

---

# Estimation and Sensor Fusion (Intelligent Autonomous Robotics)

---

**Dr. Subramanian Ramamoorthy**  
**School of Informatics**

# Motivation

Consider the following simple problem:

- You have been gifted a couple of AIBOs
- You want to give them the ability to observe the world, objects and each other through vision
- Basically, you want accurate estimates of distances and object locations
- What will you do?
- Concretely, here are two examples of vision data:

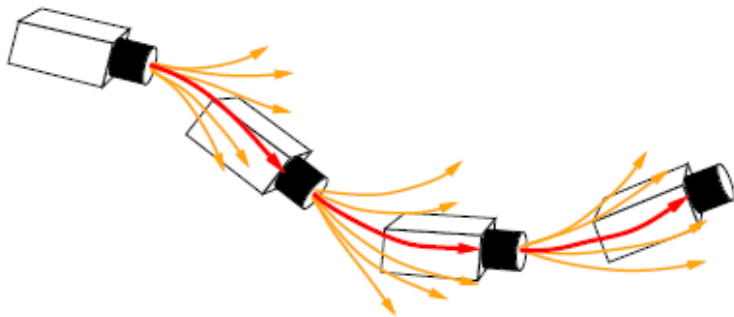
[Clip 1](#)

[Clip 2](#)

# Several Information Needs

**Know thyself – where is the body/sensor?**

**Know thy environment**



---

# Sensors must deal with...

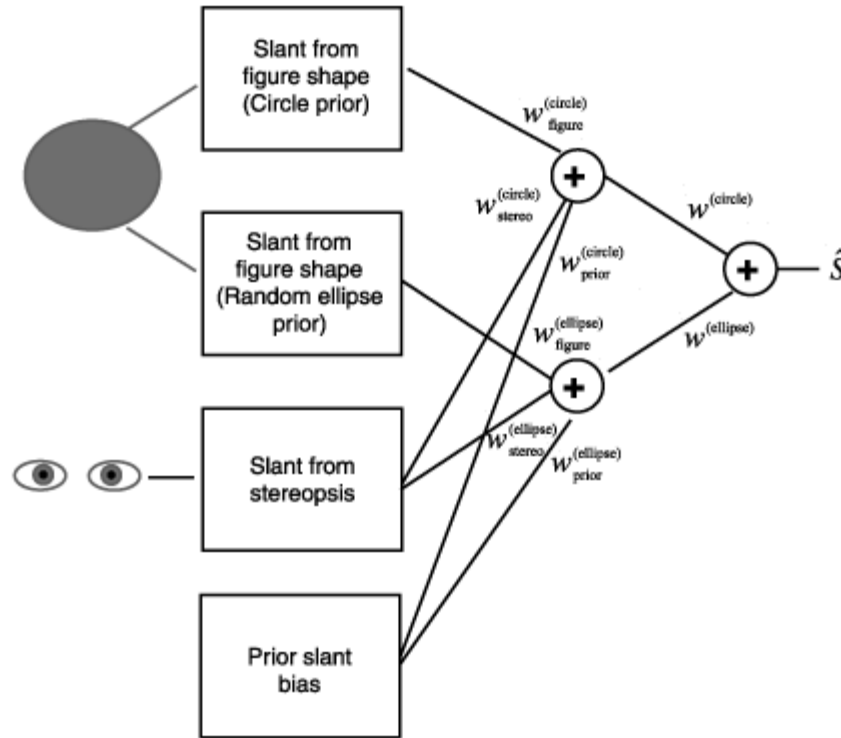
- Ambient light changes
- Colour of various targets
- Orientation of various surfaces
- Scale effects (e.g., how much of the object is in a pixel?) and how does that change with movement?

Two major techniques:

**Estimation** – extracting a “true” model from noise

**Sensor Fusion** – combining “information” coming from multiple sensory modalities

# What are we *Fusing*? An Example.



J. Vision, Vol7(7), Article 5, pp. 1-24

# Estimation: Basic Question

Given,

- a sensor model,  $y = \mathbf{H}x + e$
- $N$  sensor readings in  $y$ , generated by  $y_k = H_k x + e_k; k = 1, 2, \dots, N$

Find the *true* parameters.

Techniques to deal with a number of scenarios:

- Scalar of vector parameters
- Linear/nonlinear, Gaussian noise/other distributions
- Repeated measurements from same setup or more variation in experimental conditions

# Estimation – An Essential Tradeoff

Consider an observation  $y$  that depends on parameter  $\theta$

through a model given by *conditional probability density function(likelihood)*,  $p(y|\theta)$ .

We want an estimator  $\hat{\theta} = \alpha(y)$

If the true parameter value is  $\theta_0$ , we can define mean squared error,

$$MSE(\hat{\theta}) = E[(\theta_0 - \hat{\theta})^2]$$

$$E[(\hat{\theta} - E[\hat{\theta}])^2] + (\theta_0 - E[\hat{\theta}])^2$$

where the first term is the variance and the second term is the bias.

We have to make *tradeoffs*! So, we define some properties to consider ...

# Properties of an Estimator

- *Unbiased* if  $\text{Bias}(\hat{\theta}) = 0$ .
- *Minimum variance (MV)* if  $\hat{\theta}(y) = \arg \min_{\alpha} \text{Var}(\alpha(y))$ . Without constraints on the estimator, the MV property is useless since for instance  $\alpha(y) = 0$  gives  $\text{Var}(\alpha(y)) = 0$ .
- *Minimum variance unbiased (MVU)* principle constrains the estimator to the class of unbiased ones.
- *Best linear unbiased estimator (BLUE)* constrains the estimator to be unbiased and the linear functions  $\hat{\theta} = \alpha(y) = Ly$  that minimizes the MSE.
- *Consistent*, if  $\text{Bias}(\hat{\theta}) \rightarrow 0$  and  $\text{Var}(\hat{\theta}) \rightarrow 0$  as  $N \rightarrow \infty$ .
- *Efficient*, if  $\text{Bias}(\hat{\theta}) = 0$  and  $\text{Var}(\hat{\theta}) \rightarrow \text{CRLB}$  as  $N \rightarrow \infty$ , where CRLB denotes the *Cramér-Rao lower bound*



# Batchwise Least Squares Method

The *least squares* (*LS*) estimate is defined as

$$\hat{x}^{LS} = \arg \min_x \sum_{k=1}^N (y_k - H_k x)^T (y_k - H_k x) = \arg \min_x (\mathbf{y} - \mathbf{H}x)^T (\mathbf{y} - \mathbf{H}x).$$

Direct differentiation and setting the result to zero gives the estimate

$$\begin{aligned} \hat{x}^{LS} &= (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{y} \\ &= \left( \sum_{k=1}^N H_k^T H_k \right)^{-1} \sum_{k=1}^N H_k^T y_k. \end{aligned}$$

# Batchwise Weighted Least Squares

If  $\text{Cov}(e_k) = R_k$  and  $\mathbf{R} = \text{diag}(R_1, \dots, R_N)$ , then the *weighted least squares* estimate (WLS) is defined as

$$\hat{x}^{WLS} = \arg \min_x \sum_{k=1}^N (y_k - H_k x)^T R_k^{-1} (y_k - H_k x) = \arg \min_x (\mathbf{y} - \mathbf{H}x)^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{H}x).$$

with the solution

$$\begin{aligned} \hat{x}^{WLS} &= \left( \underbrace{\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}}_{\mathcal{I}_N} \right)^{-1} \underbrace{\mathbf{H}^T \mathbf{R}^{-1} \mathbf{y}}_{f_N} \\ &= \left( \underbrace{\sum_{k=1}^N H_k^T R_k^{-1} H_k}_{\mathcal{I}_N} \right)^{-1} \underbrace{\sum_{k=1}^N H_k^T R_k^{-1} y_k}_{f_N}. \end{aligned}$$

# Mean and Variance for Simulated Data

If the data are really generated by the model for some  $x^o$ ,

$$\mathbf{y} = \mathbf{H}x^o + \mathbf{e}, \quad \text{Cov}(\mathbf{e}) = \mathbf{R},$$

we have

$$\begin{aligned}\hat{x}^{WLS} &= x^o + (\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{R}^{-1} \mathbf{e}, \\ \hat{x}^{LS} &= x^o + (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{e}\end{aligned}$$

and get the properties

$$\begin{aligned}\mathbb{E}(\hat{x}^{WLS}) &= x^o, \\ \mathbb{E}(\hat{x}^{LS}) &= x^o, \\ \text{Cov}(\hat{x}^{WLS}) &= (\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} \triangleq P^{WLS}, \\ \text{Cov}(\hat{x}^{LS}) &= (\mathbf{H}^T \mathbf{H})^{-1} (\mathbf{H}^T \mathbf{R} \mathbf{H}) (\mathbf{H}^T \mathbf{H})^{-1} \triangleq P^{LS}.\end{aligned}$$

# Properties of Least Squares Estimators

$$\mathbb{E}(\hat{x}^{WLS}) = x^o,$$

$$\mathbb{E}(\hat{x}^{LS}) = x^o,$$

$$\text{Cov}(\hat{x}^{WLS}) = (\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} \triangleq P^{WLS},$$

$$\text{Cov}(\hat{x}^{LS}) = (\mathbf{H}^T \mathbf{H})^{-1} (\mathbf{H}^T \mathbf{R} \mathbf{H}) (\mathbf{H}^T \mathbf{H})^{-1} \triangleq P^{LS}.$$

WLS is BLUE for the linear problem, so we must have  $P^{WLS} < P^{LS}$ , which is perhaps not obvious from the algebraic expressions.

Further, if the distribution of the noise is assumed Gaussian, the estimates become Gaussian as well

$$\begin{aligned} \mathbf{e} \in \mathcal{N}(0, \mathbf{R}) &\Rightarrow \\ \hat{x}^{WLS} &\in \mathcal{N}(x^o, P^{WLS}), \\ \hat{x}^{LS} &\in \mathcal{N}(x^o, P^{LS}) \end{aligned}$$

In the Gaussian case, WLS is also the minimum variance estimator.

---

# Recap, so far.

Estimation lets you

- take data from a noisy sensor
- and, subject to assumptions on noise and system model,
- *infer* the “true” value of the underlying parameters

You can do this for different sensors

And different sensory modalities

As long as they measure the same thing, you can write their measurements in terms of mean and covariance

# Data Fusion

How do you combine two unbiased estimates with arbitrary covariance matrices?

$$\begin{aligned}E(\hat{x}_1) &= E(\hat{x}_2) = x, \\ \text{Cov}(\hat{x}_1) &= P_1, \\ \text{Cov}(\hat{x}_2) &= P_2,\end{aligned}$$

$$\begin{aligned}P &= (P_1^{-1} + P_2^{-1})^{-1}, \\ \hat{x} &= P (P_1^{-1}\hat{x}_1 + P_2^{-1}\hat{x}_2)\end{aligned}$$

# Fusion in terms of “Information”

$$\mathbf{E}(\hat{x}_1) = \mathbf{E}(\hat{x}_2) = x,$$

$$\text{Cov}(\hat{x}_1) = P_1,$$

$$\text{Cov}(\hat{x}_2) = P_2,$$

$$\mathcal{I}_i = P_i^{-1}$$

$$\mathcal{I} = \mathcal{I}_1 + \mathcal{I}_2,$$

$$\hat{x} = \mathcal{I}^{-1} \left( \underbrace{\mathcal{I}_1 \hat{x}_1}_{f_1} + \underbrace{\mathcal{I}_2 \hat{x}_2}_{f_2} \right)$$

The information can thus be seen as a weighting of the estimates. The sensor fusion formula can be extended to more than two terms by recursive application of the formulas above.

# De-Fusion

The reverse operation of *de-fusion* is sometimes needed to get rid of old information that is obsolete, or used multiple times to form a fused estimate. To remove the information  $\mathcal{I}_2$  from  $\mathcal{I}_1$ , apply

$$\begin{aligned}\mathcal{I} &= \mathcal{I}_1 - \mathcal{I}_2, \\ \hat{x} &= \mathcal{I}^{-1} \left( \underbrace{\mathcal{I}_1 \hat{x}_1}_{f_1} - \underbrace{\mathcal{I}_2 \hat{x}_2}_{f_2} \right)\end{aligned}$$



# Sequential Computation of LS Estimates

The least squares loss function can be defined recursively:

$$\begin{aligned} & \sum_{k=1}^N (y_k - H_k \hat{x}_N)^T R_k^{-1} (y_k - H_k \hat{x}_N) \\ &= \sum_{k=1}^N (y_k - H_k \hat{x}_{k-1})^T (H_k P_{k-1} H_k^T + R_k)^{-1} (y_k - H_k \hat{x}_{k-1}) - (\hat{x}_0 - \hat{x}_N)^T P_0^{-1} (\hat{x}_0 - \hat{x}_N) \end{aligned}$$

Then, the estimates can be computed recursively:

$$\begin{aligned} \hat{x}_k &= \hat{x}_{k-1} + P_{k-1} H_k^T (H_k P_{k-1} H_k^T + R_k)^{-1} (y_k - H_k \hat{x}_{k-1}), \\ P_k &= P_{k-1} - P_{k-1} H_k^T (H_k P_{k-1} H_k^T + R_k)^{-1} H_k P_{k-1}. \end{aligned}$$

# Maximum Likelihood Estimation

The *maximum likelihood* (ML) estimate is defined as

$$\hat{x}^{ML} = \arg \max_x p(\mathbf{y}|x)$$

For a linear Gaussian model, the likelihood is given by

$$p(y_{1:N}|x) = \frac{1}{(2\pi)^{Nn_y/2} \prod_{k=1}^N \sqrt{\det(R_k)}} e^{-\frac{1}{2}V^{WLS}(x)}.$$

That is, the ML estimate coincides with the WLS estimate. Further, if there is a true value  $x^o$  of the parameters

$$\hat{x}^{ML} = \hat{x}^{WLS} \in \mathcal{N}(x^o, P).$$

Note the interpretation that the estimate is Gaussian distributed.

It is a general property of the ML estimate that it is asymptotically Gaussian distributed

$$\hat{x}^{ML} \rightarrow \mathcal{N}(x^o, P), \quad N \rightarrow \infty.$$

# Marginalization

Suppose the parameter vector,  $x = (x_1, x_2)$ , consists of two parts, one with the parameters  $x_1$  of interest, and one with *nuisance parameters*  $x_2$ . There are two conceptually different approaches:

1. To eliminate the nuisance parameters by estimation, which leads to maximization of a *generalized likelihood* and the *generalized maximum likelihood estimate (GML)*:

$$\hat{x}_1^{GML} = \arg \max_{x_1} \max_{x_2} p(y_{1:N} | x_1, x_2) = \arg \max_{x_1} p(y_{1:N} | x_1, \hat{x}_2(x_1)).$$

2. To eliminate the nuisance parameters by marginalization, which leads to maximization of a *marginalized likelihood* and the *marginalized maximum likelihood estimate (MML)*:

$$\hat{x}_1^{MML} = \arg \max_{x_1} p(y_{1:N} | x_1) = \arg \max_{x_1} \int p(y_{1:N} | x_1, x_2) p(x_2 | x_1) dx_2$$

The nuisance parameters are here considered to be stochastic with a prior distribution  $p(x_2 | x_1)$  that may depend on  $x_1$ .

# Dealing with Nonlinearities

Consider a toy example: range and bearings sensor.  
How does the noise in what you sense translate into a different coordinate system?

$$y = (r, \varphi)^T = h(x_1, x_2) + e,$$

$$r = \sqrt{x_1^2 + x_2^2} + e_r,$$

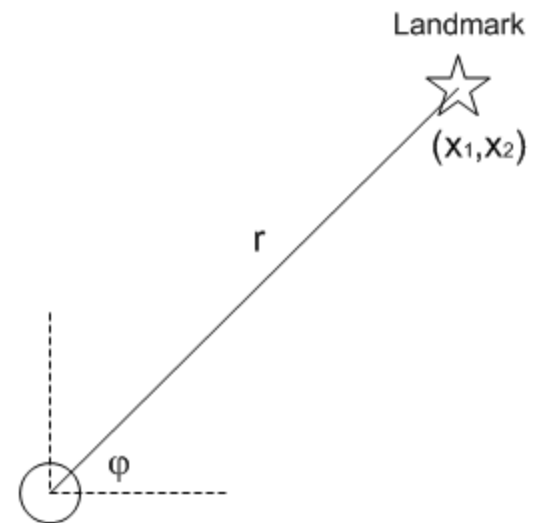
$$\varphi = \arctan(x_2/x_1) + e_\varphi.$$

inverted to

$$x = h^{-1}(y),$$

$$x_1 = y_1 \cos(y_2),$$

$$x_2 = y_1 \sin(y_2).$$



# Estimation by Analytical Approximation

If the inverse  $h^{-1}(y)$  exists, compute  $E(x) = E(h^{-1}(y))$  and  $\text{Cov}(x) = \text{Cov}(h^{-1}(y))$ :

$$\text{Cov}(x) = \frac{\sigma_r^2 - r^2\sigma_\varphi^2}{2} \begin{pmatrix} b + \cos(2\varphi) & \sin(2\varphi) \\ \sin(2\varphi) & b - \cos(2\varphi) \end{pmatrix}$$
$$b = \frac{\sigma_r^2 + r^2\sigma_\varphi^2}{\sigma_r^2 - r^2\sigma_\varphi^2}.$$

# Monte Carlo Estimation

Generate a large number of samples  $y^{(i)}$  from the distribution of  $y$ .

Propagate these through  $h^{-1}(y)$  to get  $x^{(i)}$ .

Compute mean and covariance as follows:

$$\mu_x = \frac{1}{N} \sum_{i=1}^N x^{(i)}$$

$$P_x = \frac{1}{N-1} \sum_{i=1}^N (x^{(i)} - \mu_x)(x^{(i)} - \mu_x)^T$$

# Estimation using an Unscented Transform

The *unscented transform* is a general method to transform a (Gaussian) distribution using nonlinear mappings. In short, it works as follows:

1. The so called *sigma points*  $y^{(i)}$  are computed. These are the mean and symmetric deviations around the mean computed from the covariance matrix of  $y$ .
2. The sigma points are mapped to  $x^{(i)} = h^{-1}(y^{(i)})$ .
3. The mean and covariance are fitted to the mapped sigma points.

---

# Concluding Thoughts

## How might the human brain do/use this?

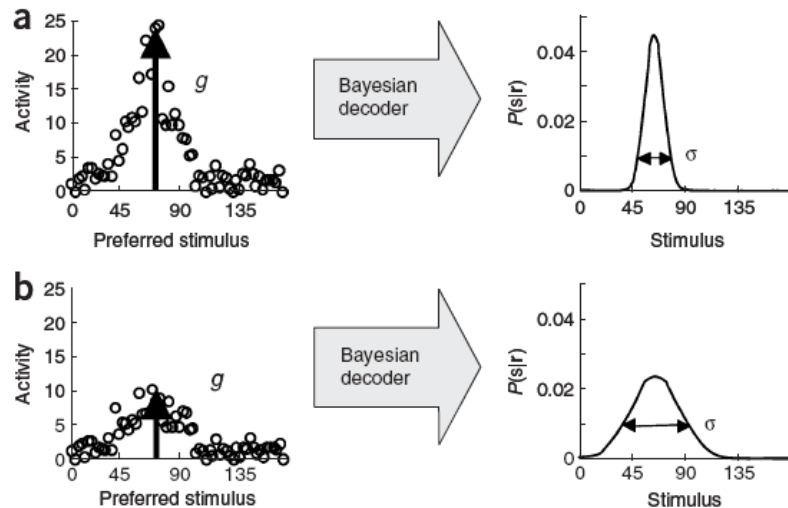
### Bayesian inference with probabilistic population codes

Wei Ji Ma<sup>1,3</sup>, Jeffrey M Beck<sup>1,3</sup>, Peter E Latham<sup>2</sup> & Alexandre Pouget<sup>1</sup>

Recent psychophysical experiments indicate that humans perform near-optimal Bayesian inference in a wide variety of tasks, ranging from cue integration to decision making to motor control. This implies that neurons both represent probability distributions and combine those distributions according to a close approximation to Bayes' rule. At first sight, it would seem that the high variability in the responses of cortical neurons would make it difficult to implement such optimal statistical inference in cortical circuits. We argue that, in fact, this variability implies that populations of neurons automatically represent probability distributions over the stimulus, a type of code we call probabilistic population codes. Moreover, we demonstrate that the Poisson-like variability observed in cortex reduces a broad class of Bayesian inference to simple linear combinations of populations of neural activity. These results hold for arbitrary probability distributions over the stimulus, for tuning curves of arbitrary shape and for realistic neuronal variability.



# Bayesian decoding



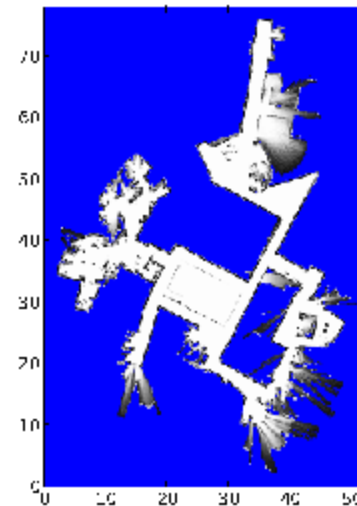
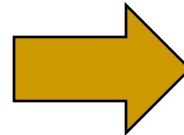
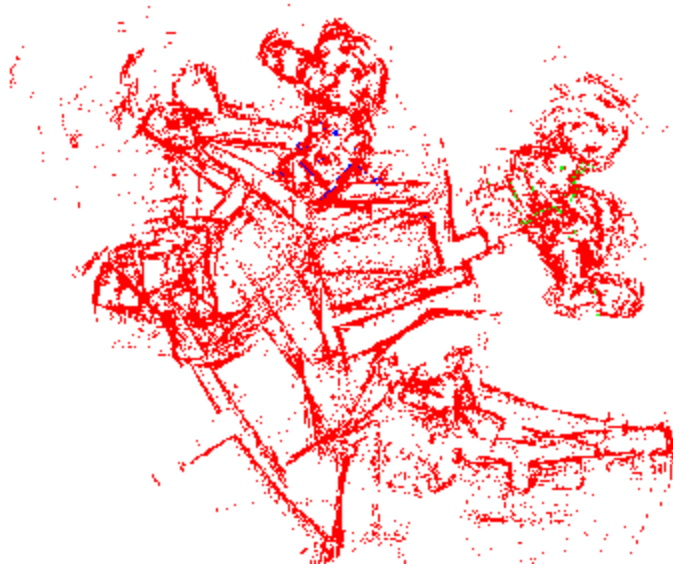
**Figure 1** Certainty and gain. (a) The population activity,  $r$ , on the left is the single trial response to a stimulus whose value was 70. All neurons were assumed to have a translated copy of the same generic Gaussian tuning curve to  $s$ . Neurons are ranked by their preferred stimulus (that is, the stimulus corresponding to the peak of their tuning curve). The plot on the right shows the posterior probability distribution over  $s$  given  $r$ , as recovered using Bayes' theorem (equation (1)). When the neural variability follows an independent Poisson distribution (which is the case here), it is easy to show that the gain,  $g$ , of the population code (its overall amplitude) is inversely proportional to the variance of the posterior distribution,  $\sigma^2$ . (b) Decreasing the gain increases the width of the encoded distribution. Note that the population activity in **a** and **b** have the same widths; only their amplitudes are different.

... and then the paper goes on to show that humans can reliably compute using such codes (e.g., in cue integration). The takeaway point is that if properly handles, many channels of uncertain data can be yield pretty good results.

# Upcoming...

- This lecture only dealt with compensating sensor noise and combining the resulting estimates
- What happens when we want to take dynamics into account?
  - Kalman Filtering – linear theory that extends the recursive update calculations mentioned here
  - Nonlinear versions – Extended/unscented Kalman filters, Particle filters, etc.
- What happens when you don't have landmarks to work with?
  - *Simultaneous* Localization and Mapping (SLAM)

# Upcoming...



---

## Reference:

S. Thrun, W. Burgard and D. Fox,  
Probabilistic Robotics, MIT Press, 2005.

Many equations in this presentation are from:

F. Gustafsson, Sensor Fusion, Course Notes  
2007.