devices, which have different sensor characteristics.
- Only uses one sensor, but processing of that sensor to keep the game fair is hard.

Accelerometer crash algorithm



Remove sensor noise

Remove bias of sensor (and gravity offset)

Get the average magnitude of the sensor responses so we can normalise for device differences in response level.

Summary

- Sensors don't directly sense what you want to know
- Multiple sensors help you better estimate what you actually are interested in.
- There are a lot of ways to combine multiple sensors
 - Not necessarily one right way
 - Draw diagrams, write pseudo-code etc. to help you work out the underlying logic.
 - Ask us in labs for help with your algorithm
 - Sensor algorithm design and interaction design are intrinsically linked (see Bellotti paper in Ubicomp Interaction Design lectures)
- Next sensor lecture – testing and evaluating algorithms
 - On moodle now in case anyone is ahead of themselves