Sensing for action

IAR Lecture 3

Barbara Webb

Sensing for Action

- Sensors transduce energy from one form to another
- From the robot control point of view we have some information – a measured value – that represents some property of the world
- This relationship is rarely a direct one:

E.G. We say the IR sensor is a ‘range’ or ‘distance’ sensor:

Distance to object →

Light scattering →

Amount of light reflected →

Resistance of sensor element →

Voltage →

Analog to Digital Conversion→

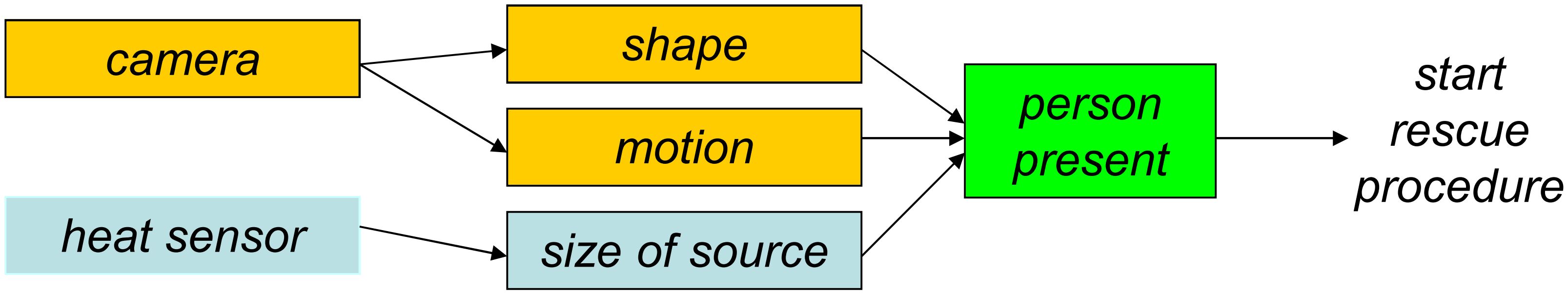
Calculation →

Distance value

But note! We may not need to know the actual distance to perform the appropriate action, such as “avoid”

Virtual/logical sensors

- An abstraction over the physics of specific sensors
- Task oriented definition: start by defining the property you want for robot control (e.g. ‘person detector’ for a rescue robot)
- Design set of input sources (may include other virtual sensors), and computation (may be in hardware) that produces appropriate output vector



- For robot control purposes can treat as ‘direct’ sensing of desired property
- Within the module, may be able to redesign, use different sensors to obtain same (or improved) effective sensing capability.

- In practice, usually need to take into account the real physics of the sensor

Describing sensors

- *Sensitivity*:
  - ratio of output change to input change
  - Usually a trade-off with *range* (min to max)
- *Resolution*:
  - Limit in resolving power of output scale (note is a property of the measuring *instrument* rather than the sensor)
- *Precision:*
  - Repeatability of measurements (under same conditions)
- *Accuracy:*
  - (lack of) error in measurements

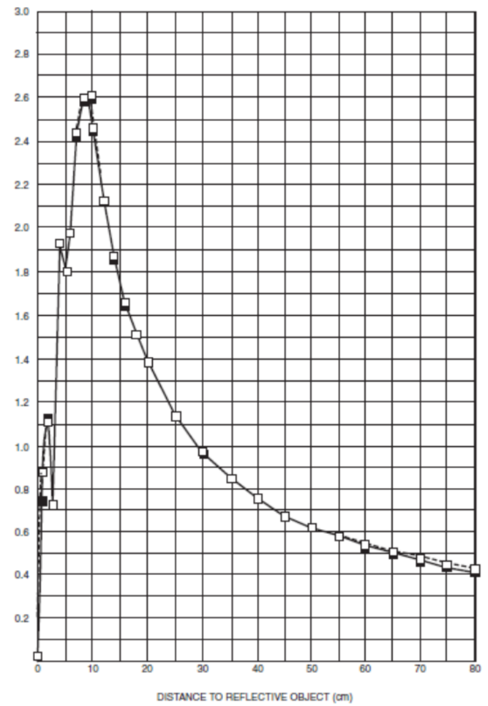
Accuracy

- Sometimes described in terms of the mean of the error (whereas *precision* relates to the variance of error)
- Calibration can remove some but not all inaccuracy. E.g. for a linear sensor there may remain:
  - Uncertainty about the offset
  - Uncertainty about the slope (% error)
  - Uncertainty about deviations from linearity
- Combined with imprecision, inaccuracy may limit the *effective resolution* much more than the output scale
- Non-linearity can make resolution dependent on input

Example: IR sensors on the Khepera

Sensor data sheet

Voltage

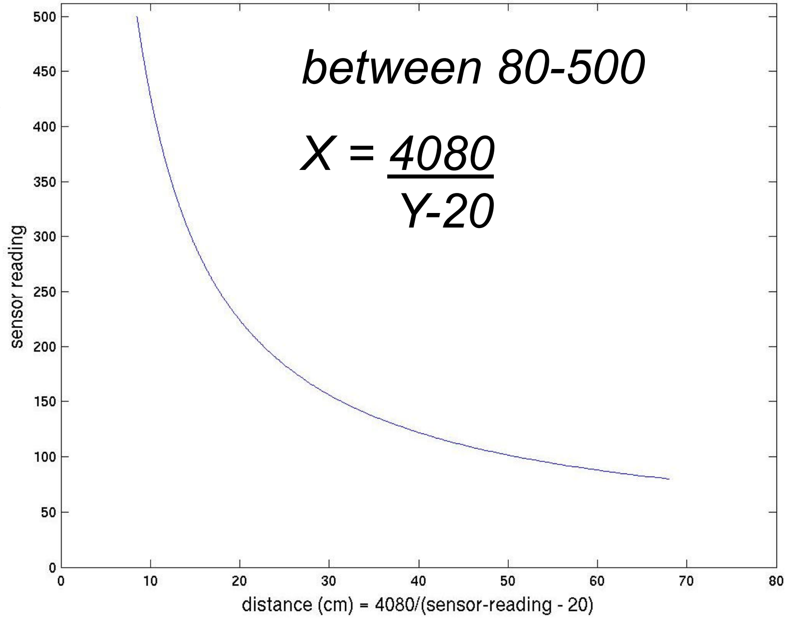


Real distance to object

10 bit Analog to Digital Conversion

Calculation of distance from reading

Sensor reading



between 80-500

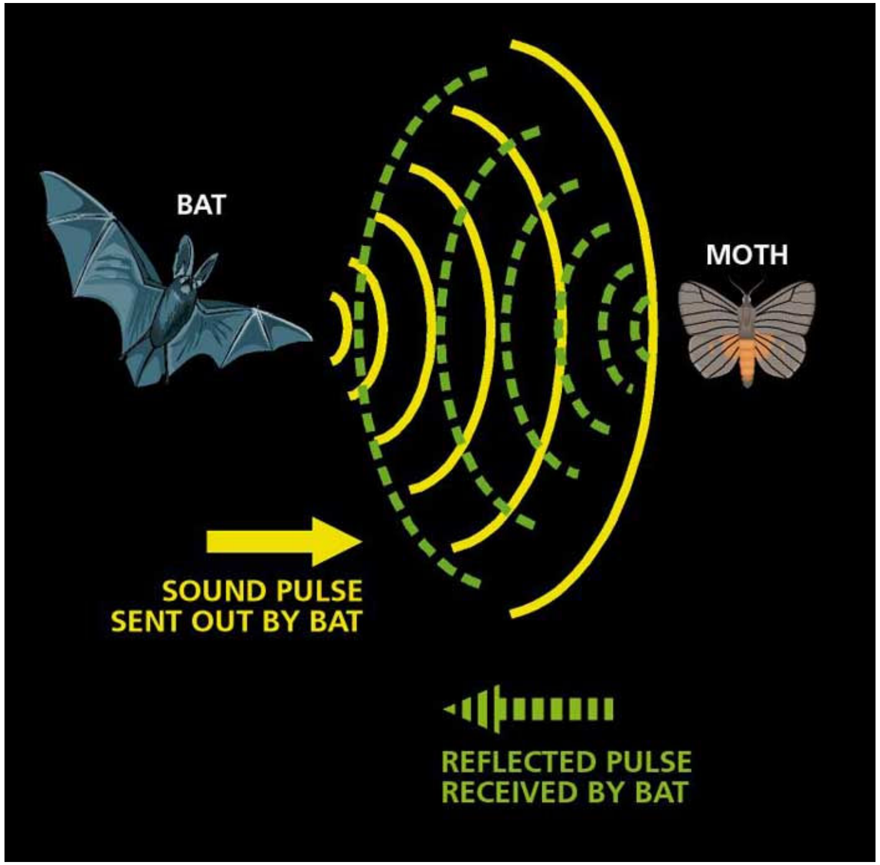
$$X = \frac{4080}{Y-20}$$

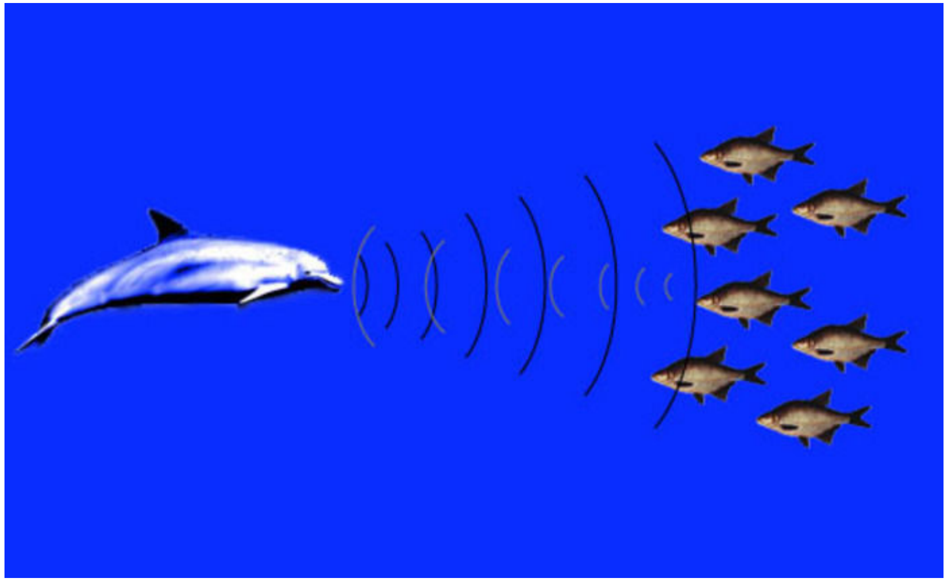
Estimated distance to object

Describing sensors

- Also need to take into account *selectivity:*
  - Inaccuracy is often the result of *cross-sensitivity* to environmental properties other than the target property
  - E.g. many transducers are affected by temperature
  - For IR, the reflectance of the object and ambient light level will alter the 'distance' reading

Sonar


BAT
MOTH
SOUND PULSE
SENT OUT BY BAT
REFLECTED PULSE
RECEIVED BY BAT

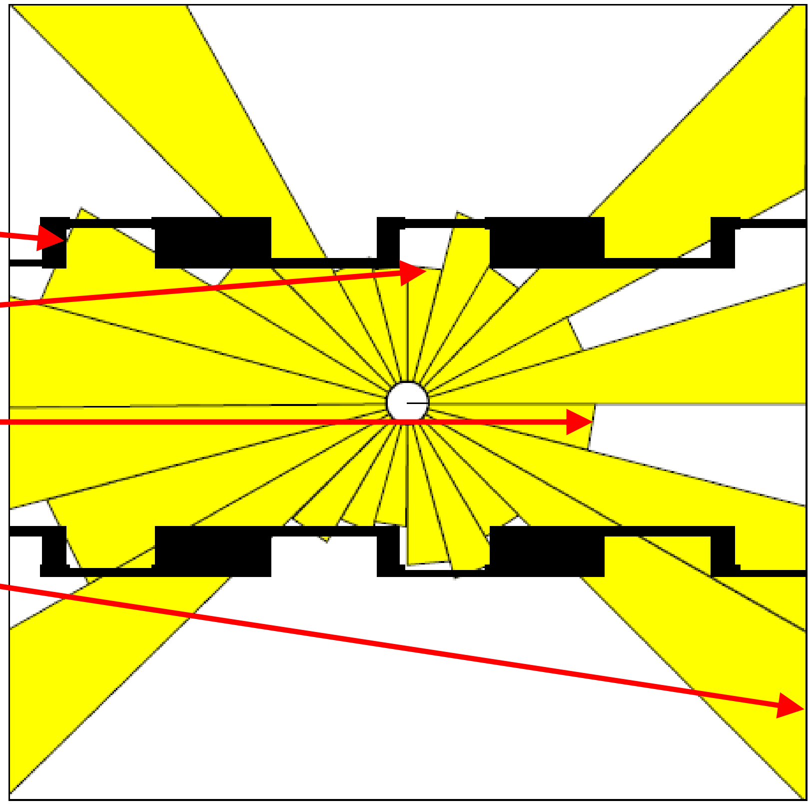


Describing in form of a *sensor model*

- E.g. what is the probability distribution of the sensor reading from a range sensor, given the wall distance?

Possible sources of error:

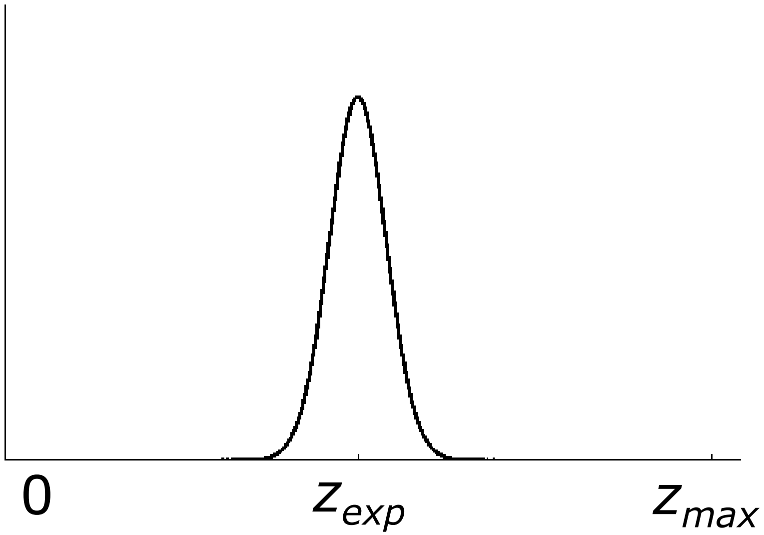
1. Noise in the actual distance measure
2. Person passing
3. Random measurements
4. Maximum range measurements



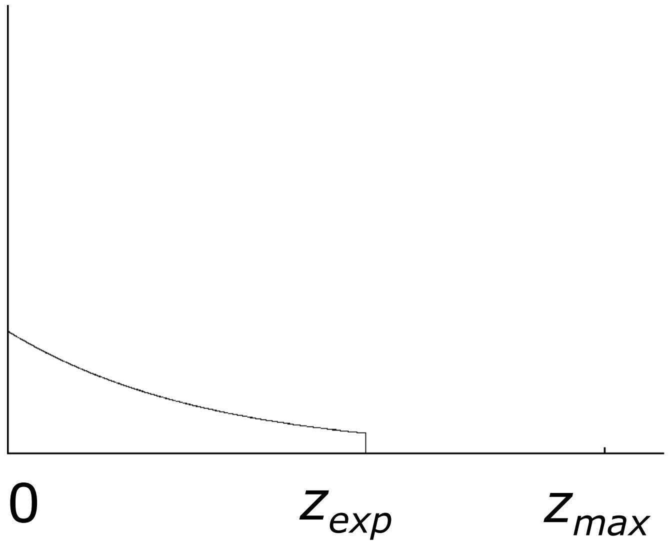
Thrun (2005)

Describe each error source as a distribution

Measurement noise



Unexpected obstacles



$$P_{hit}(z \mid x, m) = \eta \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2}\frac{(z - z_{\text{exp}})^2}{b}}$$

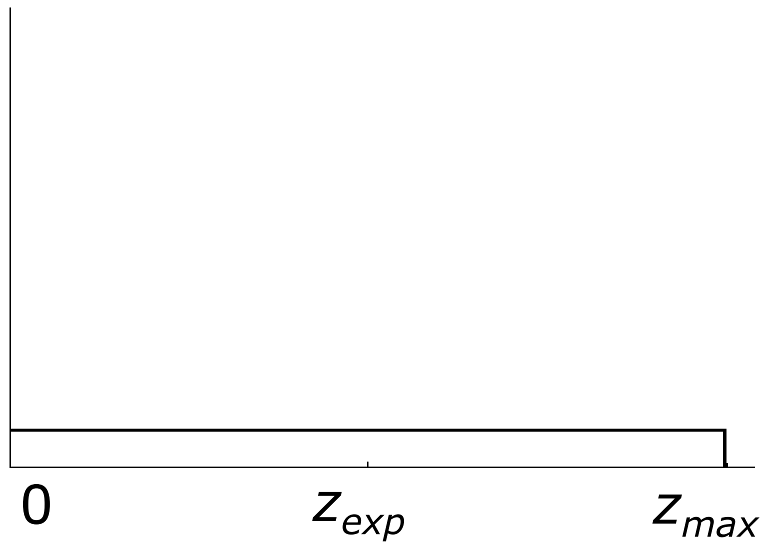
$$P_{\text{unexp}}(z \mid x, m) = \begin{Bmatrix} \eta\ \lambda\ \mathrm{e}^{-\lambda z} & z < z_{\text{exp}} \\ 0 & otherwise \end{Bmatrix}$$

z is sensor signal, x is robot pose, m is world model, $\{x,m\}$ define expected z_{exp}

Thrun (2005)

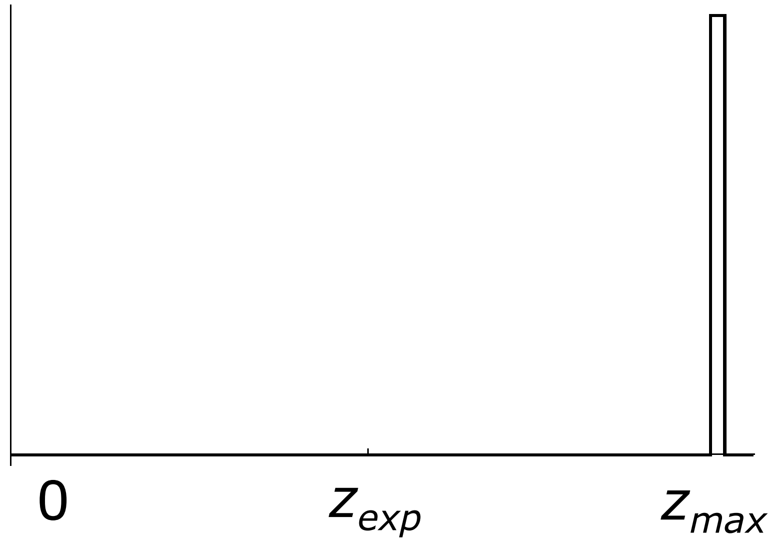
Describe each error source as a distribution

Random measurement



$$P_{rand}(z \mid x, m) = \eta \frac{1}{z_{\max}}$$

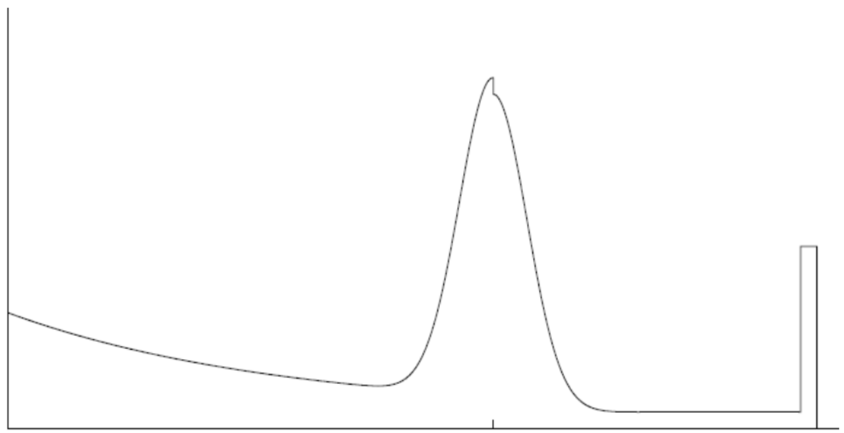
Max range



$$P_{\max}(z \mid x, m) = \begin{cases} 1 \; if \;\; z = z_{\max} \\ 0 \; otherwise \end{cases}$$

Thrun (2005)

Combine as a mixture density



$$P(z \mid x, m) = \begin{pmatrix} \alpha_{\text{hit}} \\ \alpha_{\text{unexp}} \\ \alpha_{\text{max}} \\ \alpha_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} P_{\text{hit}}(z \mid x, m) \\ P_{\text{unexp}}(z \mid x, m) \\ P_{\text{max}}(z \mid x, m) \\ P_{\text{rand}}(z \mid x, m) \end{pmatrix}$$

Note, this seems to assume we know the real distance, z_{exp}

- May be in context of calibration; use to learn parameters α
- Knowing $P(z|z_{exp})$ can be applied (through Bayes theorem) to determine $P(z_{exp}|z) = P(z|z_{exp})P(z_{exp})/P(z)$ (see later lectures)

Thrun (2005)

Note that this is an explicit example of the ‘probabilistic approach’ – handle noise by explicitly representing it.

For next practical –can you devise a sensor model for the Khepera IR sensor?

What alternative approach might there be?

From lecture 1: how to deal with uncertainty

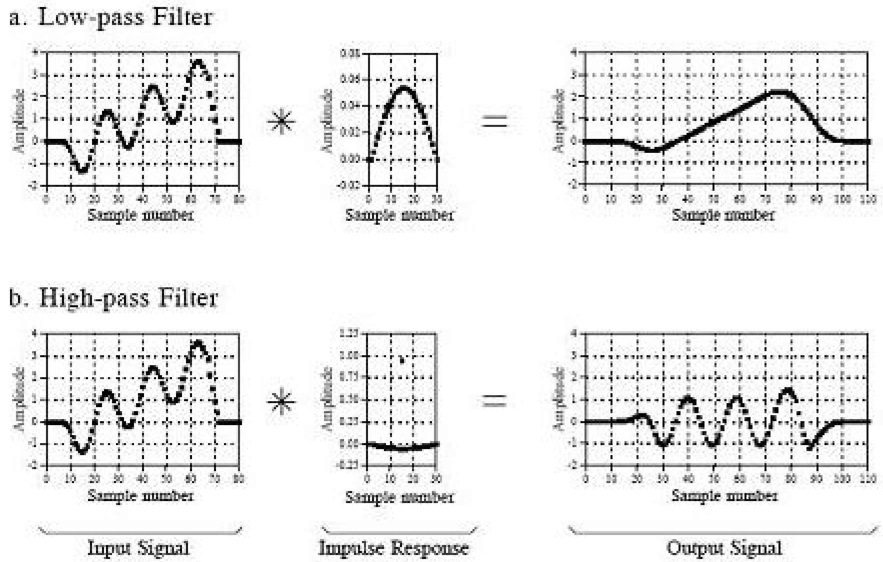
Model-based Principled but brittle	Assume everything is known, or engineer robot or situation so this is approximately true	sense→plan→act
Reactive Robust and cheap but unprincipled	Assume nothing is known, use immediate input for control in multiple tight feedback loops	sense→act sense→act
Hybrid Best and worst of both ?	Plan for ideal world, react to deal with run-time error	plan ⇩ sense→act
Probabilistic Principled, robust but computationally expensive	Explicitly model what is not known	sense→ plan → act with uncertainty

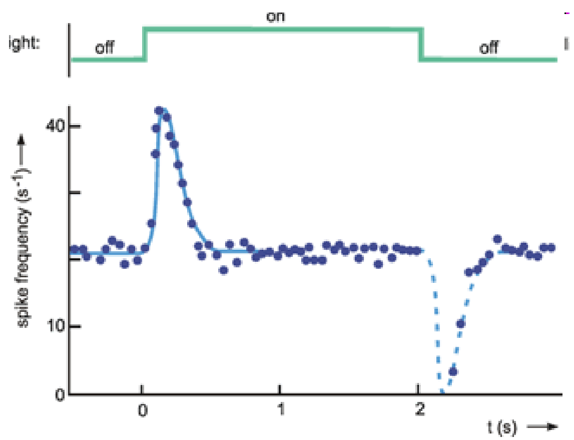
Sensor/signal conditioning

- E.g. linear transformation: $output = offset + gain \times input$
- Many signals may need non-linear transformation
- Might need to tune linear or non-linear parameters through learning methods (again, this can be action-relative)
- 'Intelligence' might be introduced at this level to make sensing adaptive, i.e., sensor/system itself detects:
  - Is the output a reasonable value (e.g. relative to previous measurements or other sensor reports)?
  - Is the full range being used?
  - Is the sensor stuck at one extreme?

Sensor/signal conditioning

- Low-pass filtering:
  - Low-pass: usually against noise or other rapid fluctuations
- Highpass filtering:
  - interested in fluctuations not background





Response of *Limulus* sensory neuron to light

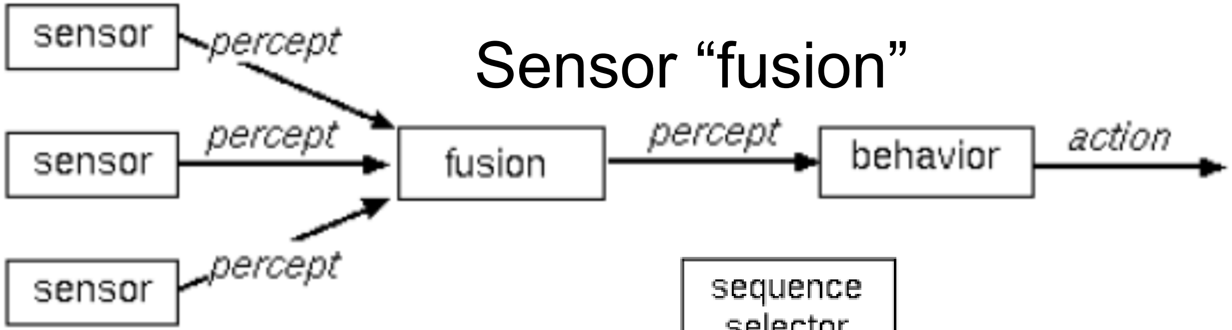
Sensor processing

Implies a more complex transformation than conditioning:

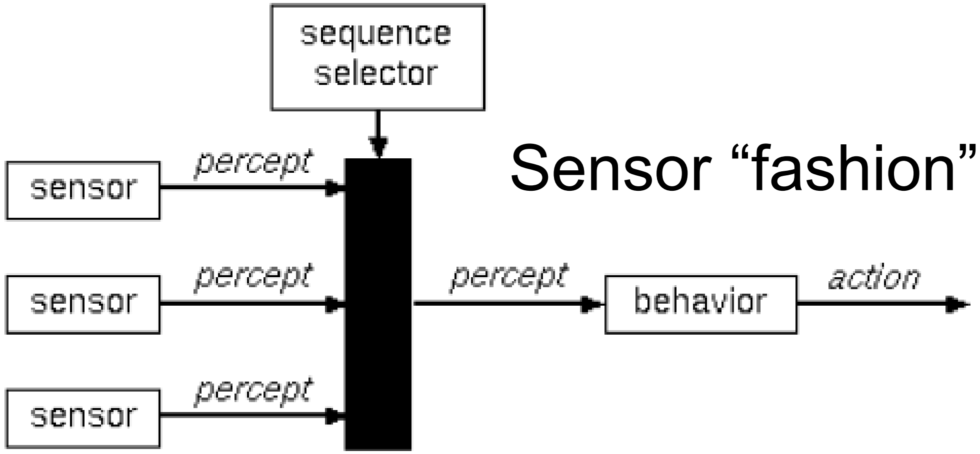
- Logic functions (e.g. triggers for action)
- Data reduction (e.g. extracting features)
- Decision making (e.g. classification)

Combining sensors

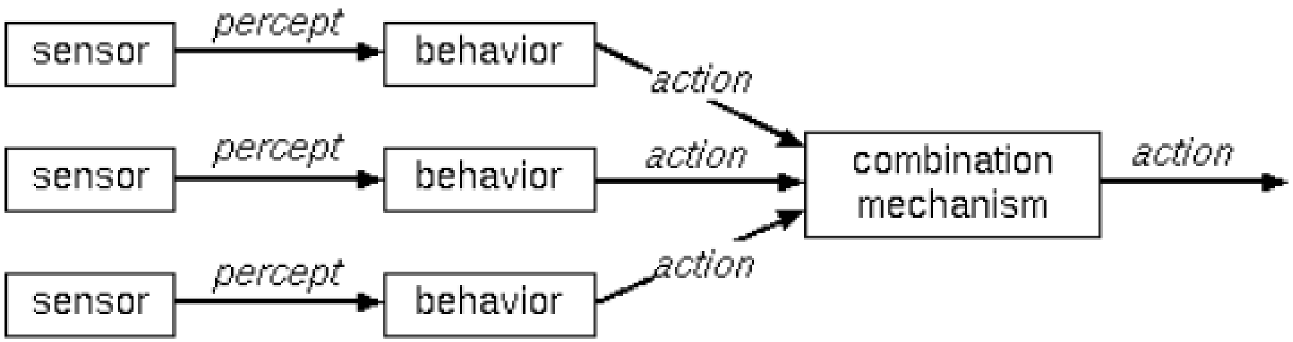
Sensor “fusion”



Sensor “fashion”



Sensor “fission”



Sensor fusion

The information provided by different sensors might be:

- Complementary: sensors that measure different attributes of same target → Fusion could provide richer description
- Co-operative: can derive new feature by combining several attributes (e.g. triangulation) → Fusion could disambiguate
- Competitive/redundant: different sensors that measure the same attribute → Fusion could provide better estimate of actual value

Sensor fusion

A standard approach is to use a weighted average.

Assume N sensors provide measurements *z* of property *x* with some Gaussian distributed noise

$$z_i = x + \varepsilon_i, \varepsilon_i \approx N(0, \sigma_i)$$

Combined estimate is weighted average:

$$\hat{x} = \sum_{i=1}^{N} w_i z_i, \quad \sum_{i=1}^{N} w_i = 1$$

Maximum likelihood estimation says optimal weighting is:

$$w_i = \frac{1/\sigma_i^2}{\sum_{j=1}^{N} 1/\sigma_j^2}$$

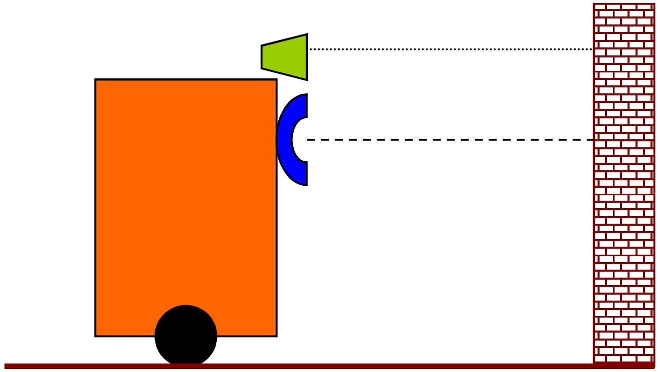
Note there are also adaptive methods that modify the weights over time, e.g. *democratic cue integration*: sensors with values near the combined estimate increase their weights, those further away decrease.

Simple example with two measurements

Robot uses two different sensors to measure distance to a wall:

z_1 with variance σ_1^2

z_2 with variance σ_2^2



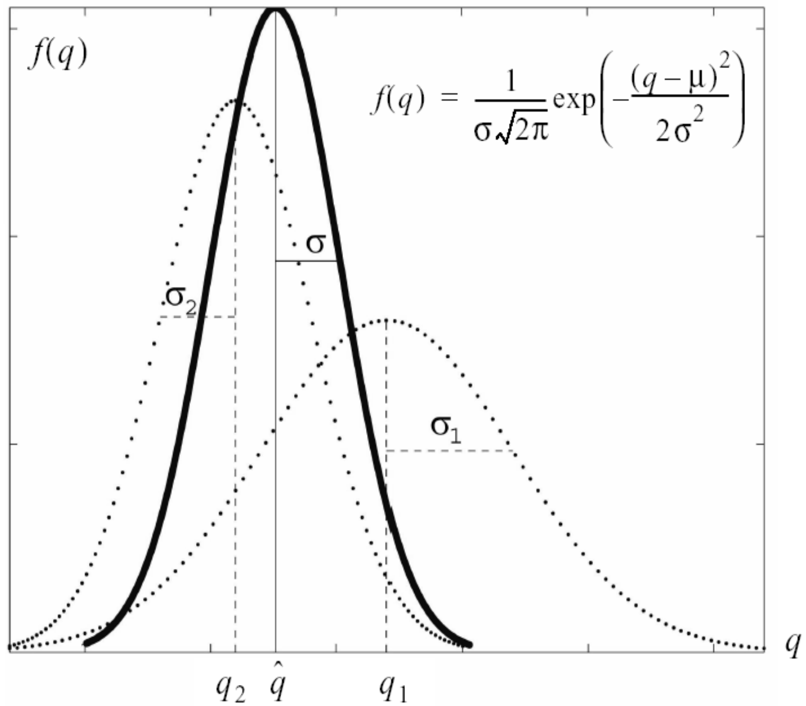
Combined estimate:

$$\hat{x} = \frac{1/\sigma_1^2}{1/\sigma_1^2 + 1/\sigma_2^2} z_1 + \frac{1/\sigma_2^2}{1/\sigma_1^2 + 1/\sigma_2^2} z_2$$

Variance of combined estimate –

$$\sigma^2 = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2}$$

– will be less than that of either single measure



Seigwart & Nourbakhsh, 2004

Can rearrange in terms of successive estimates:

$$\hat{x} = \frac{1/\sigma_1^2}{1/\sigma_1^2 + 1/\sigma_2^2} z_1 + \frac{1/\sigma_2^2}{1/\sigma_1^2 + 1/\sigma_2^2} z_2$$

$$= z_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}(z_2 - z_1)$$

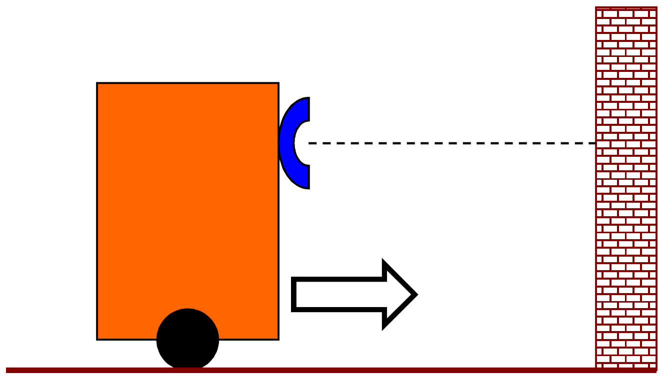
Or in general, updating estimate with the k+1th measurement:

$$\hat{x}_{k+1} = \hat{x}_k + K_{k+1}(z_{k+1} - \hat{x}_k)$$

$$\sigma_{k+1}^2 = \sigma_k^2 - K_{k+1}\sigma_k^2$$

$$K_{k+1} = \frac{\sigma_k^2}{\sigma_k^2 + \sigma_z^2} \quad ; \quad \sigma_k^2 = \sigma_1^2 \quad ; \quad \sigma_z^2 = \sigma_2^2$$

What about updating successive estimates for a moving robot?



Seigwart & Nourbakhsh, 2004

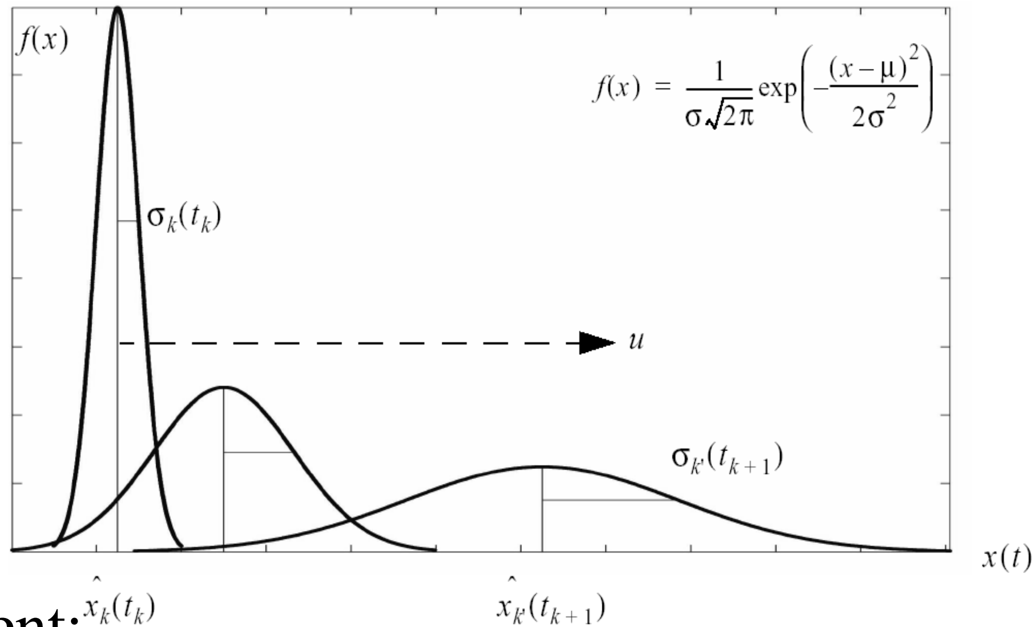
For a moving robot, can first predict the change, then combine this with the new measurement

Robot moves at velocity u, with noise w

$$\frac{dx}{dt} = u + w$$

Our estimate should incorporate the predicted change:

$$\hat{x}_{k'} = \hat{x}_k + u(t_{k+1} - t_k)$$
$$\sigma_{k'}^2 = \sigma_k^2 + \sigma_w^2[t_{k+1} - t_k]$$



Seigwart & Nourbakhsh, 2004

Then update with the measurement:

$$\hat{x}_{k+1} = \hat{x}_{k'} + K_{k+1}(z_{k+1} - \hat{x}_{k'})$$
$$= [\hat{x}_k + u(t_{k+1} - t_k)] + K_{k+1}[z_{k+1} - \hat{x}_k - u(t_{k+1} - t_k)]$$

$$K_{k+1} = \frac{\sigma_{k'}^2}{\sigma_{k'}^2 + \sigma_z^2} = \frac{\sigma_k^2 + \sigma_w^2[t_{k+1} - t_k]}{\sigma_k^2 + \sigma_w^2[t_{k+1} - t_k] + \sigma_z^2}$$

Generalised form of this is the Kalman filter…

References

Shigang Yue · Roger D. Santer · Yoshifumi Yamawaki · F. Claire Rind. Reactive direction control for a mobile robot: a locust-like control of escape direction emerges when a bilateral pair of model locust visual neurons are integrated. *Autonomous Robots* 2009.

Sebastian Thrun, Wolfram Burgard and Dieter Fox, "Probabilistic Robotics", MIT Press, Cambridge MA, 2005

Roland Siegwart & Illah Nourbakshsh "Introduction to Autonomous Robotics" MIT Press, Cambridge MA, 2004 (or 2nd edition, 2011)