Space, time and objects

Rick Grush
Department of Philosophy, UC San Diego

1. Introduction

In this paper I will outline a unified information processing framework whose goal is to explain how the nervous system represents space, time and objects. In the remainder of this introductory section I will first be more specific about the sort of spatial, temporal, and object representation at issue, and then outline the structure of this paper.

It is standard procedure to distinguish different kinds of spatial representation, and the most basic distinction is between *allocentric* (or *objective*) and egocentric spatial representation.[1] The idea is that egocentric spatial representation is how an organism represents its environment for purposes of perception and action. The directions involved in egocentric spatial representation are *above, ahead, to the left*, and so forth. Allocentric or objective spatial representation lacks any explicit reference to the organism itself: when I grasp the thought that the Arch d'Triomphe is between the Obelisk and the Grand Arch de la Defense, my spatial representation is in terms of objective spatial relations and perhaps objective units of magnitude as well. I will be concerned in this paper exclusively with the egocentric variety of spatial representation, but in order to avoid baggage connected to that expression in the literature, I will prefer the expression *behavioral space* to *egocentric space*.

[1] For more distinctions see Grush 2000.

A similar distinction can be made in the case of temporal representation. When I conceive of World War II as being after World War I, I am not thinking in terms of the temporal relation of those events to my current thought. The magnitudes and directions (so to speak) are independent of my current temporal location. By contrast, when I think that the light is about to turn red, or that I went through the door just a moment ago, neither the units nor directions (past versus future) are independent of my current temporal location and capacities. I may have a very good idea of when the light will turn red, but a very poor capacity to specify this in objective units, such as milliseconds. I will refer to the sort of time I am interested in as *behavioral time*: it is the time in whose terms the content of our current perception and action is given.

And finally, objects can be conceived of objectively, as things not tied to my current perception of, or action on, them. But it is also possible to grasp objects as entities in my behavioral field, as things upon which I can perceive and act. I will likewise be concerned with the representation of such *behavioral objects*.

Section 2 is a very brief introduction to the emulation theory of representation (more detail can be found in Grush 2004). This theory holds that the nervous system constructs and uses models of the body and environment in order to represent them in perception, action and imagery. It is important to note that the emulation theory itself is silent on *how* this internal model is implemented, and silent also on what sorts of things can be represented. It is a general architecture that describes how an internal model, or emulator, can be comported within a larger system to play certain kinds of roles in perception, imagery, off-line planning, and so forth. In this paper, I will provide some details as to how the emulation theory can be applied to the specific cases of behavioral space, behavioral time, and behavioral objects, and the specific way in which this emulator is implemented neurally.

In Section Three I briefly describe an extension of the emulation framework for temporal representation (more detail can be found in Grush 2005). This is an extension of the emulation theory from an internal model that, at any given point in time, represents the target domain as it is at a point in time, to an internal model that represents, at any given point in time, the behavior of the target domain over a temporal interval.

In Section Four I discuss spatial representation, and in particular Alaxandre Pouget's basis function model of spatial representation (see e.g. Pouget et al. 2002). This is a specific proposal about the neural implementation of spatial representation, in particular of behavioral spatial representation.[2] In brief, the model has it that the locations of objects are represented as a set of basis function values of the sensory and postural signals involved in the perceptual episode.

In Section Five I describe how to combine the basis function model of spatial representation with the trajectory emulation model of temporal representation to yield an information processing framework that genuinely represents behavioral spatiotemporal trajectories of behavioral objects. Section Six concludes.

2. Emulation theory

In this section I will introduce a theory of the information processing structure that the brain employs in order to construct and use *representations* of entities external to the brain, especially the body and the environment. This introduction will be quite brief, and will include neither much of the supporting evidence to the effect that this is in fact the

[2] Though on my preferred use of the expression 'representation' the basis function model by itself is not a theory of representation at all, but a theory of the information processing that extracts spatial information from various signals. It becomes representational, on my account, when operating within the superstructure of the emulation theory, as Section 5 will detail.

information processing structure that the brain uses, nor many of the applications of the theory that supply tremendous explanatory leverage in the attempt to understand the brain's operation. Much of this omitted material can be found in Grush (2004). Readers familiar with the emulation theory can safely skip sections 2.1 and 2.3, though I would still recommend section 2.2. If I am right, the brain has and uses *many* emulators, and I will describe a number of them. There are emulators of the body that are used for various motor control purposes and to produce motor imagery; emulators of the visual scene that produce anticipations of what will be seen, and can be used to produce visual imagery; amodal emulators of the environment that maintain representations of what is happening in the immediate vicinity. Many of these can be run in parallel, and in fact as we shall see, the operation of an emulator of the body and of the sensory modalities being run in parallel is arguably a big part of the explanation of our ability to represent egocentric space in an amodal way.

I should point out that in claiming that this is a way to understand representation, I am simultaneously trying to explain a phenomenon and define my use of terms. Other people may have some idea of what *representations* are that does not jibe with what I am about to say. That is fine. I am not interested in fighting over who gets to use the word 'representation' but rather in understanding a certain kind of phenomenon – the capacity of a sophisticated system to construct and maintain 'internal' states that track the behavior of other entities in order to assist it in its interactions with these other entities. It seems to me that the word 'representation' and its cognates are a natural fit for this phenomenon, and if one wants to couch the issue in terms of *providing an account of what representation is*, or something like that, then what I am about to do, and the 'representation' terminology I use in doing it, strikes me as reasonable and natural. But anyone who has their own axe to grind about the word 'representation' is invited to provide their own terminology for what I am about to describe.

2.1 Control, filtering, and emulation basics

The most basic division in control theory is between the thing doing the controlling, and the thing being controlled. I will typically call the first the *controller*, and the latter the *process*.[3] And there are two basic kinds of classical control architectures that describe how the controller influences the process: open-loop (aka feed-forward) and closed-loop (aka feedback) control. These are shown in Figure 1:

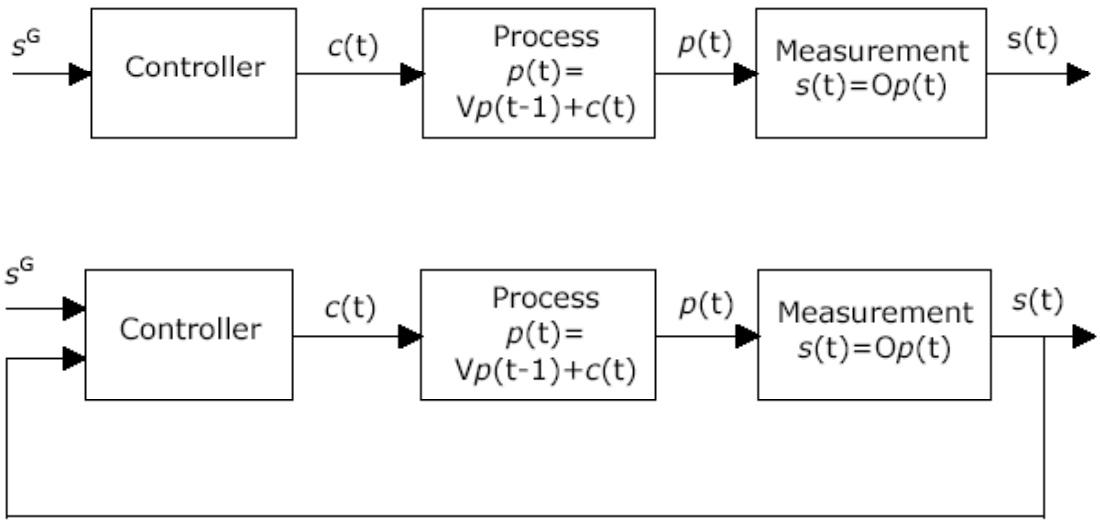

Figure 1. Open-loop and closed-loop control.

The controller is provided with some goal state — the state that the process should be in. This goal is most often specified in terms of some measured state or states of the process. For example, a thermostat is given as its goal a temperature for the room or building, and this is measured by a sensor – a thermometer – that is sensitive to that state of the

[3] 'Controller' is fairly standard terminology for the thing issuing commands. The controlled system is often called the 'plant' in the literature, but 'process' is also used, especially in signal processing contexts.

process. An autopilot controls an aircraft, and its goals are in terms of an altitude, heading, speed — all of which are measured by various instruments on the aircraft. There are of course many states of the room or aircraft that are not measured, and some of these may be important for the operation of the process. The mass of the aircraft is not measured, nor is the force applied to the wing by the engine. *Measured* states will be called sensor signals, since they are typically produced by sensors that monitor or measure some state(s) of the process. The goal state then is a target sensor signal, which I shall call s^G.[4]

The controller's job is to produce a control signal, or sequence of control signals, that will, when issued to the process, cause the process to go into the goal state (as determined by the measured sensor signals). The controller can thus be described as a function that maps a specification of the sensory goal into control signals. The process is some system such that its state at any given time *t* is a function of (at least) two factors: its state at the previous time *t-1*, and the current control signal *c*(t). If we let *V* stand for the function that determines how the state of the process would evolve over time in absence of any external influence, we have: $p(t) = Vp(t\text{-}1) + c(t)$.

The measurement is just some mechanism that measures one or more states of the process in order to produce a sensor signal. If we use the label *s*(t) for the sensor signal that is produced at any given time *t*, we have: $s(t) = Op(t)$, where O is the measurement function. Notice that the sensory goal s^G does not change over time, like the sensory state *s*(t) and process state *p*(t) do.

[4] In some cases my notation and terminology will differ slightly from the standard notation and terminology used in control theory or signal processing. These divergences aren't too frequent, and when they occur the reason will typically be that slightly non-standard terminology will make my overall goal of putting these tools to use in understanding neural information processing easier and more perspicuous. The vast bulk of my terminology and notation is entirely standard. Apologies to those readers who are surprised by the 2 or 3 deviations in notation, but such a reader should have no problem knowing what I am talking about: for example, that what I have called s^G is standardly called a *setpoint* or *reference signal*.

An open-loop controller is one that determines the control signals without benefit of information about the state of the process as the control episode unfolds. A closed-loop controller is one that has the benefit of a continual stream of feedback from the process (in the form of the sensor signal) to help determine the control signals. A thermostat is the standard example of a closed-loop controller. The process in this case is the room or building together with its heating system. The measurement is a measurement of one crucial state of the process, its temperature. The goal state is a goal temperature for the room or building, and this is often input to the controller by moving a lever along a calibrated set of marks. The controller compares the goal temperature with the actual temperature — which it has access to via a measurement of the process. In the simplest case this measurement is implemented by the angle of a bi-metallic strip. Based on this comparison, the controller either turns (or keeps) the heater on, or turns (or keeps) the heater off. The result of the controller's operation is that it produces a sequence of signals, typically in the form of electrical signals sent to the heating system, whose effect is to get the process to the goal state (as determined by the sensor signal).

An example of an open loop controller is an old-fashioned toaster. The process is the heating elements and bread. The controller is simply a timer that keeps the heating elements on for some period of time (this is as least one way toasters have been implemented, there are others). As with the thermostat, one sets the goal state by moving a lever along a calibrated scale, typically ranging from 'light' to 'dark'. On the basis of this input, the controller produces a control sequence and issues it to the process, in the form of keeping a circuit that powers the heating elements closed for some period of time. Unlike the thermostat, the controller gets no feedback concerning how the process develops during the control episode. The control sequence is determined on the exclusive basis of the input of the desired goal state. If everything works well, when this control signal is acted on by

the process, the effect is that the bread is toasted to the degree specified by the desired goal that was input.

My purpose in discussing control theory is not to discuss thermostats or toasters. It is to understand the operation of the brain. In the case of motor control (the most obvious application of control theory in the brain), much of the debate on this topic in the 20th century was a debate between proponents of closed-loop and those of open-loop control as a model for human motor control. In the closed-loop camp (which was heavily influenced by the cyberneticist, themselves largely closed-loop control endorsers whether they used that terminology or not) the motor control centers know the current state of the body and the goal state, and issue commands that will reduce the distance between the two until the difference is zero. This is standard closed-loop thinking.

The open-loop proponents, driven by data that suggested that the initial stages of a given movement appear to be the same whether or not the feedback signal is tampered with, held that the initial stages of a movement are open-loop — a motor volley determined and executed without employing any feedback. One standard way to tamper with the feedback signal is tendon vibration. The two main mechanisms by which the nervous system gets information about the position of the body are stretch receptors, which are responsive to muscle length, and Golgi tendon organs, which are sensitive to tendon tension. Appropriate vibration at muscle/tendon interface can stimulate stretch receptors, fouling the feedback they provide. The finding is that this interference appears to make a difference to the motor control episode only towards the very end of the movement, earlier stages being largely unaffected. This leads these researchers to suggest that motor control is closed-loop only at the very end of the movement (grasping and pointing are typical examples) when fine adjustments are needed. For more on this debate, see Desmurget and Grafton (2000).

Closed-loop and open-loop control, though the most well-known kinds of control scheme, do not exhaust the possibilities. In this subsection and the next I will discuss a few additional schemes, all of which make use of a construct that has not yet been introduced, the *forward model* (which I will also call an *emulator*). The simplest scheme to use a forward model is pseudo-closed loop control, which is shown in Figure 2. In this scheme we have a controller and a process, as in the closed- and open-loop schemes. In addition there is an entity that implements the same, or close to the same, input-output mapping as the process. It is a model of the forward mapping.

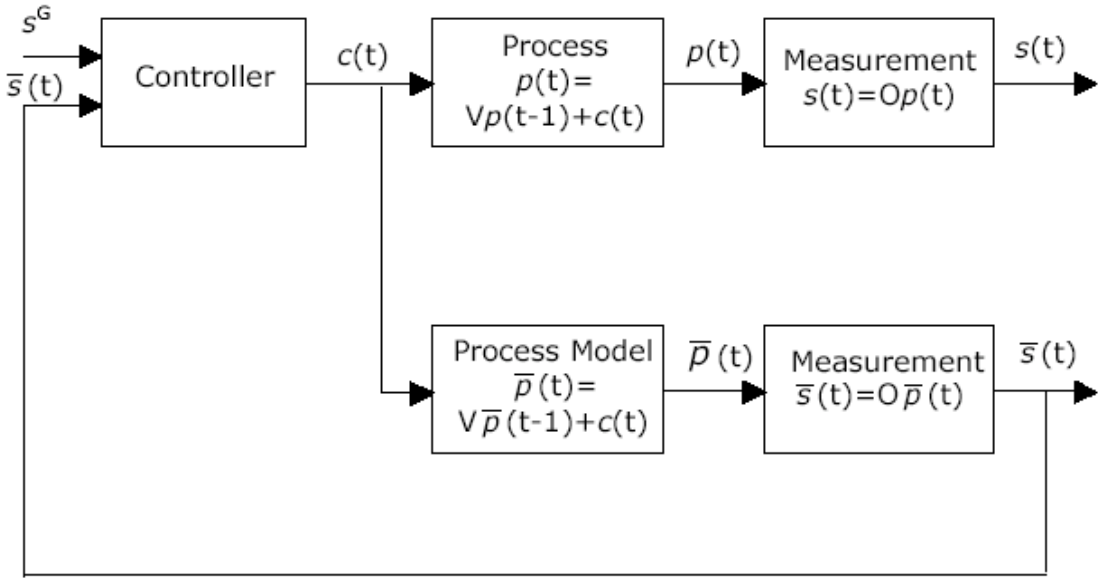

Figure 2. Pseudo-closed loop control.

In this scheme, the control signal *c*(t) is split into two copies, one of which goes to the process as in closed- and open-loop schemes. The other copy is sent to the emulator (which here is a *model* of the process plus a model of the measurement). The process model models the states of the process, and also models the way that those states evolve over

time. While the real process has a state p(t) at any given time, the model has an *estimate* of that state, $\overline{p}$(t). Just as the real process evolves over time as a function of its previous state p(t-1) and the control signal, so the process model's state evolves over time as a function of its previous state $\overline{p}$(t-1) and the control signal. The process model is subjected to a measurement that produces a mock sensory signal $\overline{s}$(t).

Since the emulator (process model + measurement) implements the same input-output mapping as the real process + measurement, and since it is being given the same input, it will produce the same output: that is $\overline{s}$(t) will be equal to s(t). Because the output is the same, the emulator's feedback $\overline{s}$(t) can be provided to the controller in lieu of the feedback produced by the measurement of the process. (These are ideal conditions of course, and complexities such as noise and random disturbances will be introduced shortly.)

There are many uses for process models, including helping the controller deal with delayed feedback from the real process; running the emulator only, with the process completely idle, in order to test out counterfactuals, to see what the process would do if certain control signals were issued to it; running the emulator and process in parallel and using the emulator's feedback to fill in or correct faulty or noisy sensor information from the measurement of the process. This last use requires some more sophisticated mechanisms, to which I turn now.

The Kalman filter (Kalman 1960; Kalman and Bucy 1961; henceforth *KF*) *per se* is, as the name implies, a method of filtering noise from a signal. As such it is not necessarily involved in control structures. I will first describe one standard version of the *KF*, and then explain how it can be integrated into a control structure. For convenience I will treat time as discrete, but extensions of *KF*s to continuous time are available, just more complicated. The top third of Figure 3 exhibits the problem that the *KF* solves. A process evolves over time, in

part as a function of its own inner dynamic, but possibly also under the influence of some external driving force, and perhaps even random disturbances.

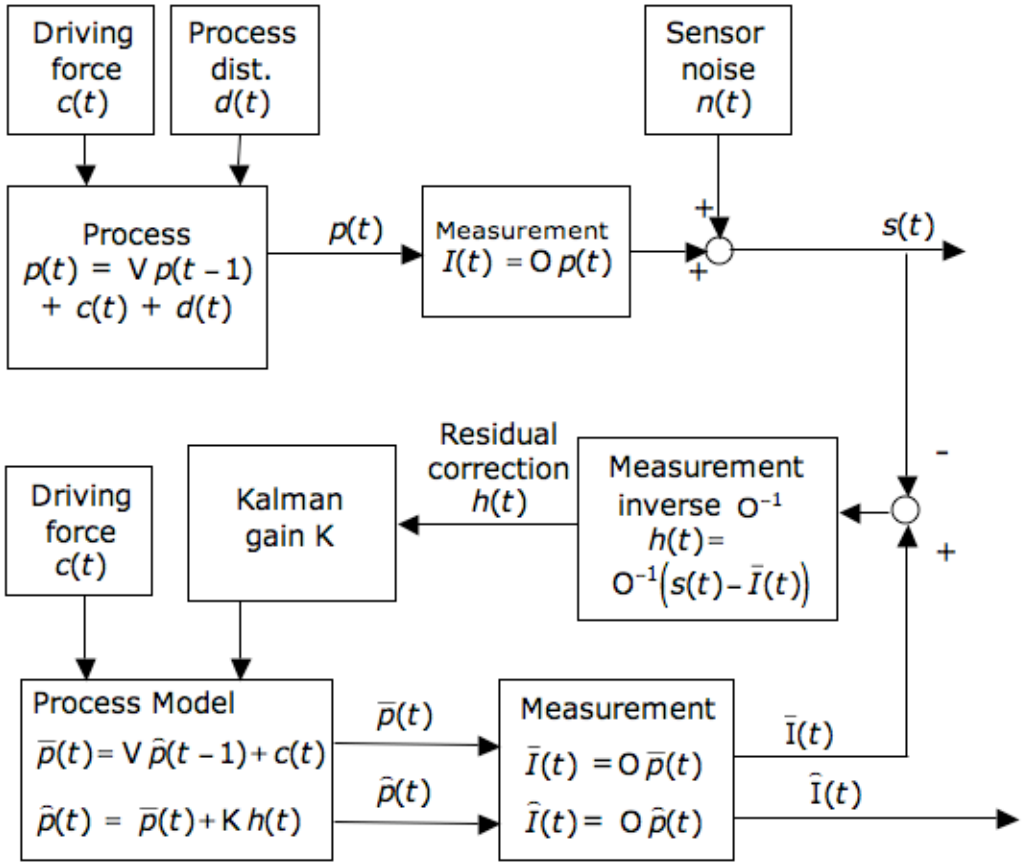

Figure 3. A Kalman filter.

At each time *t* the process is measured to produce a signal *I*(t). But this measurement process is not perfect, and the imperfection can be represented as the addition of noise *n(t)* to *I(t)*. The *observed* (noisy) signal is *s(t)* = *I(t)* + *n(t)*. So we have made the *process* as described in the previous section more realistic by accommodating the possibility of unpredictable disturbances to the process's state, as well as noise in the measurement of the process.

The *KF*'s job is to determine what the real signal *I(t)* is, or to put it another way, to filter the noise from the observed signal. (And if the measurement function *O* is invertible, then this is equivalent to knowing the process's actual state.) The *KF* itself is diagrammed on the lower two-thirds of Figure 3. The KF has knowledge of the function *V* that governs the evolution of the process, as well as the measurement function that produces signals based on the process's state. It also knows, at each time step, what the observed signal is, and what the driving force, if any, is. What it does *not* know is the state of the process, nor the process noise, nor the sensor noise.

This description may sound like it applies only to artificial situations, but in fact this is characteristic of many real world situations, such as ship navigation. The navigation team knows the driving force, if any (we can suppose that they simply listen in as the captain is giving orders), and they know the observed signals – the noisy measurements provided by the bearing takers, etc. And they have knowledge of how a ship's state evolves over time. For example, they know basic stuff such as: if a ship is at location X and is going at speed S and direction D, then at the next time it will be at Y. The team also knows how the measurement process works, in that they know how to translate from ship state to bearing measurements and vice versa (this is just basic trigonometry). They do not know how inaccurate the bearing measurements are (sensor noise); nor do they know the process disturbance – unknown winds and ocean currents, for example. And the navigation team's job is to determine the ship's actual state, which, given the capacity to translate between process state and measurements, is equivalent to determining what perfectly accurate noise-free measurement would be.

There are many situations other than ship navigation that fit this general structure as well. A relevant example is the brain, which tries to keep an accurate estimate of the body's

state. It has knowledge of motor commands that it has sent (it knows the driving force); it has a bunch of sensor information that is noisy and imperfect in various ways; it has, through experience, knowledge of how the body's state generally evolves over time; and on the basis of these sources of information it tries to determine what is really going on with the body.

Back to an explanation of the KF. The main trick of the KF is that it uses its various sources of information to maintain an optimal estimate of the state of the process. It then subjects this process state estimate to a measurement that is just like the real measurement of the real process that produces the real noise-free signal. The result is an optimal estimate of the real noise-free signal. So the question now is: how does the KF get and maintain an optimal estimate of the state of the process?

The KF begins each cycle with the state estimate it produced at the previous cycle. After I have explained the KF's operation I will return to the question of how the KF first gets a workable estimate without any previous estimate. This turns out to be relatively trivial. For now, though, it will be easiest to assume that it has a decent estimate from the *previous* cycle to begin with. The first thing the KF does is to produce an *a priori* estimate of what the current process state should be: $\overline{p}(t)$. This estimate is *a priori* in that it has not yet taken into account any of the information from the observed signal. The KF arrives at $\overline{p}(t)$ by taking its previous state estimate and the current driving force and applying its knowledge of how the process's state evolves over time, *V*. So for example the navigation team can take its estimate of the ship's state at the previous fix cycle, together with the commands that have been issued by the captain concerning engine speed and rudder angle, to produce an *a priori* estimate as to what the ship's current state ought to be. This cycle of the KF's operation is sometimes called the *time update*.

The second cycle, sometimes called the *measurement update*, of operation is where the KF adjusts its *a priori* estimate on the basis of the observed signal. The result of this adjustment is the *a posteriori* estimate, $\hat{p}(t)$, and this is the KF's final estimate of the state of the process at that time. This cycle unfolds as follows. First, the *a priori* state estimate $\bar{p}(t)$ is measured to produce an *a priori* estimate of the real signal $\bar{I}(t)$. (I will consistently use a straight bar over a variable in order to indicate an *a priori* estimate of that variable's value.[5]) This is what the KF expects the observed signal to be, given its *a priori* state estimate and current driving force. This is compared to the actual observed signal *s*(t). The difference is the mismatch between what the KF expected to see, given its *a priori* estimate, and what it actually saw from the observed signal. By pushing this difference through an inverse of the measurement function[6], the *KF* arrives at a measure of the difference between its own *a priori* estimate of the process's state, and what the process's state would have to be if the observed signal were accurate. This is called the *sensory residual*.

The tricky part is that the sensor signal is not perfect. There is noise, and in the navigation example this is a reflection of the fact that the people taking the bearings are not completely accurate. Because of this, one cannot assume that the existence of a non-zero sensory residual means that the *a priori* estimate was inaccurate. Even if the *a priori* estimate is entirely accurate, the existence of sensor noise will create a sensory residual.

[5] A straight bar was also used in the previous section in notation for the states of the process model in pseudo-closed loop control. By the end of this section it should be clear why this is appropriate: a pseudo-closed loop system does not provide for any influence from the real sensor signal to the process model, and so the process model is restricted to working with *a priori* estimates.

[6] This is called a measurement inverse because it is the inverse of the measurement function. The measurement is a function that maps process states to sensor signals. For example, the people who take bearings on a ship are implementing a measurement. They produce signals, in this case bearings to known landmarks, on the basis of the process's state, in this case the location of the ship. The set of numbers produced by the bearing takers depends on the location of the ship. The inverse of this is a mapping from sensor signals to process states. When the navigation team is given a set of numbers from the bearing takers, they implement an inverse by determining where the ship would have to be if the sensor signals they have just received are accurate.

Before explaining how the KF arrives at its *a posteriori* estimate, it is worth exploring the way that the *a priori* estimates $\bar{p}(t)$ and $\bar{I}(t)$ and the observed signal *s*(t) differ with respect to how they handle the two kinds of unpredictability inherent in the process and measurement. These two sources of unpredictability are the process disturbance (unpredictable disturbances to the process's state during the last time step) and the sensor noise (imperfections in the signal produced). The *a priori* estimates are not affected by sensor noise, for the simple reason that they do not make any use of the observed signal. The weakness of the *a priori* estimates is process disturbance. If there were no process disturbance, then the *a priori* estimates would be entirely accurate (given of course that the estimate available from the previous time was accurate). But because of unpredictable changes to the state of the process, the *a priori* estimates will not be entirely accurate even if the previous state estimate was. The sensor signal is exactly the opposite. It is unaffected by process disturbance, for the simple reason that it is a measurement of the current actual state of the process. This measurement does not depend on how the process got to the state it is in.

These observations on the complementary strengths and weaknesses of the *a priori* estimate and the measured process state provide the rationale behind the way in which the KF produces its *a posteriori* estimate. If the magnitude of the process disturbance is large compared to the magnitude of the sensor noise, then the KF will weight the observed signal more heavily than the *a priori* estimate when combining the two. In the navigation case, this would be a situation in which we know that there are strong unpredictable currents (we know we are in the middle of a storm, for example), but the people taking the bearing measurements are very accurate and reliable. In such a situation, if your prediction based on the last location of the ship was in significant conflict with what the people taking the bearings are telling you, you go with what the reliable bearing takers say, and credit the disparity between what you predicted and what you observed to process disturbance that

moved your ship off course. If, on the other hand, there is very little process disturbance (the seas and winds are calm), but the measurement process is very noisy – perhaps the bearing takers are drunk – then you will weight your prediction based on the last estimate more heavily, and credit any major difference to noise in the observed signal.

The relative magnitude of the process disturbance and the sensor noise determines a factor, called the Kalman gain, that the KF uses when combining the *a priori* estimate and the measures process state. It is, roughly, a fraction that determines the relative weight given to each when they are combined to form the *a posteriori* estimate. Given the structure of the problem faced by the *KF* and the way in which this gain is determined, this estimate represents the optimal estimate, in the sense that there is no better estimate that could be produced given the information that the KF has access to. The KF then subjects this *a posteriori* process state estimate $\hat{p}(t)$ to a measurement, and the result is the KF's *a posteriori* estimate of what the real, noise-free signal is: $\hat{I}(t)$. The *a posteriori* estimate then forms the starting point for the *a priori* estimate of the next cycle, and the cycle repeats.

It is a trivial matter to integrate the KF into a control scheme. This is shown in Figure 4.

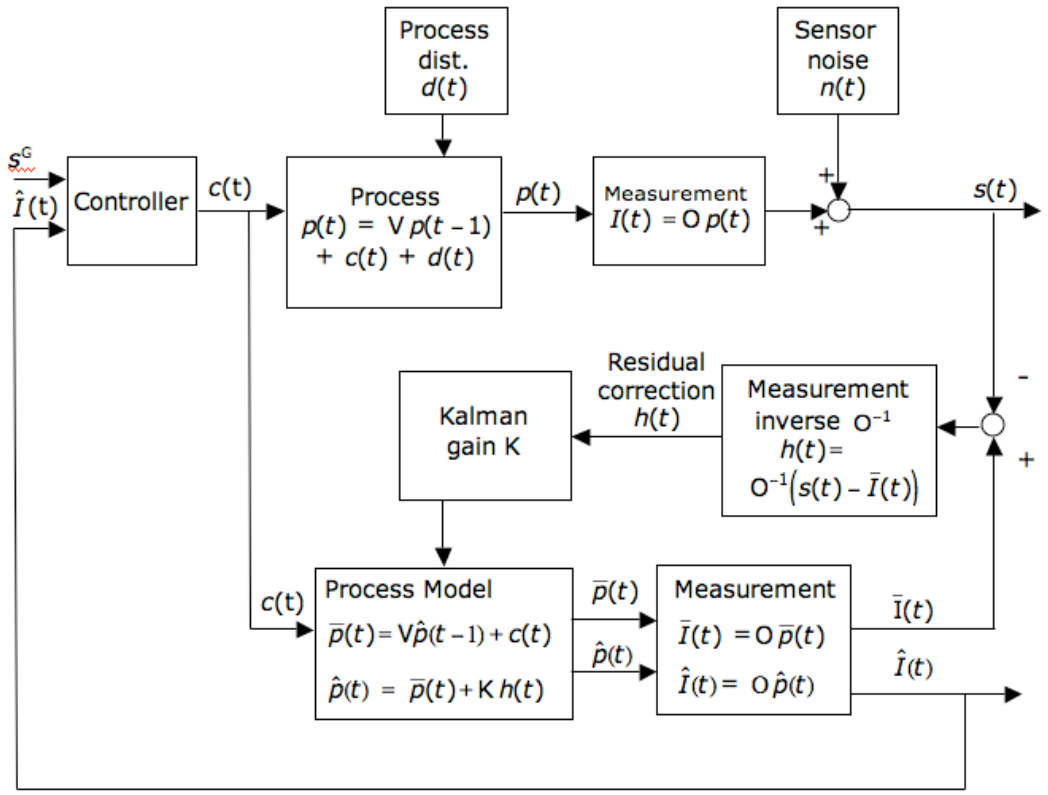

Figure 4. A control scheme that uses a Kalman filter.

The control architecture diagrammed in Figure 4 has the various control structures discussed in the previous sections as special cases. An open-loop control scheme results if the mechanisms shown in the bottom two-thirds are ignored. A closed-loop system results if the Kalman gain is set so as to always use the entire residual correction. This is because if the entire residual correction is used, then the *a posteriori* estimate will always be equivalent to the observed signal, and hence the signal sent back to the controller will always be equivalent to the signal produced by the actual measurement of the process, just as in closed-loop control. Finally, if the Kalman gain is set to ignore the sensory residual altogether, then the scheme becomes equivalent to pseudo-closed loop control. If the sensory residual is ignored, then a 'a posteriori' estimate is always equivalent to the *a priori*

estimate, and in this case the process model will simply evolve over time as a result of its own inner dynamic and the efference copies. But perhaps the most perspicuous way to understand the scheme is as a closed-loop control system that uses a Kalman filter to filter noise from the feedback signal.

I have used the KF as an example, mostly because the KF is mathematically well defined, and by using an example that is so well defined, a number of distinctions can be made more efficiently than they could have been with a more qualitative example on hand. But the emulation theory is not identical to Kalman filtering. The KF is *one example* of the emulation framework, but not the only example. The emulation framework is an information processing strategy according to which a system maintains a model of some other system, and uses this model in various ways: it can operate on the model independently of the modeled system in order to get an idea of how the modeled system would behave in various circumstances; it can operate the model in parallel to the modeled system to overcome problems with sensory access to the modeled system, such as feedback delays, noisy sensors, etc. The KF is relatively restricted: it does not run the model off-line, and when run in parallel with the process it does not use the model in such a way as to deal with feedback delays (the KF is set up to solve a problem other than feedback delays), and it combines the sensory information and the model's state in very specific ways. The emulation framework drops these restrictions. An emulator-employing system might use an emulator to help process sensor signals, but not strictly determine anything like a Kalman gain to *optimally* combine then, but use some other method, perhaps including just giving each equal weight. It might use the emulator completely off-line to produce imagery, or run thought experiments (more on this below).

2.2 Modal vs. amodal emulation

In this section I will very briefly introduce a distinction between two kinds of emulation, what I will call modal and amodal emulation. This will be a very brief discussion, more detail on the difference between modal and amodal emulation can be found in Grush (2004). All of the emulators I have described so far have been models of the *process*. Additional components, specifically measurement functions, have been appended to the process and the process model to map states of the process or process model into signals in sensory format. These are all examples of what I will call amodal emulation. The emulator itself, the process model, is not tied to any sensory modality. Rather, the sensors are distinct mechanisms, and the 'modality' would be the way the process (or process model) is measured. The following analogy should help make this clear. Your environment – the spatial vicinity around you and the objects in it – is not intrinsically visual or auditory or gustatory. The environment itself is amodal. Your body has various sensory modalities that can measure the environment in different ways. Visual sensors produce signals that give information about some of the states of the environment, but are insensitive to others. Auditory sensors produce signals that pick up on features that vision is deaf to, but audition is itself blind to many features vision picks up on. And so forth for other modalities.

The situation is similar for emulators. A model of the process can be subjected to various measurements: measurements that produce estimates of what the visual signal will be; measurements that produce estimates of what the auditory signal will be; and so forth. (One might take a navigator's ship model and 'measure' its location, but not direction or speed; or one might measure its heading, but not location.) But another kind of emulator is possible – modality-specific emulators. This can be illustrated with an example, Bartlett Mel's *Murphy*. (Mel, 1986). Murphy consists of a system that controls a robotic arm, its task being to move its arm around obstacles in a workspace and grasp targets. The arm's

movement is confined to a plane. Murphy observes the workspace and its arm's motion via a video camera above the workspace. This camera drives a 64x64 grid of pixels that function as a low-resolution image, which is Murphy's sole access to what is happening in the workspace. As Murphy issues commands to its arm, in the form of changes to its shoulder, elbow and wrist angles, it sees the results of its commands on the grid.

Murphy, however, implements a *modal* emulator of its environment. The units on the 64x64 grid are actually connectionist units that learn to predict what the results of given motor commands will be on the next workspace image. For example, if the pattern of activation on the grid is G_1, and Murphy issues motor command M_1, and this results in the pattern of activation on the grid changing to G_2, the units learns that in the future if the grid is displaying G_1, and M_1 is issued, the next image on the grid will be G_2. Once the grid has observed and learned from a sufficiently large sample of Murphy's overt operation, Murphy can take its arm and workspace off-line, and operate only with the visual grid in emulation mode. Starting with an image of an initial arm and workspace configuration, Murphy issues motor commands that are suppressed from affecting the real arm (this is what it means to take the arm off-line), but are nevertheless processed by the visual grid such that the grid changes its state into an estimate of what the visual grid state would be if the arm were still online. Murphy is thus producing visual imagery by driving its visual grid with efference copies. Once Murphy has discovered a way to move its imagined arm around the imagined obstacles to grasp the imagined target, it puts its arm back online and implements the solution it found.

The sort of emulation that Murphy implements can be diagrammed in Figure 6.

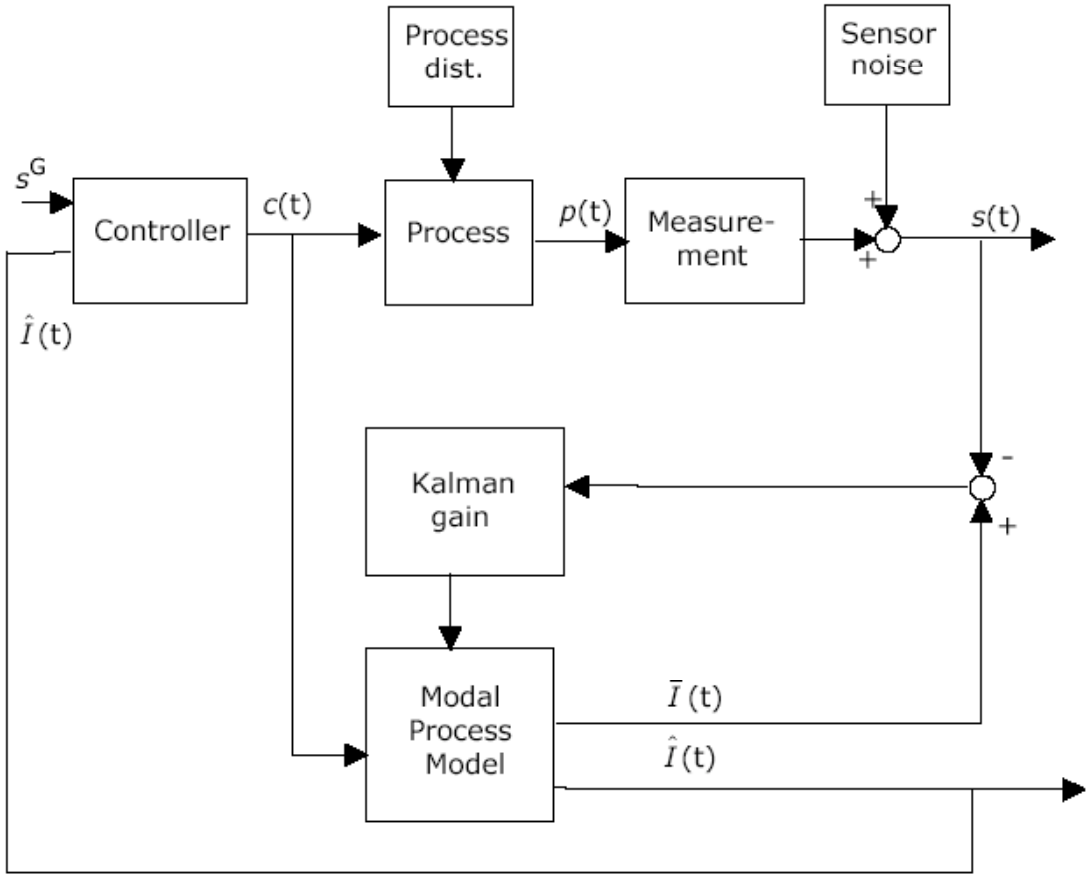

Figure 6. A modality-specific emulator.

Here, the emulator is not a model of the process per se, but a modality-specific emulator of the system consisting of the process and a specific modality of measurement. Because of this, the emulator's output is not subjected to a measurement. Its output is already, by definition, in the format of the relevant sensory system. In Murphy's case this was visual.

We can easily imagine systems that combine modal and amodal emulation. Such a system is shown in Figure 7. Here, the same efference copies drive an amodal process model as well as a modality specific emulator. Both can be used to produce *a priori* estimates: one an

estimate of what the state of the process will be, the other a modal estimate of what the sensory signal will be.

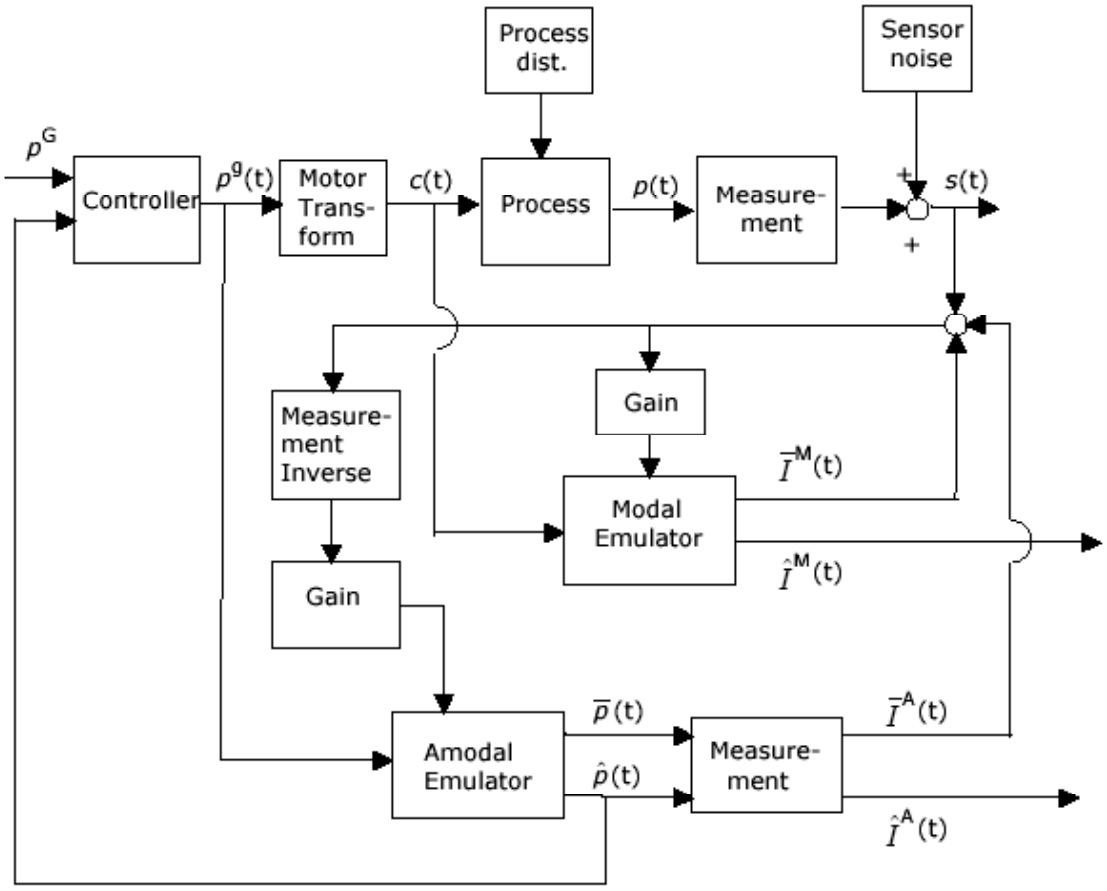

Figure 7. A system that combines modal and amodal emulation.

The system described in Figure 7 employs two emulators: one an amodal process model, the other a modality specific emulator of the sensory systems. One copy of the command is provided to the amodal process model in order to produce an *a priori* estimate of the process's state; another copy is processed by the modal emulator to produce an *a priori* estimate of the next sensory state.

When the amodal process model's *a priori* state estimate is measured, another *a priori* estimate of the sensory state is produced. In Figure 7 the *a priori* estimate produced by the modal emulator is called $\bar{I}^{M}(t)$, while the one produced by the measurement of the amodal process model is called $\bar{I}^{A}(t)$. A sensory residual is now determined by comparing the two *a priori* estimates with the observed signal[7]. This residual can then be used to produce *a posteriori* estimates of both the modal and amodal estimates.

Striking confirmation of the existence of a purely visual modality specific emulator comes from a phenomenon first hypothesized by von Helmholtz (1910), and discussed and verified experimentally by Ernst Mach (1896). Subjects whose eyes are prevented from moving and who are presented with a stimulus that would normally trigger a saccade (such as a flash of light in the periphery of the visual field) report seeing the entire visual scene momentarily shift in the direction opposite of the stimulus. Such cases are very plausibly described as those in which the perceptual system is producing a prediction – an *a priori* estimate – of what the next *visual* scene will be on the basis of the current visual scene and the current motor command. Normally a motor command to move the eyes to the right will result in the image that is currently projected on the retina (and hence fed to downstream topographically organized visual areas of the brain) shifting to the left. And some region in the nervous system is apparently processing a copy of this driving force and producing an anticipation of such a shifted image – an anticipated image so strong that subjects actually report seeing it briefly. Normally such a prediction would provide to specific areas of the visual system a head start for processing incoming information by priming them for the

[7] Generalizing the process of arriving at a sensory residual from a case of one *a priori* estimate and one observed signal to perhaps more than one of each is straight forward. In the case where one wants to maintain the optimal estimation capability of the KF, one would of course combine them as a function of their respective error covariation. Where optimality is not a desideratum, much simpler procedures would suffice, such as averaging the *a priori* estimates and then subtracting the observed signal.

likely locations of edges, surfaces, etc. This *a priori* prediction would normally be largely confirmed, and seamlessly absorbed into ongoing perception as part of the *a posteriori* estimates. You don't normally notice these images, but there are ubiquitous.

Just less than one hundred years after Mach published his experimental result, Duhamel, Colby and Goldberg (Duhamel et al. 1992) published findings that seem to point to the neural basis of this effect. They found neurons in the parietal cortex of the monkey that remap their retinal receptive fields in such a way as to anticipate immanent stimulation as a function of saccade efference copies.

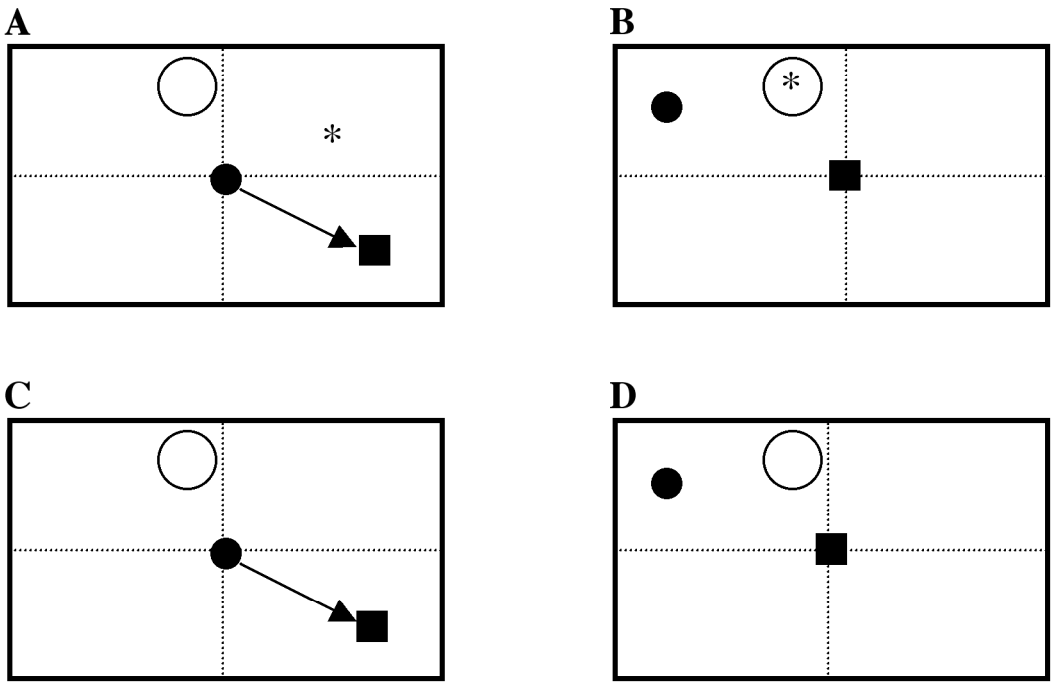

Figure 8. Anticipation of visual scene changes upon eye movement. See text for details.

The situation is illustrated in Figure 8. Box A represents a situation in which the visual scene is centered on a small disk. The receptive field of a given PPC cell is shown in the empty

circle in the upper left quadrant. The receptive field is always locked to a given region of the visual space, in this case above and just to the left of the center. Since nothing is in this cell's receptive field, it is inactive. The arrow is the direction of a planned saccade, which will move the eye so that the square will be in the center of the visual field. There is a stimulus, marked by an asterisk, in the upper right hand quadrant. This stimulus is not currently in the receptive field of the PPC neuron in question, but it is located such that if the eye is moved so as to foveate the square, the stimulus *will* move into the cell's receptive field, as illustrated in Box B. The Duhamel et al. finding was that given a visual scene such as represented in Box A, if an eye movement that will result in a scene such as that in Box B is executed, the PPC neuron will begin firing shortly after the motor command to move the eye is issued, but before the eye has actually moved. The PPC neuron appears to be anticipating its future activity as a function of the current retinal projection and the just-issued motor command. The control condition is shown in Boxes C and D. In this case the same eye movement to the square will not bring a stimulus into the receptive field of the neuron, and in this case the neuron does not engage in any anticipatory activity. (Or, more accurately, it *does* engage in anticipatory activity, and what it is anticipating, correctly, is that nothing will be in its receptive field.) The control condition effectively rules out the hypothesis that the PPC cell is firing merely as a result of the motor command itself. It is only if the motor command will have a certain sensory effect that the PPC cell fires. This is a neural implementation of the construction of an *a priori* estimate.

A full set of these neurons, covering the entire visual field, would explain the Helmholz phenomenon. And they would also constitute a modality specific emulator of the visual scene.

2.3 Application to motor control

Motor control

One of the challenges in our understanding of motor control is the fact that the temporal length of the control loop — how long it takes for a signal to travel along biological axons from the motor areas of the brain to the body and for peripheral signals carrying information about the results of the execution of that command and be processed so that they can beneficially influence the ongoing motor commands — is not short; on the order of 250-400ms (Dennier van der Gon, J.J., 1988; Ito, 1984). This *by itself* is interesting, for it seems as though motor control is relatively good during such short bursts, such delays notwithstanding. These twin facts have been one of the reasons that people have been drawn to models of motor control that are open-loop (a closed-loop control scheme with significantly delayed feedback would face significant problems) except for the very end of the movement, where feedback is presumed to be used to make refinements (see Desmurget et al. 2000). Even more interesting, though, is the fact that it appears as though the motor centers make corrections to the motor plan as quickly as 70ms or so after movement onset, corrections that appear to be made on the basis of peripheral information (van der Meulen et al., 1990). Thus it appears to be the case that the motor centers are getting and acting on peripheral feedback *before peripheral feedback should be available*!

In view of these facts and others, a growing number of motor control researchers are developing models of motor control that exploit forward models and (in some cases) Kalman filters (e.g. Blakemore et al. 1998; Kawato 1999; Wolpert et al. 2001; Krakauer et al., 1999; Houk et al., 1990; for a very recent neurobiological vindication of forward models in motor control, see Mehta and Schaal, 2002). The main idea is that emulators of the body (specifically the musculoskeletal system and its dynamics) can process efferent copies and

provide predictions of what the peripheral signal will be before the real peripheral signal is available (this was Ito's (1970, 1984) initial motivation for speculating on the existence of emulators in the cerebellum). A particularly clear example is Wolpert, Ghahramani and Jordan (1995). These authors developed a model of the information processing structure of human motor performance with the aim of explaining the temporal patterns of errors and corrections in various movement conditions, one particular aspect being the potentially disruptive effects of feedback delays in fast real-time movement. The authors develop a model that is essentially of the form in Figure 4, and show how patterns of movement errors under varying conditions can be explained on the assumption that during the initial phases of a movement, before proprioceptive feedback is available, the motor centers exploit feedback from an internal model of the musculo-skeletal system (*a priori* predictions), while as time progresses, feedback from the periphery is incorporated into the estimate (*a posteriori* estimates).

Motor imagery/motor planning

The previous subsection focused on the use of emulation during motor control — when the emulator is run in parallel with the real musculo-skeletal system so that its timely feedback could stand in for the delayed feedback from the periphery. Now we turn to off-line uses. There has been a growing interest in the so-called 'simulation' theory of motor imagery (Johnson 2000; Jeanerrod 2001). According to this theory, motor imagery — the imagined proprioceptive feelings of movement and force — are the result of the off-line operation of the motor areas of the brain. This is entirely in line with the emulation theory, which holds

that faux proprioception is the result of running the musculo-skeletal emulator off-line, and the motor centers are what drive the emulator.[8]

The benefits of such off-line operations are many, but I will mention only one here, *motor planning*. All movement tasks admit of an infinite number of solutions, some better than others. Depending on how your body is configured, and how your backpack is resting on the floor, and what you will be doing after you pick it up (opening it to look in it, grabbing it and walking away), taking an overhand or underhand grip on the strap might be better. We often choose the best grip quickly and without any conscious effort or awareness, but the motor systems are hard at work during the fractions of a second before the event. There is evidence (see Johnson 2000 and references therein) to the effect that this motor planning involves covert motor imagery produced in order to determine which of a small number of options will work best.

3 Trajectory emulation

3.1 Temporal generalization of the emulation theory

The KF is temporally degenerate in two senses. First, it produces state estimates of only one time at a time. Second, these two times are always the same: for all times t_i, the state estimate that the KF produces at t_i is an estimate of the state of the process at t_i. This last claim might seem surprising at first, since doesn't the KF produce a *prediction* when it produces its *a priori* estimate? No. At least not in the relevant sense. At the beginning of

[8] In Grush (2004) I discuss the simulation theory in more detail, and point out differences between the simulation theory and the emulation theory, as well as considerations that favor the emulation theory.

each operational cycle, the KF produces an *a priori* estimate of the process's state. This estimate is a prediction in the sense that, once measured, it provides a prediction of what the observed signal is, and this 'prediction' is compared with the observed signal to produce the sensory residual. But it is not a prediction in the sense that it is looking ahead in time. The *a priori* state estimate is an estimate of the state of the process as it *currently* is. It is a prediction in the sense that I might draw a card from a deck and ask you to predict what you will see when I turn it around. The card is already drawn when you produce your guess, so you aren't predicting what card I will draw. You are predicting what you will see when I turn the card around. It is only this sense in which the *a priori* estimate is a prediction.

The first generalization I will undertake in this section takes us away from the KF's ubiquitous focus on the co-temporaneous process state. The mechanisms already in place in the KF can easily produce *a priori* estimates of what are genuinely future states of the process. At time *t*, the KF can produce (as we have seen) an *a priori* estimate of the state of the process at *t*, by taking its previous *a posteriori* estimate together with any current driving force signal and updating it via the function *V*. Normally this estimate then gets measured and compared with the observed signal to produce the *a posteriori* estimate. But there are other options. The system might simply do another iteration of the time-update. That is, it can take $\overline{p}(t)$, together with any driving force that *will* be executed at time *t+1*, and use that as input to *V* again to produce an *a priori* estimate of the process's state at *t+1*: $\overline{p}(t+1)$. These iterations can continue indefinitely far into the future. The system can, at *t*, produce estimates of what the process's state will be at any arbitrary future time step by simply iterating the time-update as many times as it takes.

There are obvious limitations. Depending on the magnitude of the process disturbance, these state predictions may fail to be reliably close to the real future state of the process beyond some number of iterations. Similar limitations arise if the driving force that is

predicted to apply during future time steps is not perfectly predicted. But these limitations aside, the point is merely that such predictions can be produced by mechanisms already at hand. This production of estimates for genuinely future states of the process is *prediction*.

A similar but not entirely parallel process concerns the production of estimates of past state of the process. There are several ways that a system that has the tools of the *KF* could produce such past state estimates. First and easiest, it could simply *remember* the estimates that it actually produced at those previous times. Second, it could produce state estimates of previous states by taking its current state estimate and pushing it through an inverse of the time-update function. Just as one can produce a future estimate by employing the time update function V as in: $\overline{p}(t+1) = V\overline{p}(t)$, one can produce a previous estimate from a current one via $\overline{p}(t-1) = V^{-1}\overline{p}(t)$. Of course, one would get a better estimate of the state at *t-1* if one used as input the *a posteriori* estimate at *t*: $\overline{p}(t-1) = V^{-1}\hat{p}(t)$. And this process can also be iterated to produce estimates of states arbitrarily far in the past. (E.g., if the ship is now at location X, and its speed is S and heading is H, then it was probably at Y at the last fix cycle.)

While either of these will produce an estimate of the state of the process at some previous time, each has a shortcoming.[9] A third method, *smoothing*, effectively combines the two. At time *t*, a smoothed estimate of the state at time *t-1* is arrived at by combining the *a posteriori* estimate of the processes state at *t-1* that was actually produced at *t-1* (this is the first method above), with a state estimate arrived at by backtracking one time step

[9] I've decided to unburden the main text of an explanation of these shortcomings. But for those who are interested the following remarks and reference should suffice. Neither of the two methods described above makes use of all the available data. A previous remembered estimate does not take into account observations made after that estimate, and these observations may be crucial. Estimates of past states made by back iterating a future estimate through an inverse of the time update tell us what the most likely past state is given only the current state. But the most likely past state given only the current state might differ from what is the most likely past state given both the current state and the past observations. Each of these methods ignore some chunk of the observed signal. See Bryson and Ho (1975).

from the *a posteriori* state estimate of the process at *t* that was produced at *t* (this is the second method above):

$$(1)\ \tilde{p}(t\text{-}1) = \hat{p}(t\text{-}1) + K_S(V^{-1}\hat{p}(t))$$

Here, $\tilde{p}(t\text{-}1)$ is the smoothed estimate of the process's state at *t-1*; K_S is a factor analogous to the Kalman gain that determines the relative weight given to the *a posteriori* estimate from *t-1* and the backtracked *a posteriori* estimate from *t*.

With the addition of smoothing and prediction we have generalized away from the KF's myopic focus on the *now*, to a system capable of generating state estimates of the process as it was in the past or will be in the future. Next we must generalize from the focus on estimates of the process's *state* at a *single time* (whether past, present or future) to estimates of the process's *trajectory* over a *temporal interval*. The basics of this are easy enough to do. We can describe a system that maintains estimates of the states of the process throughout an interval from *t-l* to *t+k*. Of course, if *l*=*k*=0, then we have the degenerate case of an estimate of an interval that consists of only one time, *t*. I am interested in cases where *l* and *k* are both greater than 0:

$$(2)\ (p_{t-l}, p_{t-l+1}, \ldots, p_t, \ldots p_{t+k-1}, p_{t+k})$$

To make the notation a bit more elegant, I will use $\tilde{p}_{[b,c]/a}$ to be the estimate of the trajectory of *p* over the interval from time *b* to time *c*, produced at time *a*. Hence the estimate in (2) would be $\tilde{p}_{[t-l,t+k]/t}$. I will call a system that constructs and maintains estimates of the trajectory of the process over a temporal interval a *trajectory emulator*.

3.2 Psychological phenomena

The overall idea that the brain employs a trajectory emulator that maintains a trajectory estimate over an interval can be illustrated with a number of phenomena. First consider apparent motion. Two successive flashing dots presented within some spatial distance and within some inter-stimulus interval will appear to be a single moving dot, moving from the location of the one that flashes first to the location of the one that flashes second (see Figure 9). This can look to be merely a spatial illusion, in that it looks as though a dot has moved through spatial areas where no dot has in fact been. For example, it appears as though the dot occupied and moved through location B as indicated on the right hand side of Figure 9. To bring out the temporality of the phenomenon, consider that the subject will appear to see the dot first at the location A, then location B, and then finally at the location C – the motion is actually perceived to be continuous, but I am just drawing attention to the temporal relations between three of the posits on the continuous path.

Figure 9. Apparent Motion. The left hand side represents actual stimuli, a flashing dot (1) followed by a second flashing dot (2). The right hand side represents what is perceived: a single dot moving from location A (the location of the first dot's flash), through location B and to location C (the location of the second dot's flash).

Notice, however, that if the second flashing dot were above, or below, or to the left of the first, then the subject would have seen the dot as moving upward, or leftward, or downward. And accordingly, the intermediate location B would be either above, to the left

of, or below, location A. But – and this is the crucial bit – until the second dot actually flashes, the subject cannot know in which of these four spatial regions the interpolated motion (the location of B) should occur. Yet the *subject sees the dot as being at the interpolated location before being at the terminal location where the second flash occurs*. It can seem as though the perceptual system is able to foretell where the second flash will be in order to appropriately begin filling in the intermediary phases of the apparent motion.

The trajectory estimation model explains this without recourse to the supernatural. At the time of the first flash, the estimate is that there was a single flash at time *t*. If nothing else happens, then as time progresses this estimate will not change. At *t+1*, and *t+2*, and so on, the estimate will continue to be that there was a flash at *t*. Suppose, however, that at *t+2* a second flash occurs at location C. The visual system has models of what happens in the environment, and the relative likelihood of various events, according to which two discrete sensed flashes at A and C at times *t* and *t+2* respectively is, if the temporal and spatial magnitudes are small enough, more likely an imperfectly (noisily) sensed continuous stimulus traversing the path from A to C, than a perfectly sensed pair of distinct stimuli in close spatial and temporal proximity. So at *t+2*, the trajectory estimate is revised to represent a stimulus as having been at location B at *t+1*.

Another fascinating phenomenon that is potentially explained by the mechanisms introduced so far is *representational momentum*. The basic phenomenon is this: subjects are shown a movie or sequence of images that shows some sort of movement, such as a rotating rectangle or moving ball. The scene stops, and subjects are probed to determine their assessment of the final state of the motion they saw. The result is that subjects judge that the motion or rotation continued farther than it in fact did. Not surprisingly, the phenomenon does require that the motion is predictable (Kerzel 2002).

The trajectory emulation framework explains this phenomenon easily. Even though there is only a sensory signal corresponding to the motion up to time *t*, the prediction end of the trajectory emulator produces predictions of the future state of the process. These predictions will be possible because the emulator has knowledge of the way the process typically behaves. Senior et al. (2002) describe the situation this way:

> Representational momentum is thought to occur due to the encoding of specific contexts and semantics inherent within the stimuli (e.g., the encoding of the effects of gravity in a picture of a man jumping from a ledge). These contexts are encoded on the basis of prior knowledge of how complex objects behave in the real world. (Senior et al. 2002)

When the subject is probed to determine the extent of their representation of the object's motion, this probe is picking up on representations that were constructed as predictions at the leading edge of the moving window.

4. Spatial information processing and basis functions

We have to switch gears now, from one kind of information processing structure to another – from time to space. The question how the brain represents space is not as straight-forward as one might have thought. Surprisingly, even though the brain makes use of a great many topographic maps – maps of the body surface and of the retinae, to name just two of dozens – it does not appear to represent egocentric space by means of anything like a map. The parietal cortex is certainly the most important cortical area for representing egocentric space. Lesions to this area result in characteristic spatial deficits, both perceptual and behavioral. But single cell recordings have failed to even hint at anything resembling such a map in the parietal lobes. What has been found by single cell recordings are cells

that are responsive to a number of sensory and postural signals. Understanding what these have to do with spatial representation requires the right theory, to which I turn.

Perhaps the major key to understanding the operation of PPC neurons and hence the PPC as a whole was a connectionist simulation by Zipser and Andersen (1988), a model that is also one of the most prominent triumphs of connectionist computational neuroscience. Zipser and Anderson constructed an idealized version of the problem confronted by the PPC. A system with a one rotatable eye was to determine the direction (in head-centered coordinates) of a stimulus projected onto its retina – in other words, determine the direction of an external stimulus given only information about stimulation on sensory receptors and postural information. The system in question was a connectionist network trained by back-propagation (see Zipser and Andersen (1988) for details). During training the network was given an input vector that contained information about where on the retina a stimulus was projecting, and how the eye was oriented. The output vector was a specification of the direction of the stimulus in head-centered coordinates. During training the network was provided with training sets consisting of correct input-output pairs, and it was trained to learn to match them. Subsequent testing on novel inputs revealed that it could produce correct outputs.

Because this was an artificial network, it was possible to subject the units to any kind of analysis one might want, unlike real neurons that are difficult to study. The key as always is the hidden units of the network – the processing elements that lie between the inputs and outputs, the details of whose operation is forged during the training period in such a way as to produce the correct output vector (in the output units) upon receipt of an input vector (from the input units). The result of this analysis was that *each* of the hidden units was acting as a linear gain field combining both sensory and postural information, which in this case were the location of retinal projection and eye's orientation respectively. And in

particular, each hidden unit's activity was the product of (i) a Gaussian function of the stimulus's distance on the retina from the unit's preferred retinal location, and (ii) a linear gain determined by the unit's preferred eye orientation. The output units each performed a linear combination of the hidden units' activities.

It will help to walk through an example involving two hidden units, and restricting discussion to two dimensions (what I am about to describe are not actual hidden units in the model, but are fictitious exemplars of how the actual hidden units operate). Hidden unit h_1 would have a preferred location of retinal stimulation, and one factor in its response would be a Gaussian function of distance from that spot (see Figure 10). If we number the retinal locations in order from 0 to 100 (we are simplifying to a two-dimensional realm, and so the retina will be one-dimensional), unit h_1 might have its preferred location at location 70, meaning that this factor in determining the unit's activity is strongest if the stimulus projects directly on retinal location 70, and decreases with distance from 70. It still is pretty strong at 68 or 72, but very weak at 10 or 98. The second factor influencing h_1's activity is a linear function of deviation from preferred eye orientation (see Figure 11). This factor would be strongest if the eye was oriented either completely to the left or completely to the right (whichever the preferred direction is, let's say it is *right*), and would taper off linearly as the eye's orientation went to the other direction (left).

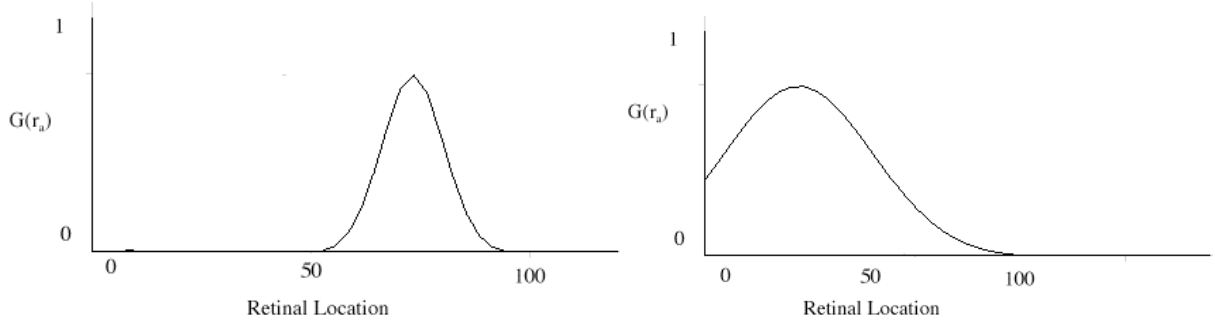

Figure 10. Gaussian functions. The graph shows the value of $G_1(r_a)$, the Gaussian used by unit *h1* (left), and $G_2(r_a)$, the Gaussian used by unit *h2* (right). The Gaussians differ both in terms of their preferred retinal location (peak response of $G_1(r_a)$ is at about 70; $G_2(r_a)$ is about 30), and their tuning ($G_1(r_a)$ is narrowly tuned, meaning it falls off quickly with distance from the preferred location; $G_2(r_a)$ is broadly tuned, meaning it falls off less quickly with distance from preferred location).

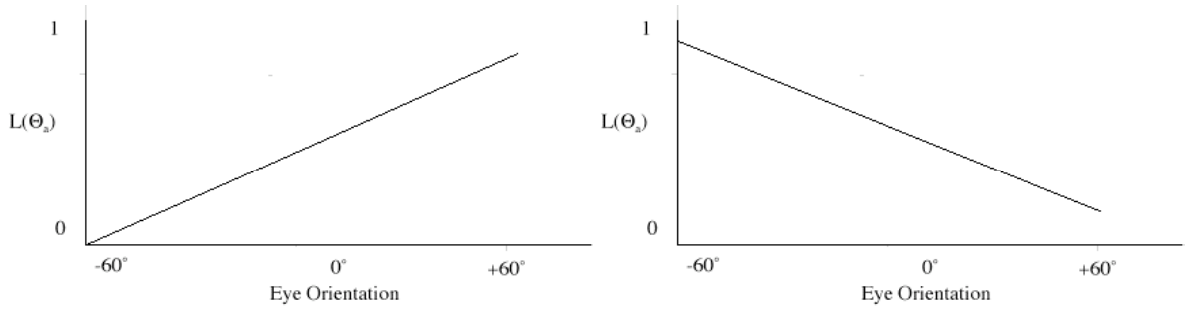

Figure 11. Linear functions. The graph shows the value of $L_1(\theta_a)$, a linear function used by unit *h1* (left), and $L_2(\theta_a)$, a linear function used by unit *h2* (right). The function is at its maximum when the eye is oriented either to the far left or far right, depending on the preferred orientation. The activity tapers off linearly as the eye's orientation differs from the preferred orientation.

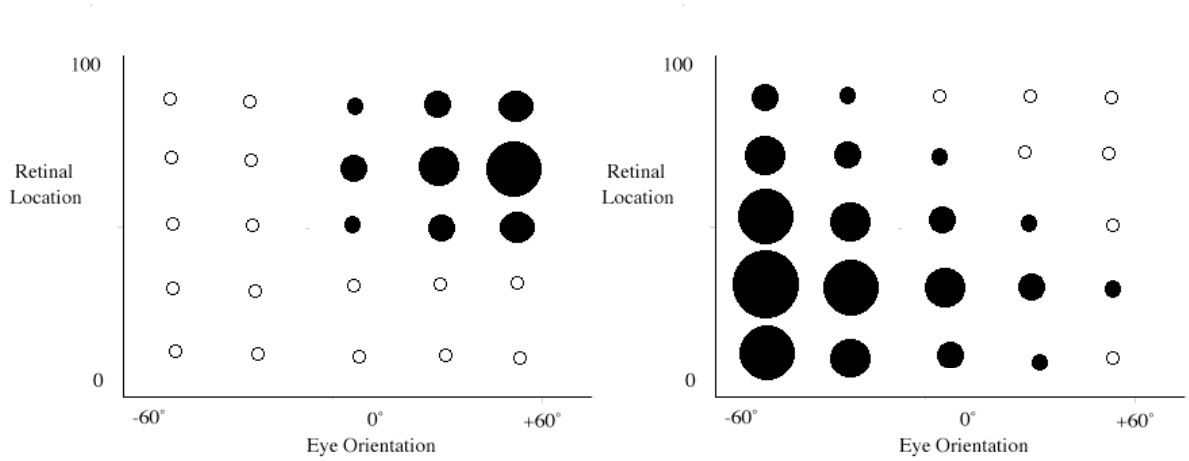

Figure 12. Graphical representations of the *product* of a Gaussian and linear gain. The figures show the activity of hidden unit h_1 (left) and h_2 (right) for 25 different combinations of retinal location and eye orientation. Empty circles represent no activity (a product at or very near 0), while filled circles' sizes are proportional to the unit's activity for that combination. As can be seen, the product of $L_1(\Theta_a)$ and $G_1(r_a)$ is highest with a combination of retinal location around 70 and eye orientation fully to the right. The product of $L_2(\Theta_a)$ and $G_2(r_a)$ is highest with a combination of retinal location around 30 and eye orientation fully to the left.

So for a given combination of retinal location of stimulation and eye orientation, h_1 would have its firing rate determined as a product of these two factors: a Gaussian of distance from preferred location and a linear function of eye orientation. If the eye is orientated completely to the right, and the stimulus falls directly at retinal location 70, h_1 will fire at its maximal rate (see Figure 12). The actual location of the stimulus's projection on the retina together with the eye's orientation during the sensory episode will determine a firing rate. But of course the firing rate of h_1 by itself won't tell one where the stimulus is. Suppose that h_1 is firing at half its maximal rate. This might be a case where the eye orientation is all the way to the right, but the retinal stimulation is at location 80, or 60, and hence a bit distant from its preferred spot. Or the stimulus might project directly to retinal location 70, but the eye might be oriented half way from its preferred direction. Many combinations would yield the same firing rate. In any case, each cell's activity is a function of these two factors:

(3) $h_1 = G_1(x_a)L_1(\Theta_a)$

Where h_1 is the activity (firing rate) of unit h_1; x_a is the actual location of retinal stimulation; G_1 is the Gaussian function used by unit h_1; Θ_a is the actual eye orientation; and L_1 is the linear function used by unit h_1.

The activity of unit h_2 will be determined similarly:

(4) $h_2 = G_2(x_a)L_2(\Theta_a)$

The Gaussian function G_2 may be different from G_1 (see Figure 10), in particular σ (a constant that determines how narrowly or broadly tuned the Gaussian curve is) may be larger or smaller; and it may be centered at a different retinal location. And the linear function L_2 may be different from L_1, i.e. have a different slope. We can streamline our notation as follows:

(5) $h_1 = f_1(x_a, \Theta_a)$

(6) $h_2 = f_2(x_a, \Theta_a)$

The function associated with each unit (f_1, f_2, etc.) will be a linear gain field, a Gaussian multiplied by a linear gain. But since the exact form of the Gaussian (center and σ) and the slope of the linear gain might be different in each case, they are different functions. More generally, for each unit h_i, its activity can be expressed as

(7) $h_i = f_i(x_a, \Theta_a)$

In Zipser and Andersen's model, as explained above, the functions f_i are all linear gain fields. As is the case with such connectionist models, the output units' activities – where the answer gets produced – are simple linear combinations of the activities of the hidden units. That is, each output unit O_i is connected to each hidden unit with a scalar connection strength or *weight*, $w_{i,j}$, which is the strength of the connection from hidden unit *i* to output unit *j*. This output unit's output is the sum of all of these weighted connections.

(8) $O_i = \sum_{j=1}^{n} w_{i,j} h_j$

That is, each output unit's activity O_i is the sum, for all *j*, of the product of the activity of hidden unit *j* and the weight from hidden unit *j* to output unit *i*. The set of these output unit's activities is the connectionist model's answer. It is a vector representation of the direction of the stimulus relative to the head.

(9) $S = (O_1, O_2, \ldots O_n)$

The mathematical representation is helpful, but not essential. For those who have a distaste for math, the qualitative idea is this. There are two stages to the solution. In stage one, the set of hidden units has learned a particular way of combining the sensory and postural signal. This way is that each unit has its activity determined as a slightly different function of both of these factors. This is what equation (7) expresses. The second stage involves decoding the activities of these hidden units in a certain sort of way. This is codified in equations (8) and (9).

Note that the fact that the model is limited to one sensory modality (vision), and only one postural signal (eye orientation) is only for simplicity. The same mechanisms can easily generalize to more realistic cases where not only can the eyes move in the head, but the head can move with respect to the torso, etc. In such cases additional signals are needed, such as signals coding the head's orientation with respect to the torso. Similarly, feeling something on the tip of my right index finger does not tell me where that object is located in egocentric space, unless I have postural information about how my index finger is angled with respect to my hand, how my wrist is comported with respect to my forearm, and my elbow and shoulder angles, and so forth. In this case too a sensory signal needs to be combined with postural signals in order to have enough information to narrow down the egocentric location of the stimulus.

Generalizing the notation to these more complicated cases is straight-forward. In the special case described in (7), the value x_a is a sensory signal – it is information about what is happening on the sense receptor, where the sense organ is being stimulated. The value of Θ_a is a postural signal – it is information about how the sense organ is oriented with respect to other body parts. We can make the notation more general as in (10):

(10) $h_i = f_i(s_a, q_a)$

Here, s_a and q_a are the sensory and postural signals associated with stimulus a, whatever form they may take.

What makes this model particularly interesting is that after it was determined that these artificial units had solved the problem by combining the sensory and postural signal in this

particular way (linear gain fields as per equation (3)), this suggested that it was at least possible that this was what individual neurons in the PPC were doing. And in fact, single cell recordings of neurons in PPC during tasks that required spatial localizations of directions in head-centered coordinates revealed that many neurons appeared to have their activity modulated in just this way (or close to it, see below). That is, the activity of individual neurons in PPC was found to vary with both eye orientation and location of retinal stimulation in the non-obvious way suggested by equation (3). Close enough anyway to provide some plausibility to the claim that the way these artificial neurons were solving the problem might be a window into how the PPC neurons were solving it.

This leads to a more recent proposal by Alexandre Pouget. Pouget's *Gauss-sigmoid* model also can be discerned into two stages, both of which are similar to the corresponding stages of the Zipser and Andersen model, but with some significant differences. (Pouget describes his model as a 'basis function' model, but for reasons I will explain shortly, I want to reserve the name 'basis function model' for a generalization of Pouget's model, and so will use the specific kind of basis function encoding described by Pouget – Gauss-sigmoid – as the name for his model). As in the Zipser and Andersen model, the first stage is an encoding stage, in which individual units or neurons combine the sensory and postural signals. The difference is that Pouget's model has each unit combining the signals not as a *linear* gain field (a Gaussian times a linear gain as with Zipser and Andersen's model), but by a non-linear basis function consisting of the product of a Gaussian and a non-linear sigmoid function (see Figure 13). There are information-processing reasons and physiological reasons for preferring these basis functions to the linear gain fields of Zipser and Andersen. On the information processing side it turns out that non-linear basis functions have convenient computational and mathematical properties. I will return to this shortly. The physiological reason is that upon closer scrutiny the PPC neurons look as though they may actually be implementing these Gauss-sigmoid functions rather than the linear gain fields. (The output

of both of these is very similar over most of the dynamic range of neuron's response, and so it takes some subtle measurement to tell which is the better fit.) Generally, Gaussians and sigmoids are functions that real neurons are known to be able to compute.

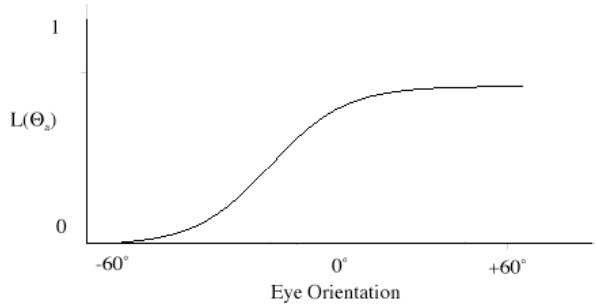

Figure 13. A sigmoid function.

The second stage is where the more interesting innovation is to be found. On Pouget's model, the basis functions are not just there to supply information to another set of units, the output units, that read out the 'correct answer', but are instead used to guide a motor response. This works as follows. Every possible *type* of behavior, such as 'grasp with the right hand' involves a complex set of motor control signals, and the exact nature of this motor control sequence depends on where the stimulus is. Grasping the coffee cup in front of you with your right hand involves a different set of motor commands than grasping it when it is down near your left foot. What Pouget has shown is that given a non-linear basis function representation of the sensory and postural information, the details of any behavior can be appropriately determined as a proprietary linear combination of the values of these functions.

Let's walk through this more carefully. A stimulus will be processed by the PPC as a set of basis functions of the form

(11) $B_i(s, q)$

Here, each basis function (B_1, B_2, ...) is an activity of a PPC neuron, its activity being a non-linear function of the sensory and postural signal, a non-linear version of the Zipser and Anderson model; that is, ((3) is essentially a linear version of (11)). The result of this stage is similar enough to the Zipser and Andersen model that the differences are not worth graphing: except for the change from a line to a sigmoid and a corresponding slight difference in the exact form of graphs such as those in Figure 12, a parallel set of figures corresponding to a Gauss-sigmoid field would look very similar. A given perception of a stimulus will involve a certain sensory signal and set of postural signals, the *s* and the *q* associated with the sensing of the stimulus, and the PPC applies a large number of basis functions of the form of (11) to these signals. The result is a lot of PPC neurons, each firing at a rate determined by its own version of (11).

A given kind of behavior, such as a *grasp with the left hand*, or a foveating eye movement, is associated with a set of scalar coefficients (the mathematical version of neural connection strengths). So for example a motor behavior such as a *left-hand grasp* would have a proprietary and constant set of numbers, g_1, g_2, …, g_n, such that when those coefficients are used to produce a linear combination of the basis function values associated with stimulus *a*, the stimulus-*a*-targeted behavior is correctly executed:

$$(12)\ m_j^{a,g} = \sum_{i=1}^{n} g_{i,j} B_i^a$$

$$(13)\ M^{a,g} = (m_1^{a,g}, m_2^{a,g}, \ldots, m_p^{a,g})$$

What (13) says is that the neural motor commands that result in a *left-hand grasp* of stimulus *a* can be represented as a vector $M^{a,g}$, each component of which is $m_j^{a,g}$, and is arrived at by multiplying numbers associated with a *left-hand grasp* (the $g_{i,j}$ coefficients) and the basis functions associated with stimulus *a*, (the B_i^as) and adding them together, as per (12).

Of course a left-hand grasp directed at stimulus *b* will require a different motor command if *b* is located at a different spot in egocentric space. A left-hand grasp directed at *b* would be determined by:

$$(14)\ m_j^{b,g} = \sum_{i=1}^{n} g_{i,j} B_i^b$$

$$(15)\ M^{b,g} = (m_1^{b,g}, m_2^{b,g}, \ldots, m_p^{b,g})$$

Here, the different motor command $M^{b,g}$, the command that results in a left-hand grasp of stimulus *b*, is produced by taking *the same set of left-hand grasp coefficients*, the $g_{i,j}$s, and multiplying them with a different set of basis function values – the ones that the basis functions yield when applied to the sensory and postural signals produced during the sensing of stimulus *b*: B_i^b. A different kind of action, like an *eye movement* that foveates stimulus *a* or *b*, would be determined in an analogous way: by multiplying the eye

movement coefficients ($e_{i,j}$) with the basis functions produced by the stimulus according to the following equations that should not need elaboration at this point:

$$(16)\ m_j^{a,e} = \sum_{i=1}^{n} e_{i,j} B_i^a$$

$$(17)\ M^{a,e} = (m_1^{a,e}, m_2^{a,e}, \ldots, m_p^{a,e})$$

$$(18)\ m_j^{b,e} = \sum_{i=1}^{n} e_{i,j} B_i^b$$

$$(19)\ M^{b,e} = (m_1^{b,e}, m_2^{b,e}, \ldots, m_p^{b,e})$$

5. Emulation and basis functions

The emulation theory and the basis function model are not in conflict. The emulation theory is a description of how a representation of some process can play a role in a larger information processing structure that underwrites various capacities, such as imagery, perceptual processing, the dulling of the detrimental effect of feedback delays, and the filtering of sensor noise. The emulation theory makes no requirements on *how* the emulator itself is implemented, with the exception of the requirement that however it is implemented it must be able to play its role in the larger structure. The emulator might be implemented in a physical model, like in old-school ship navigation; or it might be implemented in a digital data structure. So long as it can be used to provide *a priori* estimates, be updated to form *a posteriori* estimates, be driven off-line to produce imagery, etc., the details don't matter – at least not so far as the emulation theory itself is concerned.

I claimed that for higher organisms, including humans, normal perception of the environment involved the use of an amodal environment emulator. This emulator maintains representations of the spatial comportment of objects and surfaces in the environment, as well as their movement and force-dynamical interactions. One aspect of this is clearly the *spatial* aspect. The emulation theory does not specify how these spatial properties and relations are represented, so long as they are able to engage in the kinds of processes the emulation theory posits.

The basis function model is a theory of how the brain processes information to track the location of objects in the environment. This theory was presented (both by me in Section 4, as well as by the original authors, though they did not put it in these terms) as a pure *measurement inverse* – one that goes from sensed signals (in this case, sensory and postural signals) to a construction of a representation of the spatial properties and relations of entities in the environment, which in this case is in terms of a set of basis function values that were determined by the relevant sensory and postural signals:

(20) $B_i(s, q)$

The advantage of representing spatial locations as sets of basis function values is that in this format they can be used immediately (without further processing) by the motor centers to produce appropriate actions, as described in the previous section.

But note that the 'sensory' and 'postural' signals are *observed* signals – unfiltered signals going directly from the process (the body and its sense organs) to the PPC. These observed signals are then combined via basis functions and linearly combined with a set of

coefficients appropriate to the target behavior. The result is a motor command that, if all works well, results in the target behavior being produced. To put it in other words, the basis function model as described by Pouget and collaborators, and as I have described it above, is an implementation of a closed-loop control scheme. It specifies one way to construct a closed-loop controller that will produce the right motor command when given feedback (the observed sensory and postural signals) from the process.

During the discussion of the control theoretic material we never had a need for notation for a process state estimate that was nothing more than the observed signal pushed through a measurement inverse. I will now use an inverted hat for this purpose: $\breve{p}(t) = O^{-1}s(t)$. Given this, the basis function values as discussed by Pouget and myself above are:

(21) $\breve{B}_i(\breve{s},\breve{q})$

Here, the sensory signal and the postural signal are completely unfiltered (hence the upside down hats), and as a result the basis function values are an unfiltered measurement inverse.

Just like any other observed signal and associated process state estimate, the sensory and postural signals and the resulting basis function values might benefit from some filtering. What this requires in the present case is an emulator or emulators of the spatial features of objects in the environment. Such an emulator will simply be a mechanism that can take a representation of the spatial features at *t*, together with an efference copy, if any, and produce an estimate of the spatial features at *t+1*. If we let $B_i(t)$ be the values of the basis functions produced at time *t*, then the spatial process model would be some mechanism that implemented the following:

(22) $\overline{B}_i(t+1) = V\hat{B}_i(t) + c(t)$

That is, it produces an *a priori* estimate of what the next set of basis function values will be on the basis of the previous set (in this case, the previous *a posteriori* estimate) and any efference copies. There are two ways this might be implemented, and there is no reason to assume that only one of them is in play in the PPC.

First, since each set of basis function values is determined by sensory and postural signals, then any *a priori* estimate of the sensory and postural signals will be suitable to produce an *a priori* estimate of the basis functions:

(23) $\overline{B}_i(t+1) = B_i(\overline{S}(t+1), \overline{q}(t+1))$

(24) $\overline{S}(t+1) = V_s\hat{s}(t) + c(t);\ \overline{q}(t+1) = V_q\overline{q}(t) + c(t)$

Here $\overline{S}(t)$ and $\overline{q}(t)$ are *a priori* estimates, at time *t*, of the sensory and postural signals. The functions V_s and V_q are functions that describe how the sensory states and postural states evolve over time (they are sensory and postural versions of the function V that was used by the KF to model the process's dynamics).

The attentive reader will have noted that all of this can be accomplished by mechanisms already discussed in previous sections: the *a priori* and *a posteriori* estimates of postural signals just are signals concerning the configuration of the body, and are supplied by the musculoskeletal emulator, the same one used for motor imagery, motor planning and so forth; the sensory signals are signals concerning the information incoming from the various sense organs, and are supplied by modality-specific emulators of the various senses (for

example, the Duhamel et al (1992) result that I analyzed as a visual image emulator in section 2.3). In other words, emulators already argued to be in play on independent grounds supply the information required to produce *a priori* estimates of the basis function values associated with a stimulus's spatial location.

A second method involves mechanisms that have not already been introduced. This would be a mechanism that simply evolves the basis function values themselves over time in accordance with information about how the spatial features of perceived objects typically evolve over time:

$$(25)\ \overline{B}_i(t+1) = V_B \hat{B}_i(t) + c(t)$$

Here V_B is a function that describes how the basis function values evolve over time, which maps the way that the spatial features of represented objects evolve over time.

At the same time that an *a priori* estimate of the basis function values is produced as per (23) and/or (25) at time *t*, a purely bottom up process – a measurement inverse – produces a set of basis function values that are what the sensors say about the spatial features of the environment at *t*. The two are combined in the way described in Section 5.2 order to produce an *a posteriori* estimate of the spatial features of the environment at *t*:

$$(26)\ \hat{B}_i(t) = \overline{B}_i(t) + K[\overline{B}_i(t) - \breve{B}_i(t)])$$

This is exactly the procedure already discussed in Section 2 as an amodal emulation of the spatial features of the environment. The current *a posteriori* estimate of the basis function values is the current *a priori* estimate plus a correction, which is the difference between the

a priori estimate and the values based solely on the observed signal (the sensory residual) multiplied by a gain term *K*.

To this point, the synthesis of emulation theory and the basis function approach to spatial representation has been limited to temporally punctate case of purely spatial representation. In this section I will expand this framework in order to explain the implementation of *spatiotemporal trajectory* representation. The generalization in the case of the emulation theory has already been discussed in Section 3. Now what remains is to state the basis function model in such a way that it can be implemented in a trajectory emulation system.

A conceptually simple generalization is entirely straight-forward. The production of smoothed and predicted estimates of both the sensory and postural signals over the temporal interval [*t-l, t+k*] can yield the set of basis function values subserving the representation of the spatial trajectory of the stimulus over that interval. If we let $\hat{B}_{i,a/[b]}$ be the estimate of the *i*th basis function value at time *b*, produced at time *a*, then this ordered set can be expressed as:

(27) ($[\tilde{B}_{1,[t-l]/t}, \ldots, \tilde{B}_{n,[t-l]/t}], \ldots, [\hat{B}_{1,[t]/t}, \ldots, \hat{B}_{n,[t]/t}], \ldots, [\overline{B}_{1,[t+k]/t}, \ldots, \overline{B}_{n,[t+k]/t}]$).

This simple generalization posits, for each time step in the interval spanned by the trajectory estimation emulator, a separate set of basis function values, each based upon the smoothed ($\tilde{B}_i$), filtered ($\hat{B}_i$), or predicted ($\overline{B}_i$) basis function value estimates corresponding to that time step. While this generalization is conceptually simple, it is also conceptually inelegant. A slightly more sophisticated generalization is not only more aesthetically comely,

but interfaces with other elements of the basis function model better – and more realistically – than the simple generalization.

Given that there are estimates of the sensory and postural states throughout the interval available, it is not necessary to produce a separate set of basis function values for each time step. Rather, a single set of basis function values can be produced based on the entire interval estimates of the sensory and postural signals. This will require the use of more complicated basis functions, of course, but I won't go explore the details of that here. If we let $\tilde{s}_{[t-l,\ t+k]/t}$ be the smoothed estimate, produced at time *t*, for the *trajectory* of sensory states over the interval [*t-l, t+k*] (and use similar notation for postural estimates), then we can define a single vector of basis function values as:

(28) $(\tilde{B}_1(\tilde{s}_{[t-l,t+k]/t},\ \tilde{q}_{[t-l,t+k]/t}),\ \ldots,\ \tilde{B}_n(\tilde{s}_{[t-l,t+k]/t},\ \tilde{q}_{[t-l,t+k]/t})$

Or yet more notationally conveniently as:

(29) $(\tilde{B}_{1,[t-l,t+k]/t},\ \ldots,\ \tilde{B}_{n,[t-l,t+k]/t})$

This is not merely an exercise in notational parsimony. We must keep in mind the point of these basis functions, which is to guide motor behavior by combining with sets of coefficients appropriate for those behaviors. In the static case we had a motor behavior, such as a *left-hand grasp* of stimulus *a*, determined by something like (12) and (13), repeated here as (30) for convenience:

(30) $M^{a,g} = (m_1^{a,g}, m_2^{a,g}, \ldots, m_p^{a,g})\ ;\ m_j^{a,g} = \sum_{i=1}^{n} g_{i,j} B_i^a$

This was fine for a stationary object, for in such a case the fact that the motor behavior takes time can be ignored. The basis function values associated with the egocentric location of the object will not themselves change during the course of the motor behavior. But in the temporally non-degenerate case this cannot be ignored. Grasping a moving object requires movement of the hand not to where the object *now is*, but to where it *will be*. And if there are different sets of basis function values for each time, then there will be different sets of basis function values associated with the object's location now, and its location at the time of the grasp.

So why not just let each motor behavior be determined as in the static case, but with the right set of temporally punctate basis function values – the ones that correspond to the location of the object at the anticipated time of the grasp? If this could be done, then there would be no reason to prefer (29) over (27). The problem is that which time in the future is the appropriate one for timing the grasp depends on the object's trajectory. If the object is moving very quickly then the grasp will have to be sooner than a case where the object is moving very slowly, even if the spatial location of the grasp is the same in both cases. If the trajectory is represented by an ordered set of sets of basis functions, then some other mechanism would have to have access to this set, and determine which of them is the appropriate time of a grasp interception, and then indicate to the motor system which set of basis functions it should use to combine with the behavior's coefficients.

While this could work in principle, the approach cuts completely against the conceptual grain of the basis function model. The point of this model is to explain how combinations of sensory and postural signals combine so as to organically guide motor behavior without the need for intervening levels of representation or micromanagement. The simple generalization as expressed in (27), when we attempt to reintegrate it with the behavioral

output part of the model, introduces exactly such intermediaries – a system that examines the set of basis functions, somehow picks one out as appropriate, and then feeds that one to the behavior-appropriate coefficients.

The more sophisticated generalization maintains the elegance of the static case. A motor behavior can still be associated with a set of proprietary coefficients that combine with a set of basis functions. A grasp will be produced by something like

(31) $M^{a,g} = (m_1^{a,g}, m_2^{a,g}, \ldots, m_p^{a,g})$;

(32) $m_j^{a,g} = \sum_{i=1}^{n} g_{i,j} B^{a}_{i,[t-l,t+k]/t}$

Here (31) is just as before: the motor command $M^{a,g}$ to execute a left-hand grasp of moving stimulus *a* is a vector each of whose elements is described in (32). This is where the difference is. Each of these elements $m_j^{a,g}$ is a linear combination of basis function values coding the estimate for the target stimulus's trajectory over the window's interval, not simply its location at one time.

Because this single set of basis functions of the form $B^{a}_{i,[t-l,t+k]/t}$ contains all the relevant information about the object *a*'s trajectory over the represented interval (not just its location at the present time), the values these basis functions yield on any given occasion are capable in principle of combining with a set of behavior-specific coefficients to produce an appropriate motor output, even for moving stimuli.

6 Discussion

It will be noted that I advertised my goal as providing a unified neural information processing structure for the representation of behavioral space, behavioral time, and objects. And while I have had sections on space and time, objects seem to have not made an appearance. I will close with a few words about objects.

Though it has been implicit, objects have been addressed all along. The behavioral-spatial representation, both in temporally punctate and temporally extended form, is built around behavioral objects as defined in Section 1. The basis functions are functions of sensory and postural signals, and the sensory signals are, in the first instance, signals from some *stimulus* – such as a light that is detected on the retinae. And the basis functions' entire purpose is to support a linear decoding via a motor-behavior-specific set of coefficients to produce a bodily action that is directed upon *that stimulus* – such as grasping or foveating the seen object. The stimulus, the entity that is the accusative of both perception and action, is the behavioral object that is represented by the basis function values.

Now of course this is not an object in the usual sense of 'object' beloved of philosophers. The objects playing a role in my account could be rocks or bugs or branches, but could just as well be shadows, or luminance edges, or holes. There is little conceptual room at this stage of representation between a location in behavioral space and a behavioral object. They are 'objects' in the sense of a potential focus of perception and action.

I believe that the capacity to represent behavioral-space, -time, and –objects is a very fundamental feature of cognition and mentality. Surely there are more basic

functions of the nervous system, functions that pre-date those I have discussed here. There is regulation of internal processes, managing of reflexes, and central pattern generators. But my hunch is that if we are interested in those nervous system functions that are fundamental to cognition and mentality, then the capacity to represent an environment of actionable objects is about as fundamental as it gets.

Acknowledgements: I would like to thank Lisa Damm, Holly Andersen, and participants in seminars on temporal representation I taught at the University of Pittsburgh in 2004, and UC San Diego in 2005 for feedback on various parts of this project in various formats. I am grateful to the McDonnell Project in Philosophy and the Neurosciences, and the Project's Director Kathleen Akins, for financial support during which this research was conducted.

References:

Blakemore, S. J., Goodbody, S. J. & Wolpert, D. M. (1998) Predicting the consequences of our own actions: The role of sensorimotor context estimation. *The Journal of Neuroscience* 18(18):7511–18.

Denier van der Gon, J.J. (1988). Motor Control: Aspects of Its Organization, Control Signals and Properties. in Wallinga et al. (eds), *Proceedings of the 7th Congress of the International Electrophysiological Society.*

Desmurget, Michel, and Scott Grafton (2000). Forward modeling allows feedback control for fast reaching movements. *Trends in Cognitive Sciences* 4(11):423-431.

Duhamel, J.-R., Colby, C. & Goldberg, M. E. (1992) The updating of the representation of visual space in parietal cortex by intended eye movements. *Science* 255(5040):90–92.

Eliasmith, C. & Anderson, C. (2003) *Neural engineering: Computational, representation, and dynamics in neurobiological systems.* MIT Press.

Getzmann, Stephan, Jörg Lewald, and Rainer Guski (2004). Representational momentum in spatial hearing. *Perception* 33(5):591-599.

Grush, Rick (2000). Self, world and space: the meaning and mechanisms of ego- and allocentric spatial representation. *Brain and Mind* 1(1):59-92.

Grush, Rick (2004). The emulation theory of representation: Motor control, imagery, and perception. *Behavioral and Brain Sciences* 27(3):377-442.

Grush, Rick (2005). Internal models and the construction of time: generalizing from *state* estimation to *trajectory* estimation to address temporal features of perception, including temporal illusions. *Journal of Neural Engineering* 2(3):S209-S218.

Houk, J.C., Singh, S.P., Fischer, C., and Barto, A. (1990). An adaptive sensorimotor network inspired by the anatomy and physiology of the cerebellum. In Miller, W.T., Sutton, R.S. and Werbos, P.J., eds. *Neural Networks for Control*. Cambridge, MA: MIT Press.

Ito, Masao (1970). Neurophysiological aspects of the cerebellar motor control system. *International Journal of Neurologv,* 7:162-176.

Ito, Masao (1984). *The cerebellum and neural control*. New York: Raven Press.

Jeannerod, M. (2001) Neural simulation of action: A unifying mechanism for motor cognition. *NeuroImage* 14:S103–S109.

Johnson, Scott H. (2000a). Thinking ahead: the case for motor imagery in prospective judgements of prehension. *Cognition* 74 (2000) 33-70.

Kalman, R.E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(d):35-45.

Kalman, R., and Bucy, R.S. (1961). New results in linear filtering and prediction theory. *Journal of Basic Engineering* 83(d):95-108.

Kawato, Mitsuo (1999). Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology* 9:718–727.

Dirk Kerzel, Dirk (2002). A matter of design: No representational momentum without predictability. *Visual Cognition 9*(1/2):66–80.

Krakauer, J. W., Ghilardi, M.-F. & Ghez, C. (1999) Independent learning of internal models for kinematic and dynamic control of reaching. *Nature Neuroscience* 2(11):1026-31.

Mach, Ernst (1896) *Contributions to the analysis of sensations*. Open Court Publishing

Mehta, B & Schaal, S. (2002) Forward models in visuomotor control. *The Journal of Neurophysiology* 88(2):942-953.

Mel, B. W. (1986). A connectionist learning model for 3-d mental rotation, zoom, and pan. In *Proceedings of Eighth Annual Conference of the Cognitive Science Society,* 562–571.

Mel, B. W. (1988). MURPHY: A robot that learns by doing. In *Neural information processing systems,* 544-553. New York: American Institute of Physics.

Pouget, Alexandre, Sophie Deneve and Jean-René Duhamel (2002). Nature Reviews: Neuroscience 3:741-747.

Senior, Carl, Jamie Ward and Anthony S. David (2002). Representational momentum and the brain: An investigation into the functional necessity of V5/MT. *Visual Cognition 9*(1/2):81–92.

von Helmholtz, H. (1910) *Handbuch der Physiologischen Optik*, vol. 3, 3rd ed., ed.

A. Gullstrand, J. von Kries and W. Nagel. Voss.

Wolpert, D.M., Ghahramani, Z. and Jordan, M.I. (1995) An internal model for sensorimotor integration. *Science* 269:1880–1882.

Wolpert, Daniel M., Zoubin Ghahramani and J. Randall Flanagan (2001). Perspectives and problems in motor learning. *Trends in Cognitive Sciences* 5(11):487-494.