In Garuav Sukhatme, Ed., *The Path to Autonomous Robots*, Springer Verlag, 2008

Neurorobotics Primer

M. Anthony Lewis and Theresa J. Klein

Abstract Neurorobots use accurate biological models of neurons to control the behavior of biologically inspired or biorobots. While highly simplified neural models (e.g. ANN) have been used in robotics, recent innovations in mathematics, science and technology have made possible the real-time control of robotics by highly complex, biologically realistic neural models. In this chapter we present a self-contained primer on Neurorobotics which serves to give an integrated view of the possibility of this nascent trend with important ramification in science and technology. In particular, we argue that neurorobotics will replace the conventional computer simulation for many neural-system models. Further, within a relatively short-time it will be possible to simulate 10^{11} neurons in real-time, roughly the number of neurons in the human brain, on a desktop computer. If we can understand how to harness this power, and productize it, we will be able to create robots of incredible complexity and behavioral richness.

1 Introduction

I can not believe that the brain computes Jacobians - George A. Bekey circa 1992

Biorobots are artificial devices built to test hypotheses in biology. Examples include work by Webb [36], Beer [12, 7] Lewis [29, 28, 27] and many others. Biologically inspired robots, on the other hand, are robots that use biology as metaphors to solve practical problems in robotics. Examples include work by Brooks [9, 10] and Arkin [3]. A neurorobot is a special kind of biorobot that explicitly uses models

M. Anthony Lewis
Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 85721, e-mail: malewis@ece.arizona.edu

Theresa J. Klein
Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 85721, e-mail: tjk@ece.arizona.edu

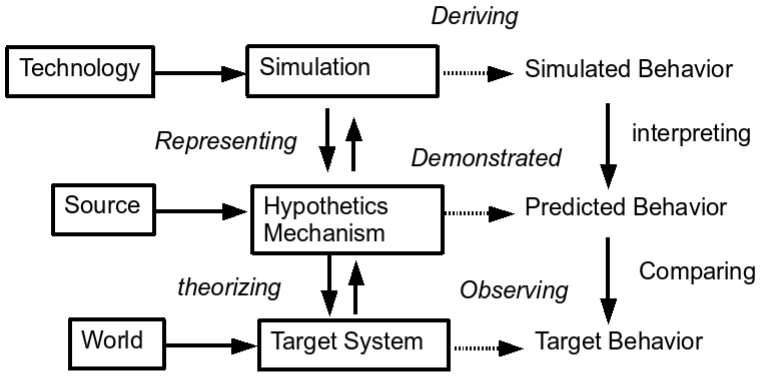

Fig. 1 Integration of simulation studies with the scientific processes. Redrawn from [36].

of biologically realistic neurons as its computational substrate. In a neurorobot, algorithms are replaced by extremely high dimensional dynamical systems based on neural sub-units. These subunits range in complexity from leaky-integrator models to integrate-and-fire models, Hodgkin-Huxley neurons and the recently discovered model by Izhikevich [22]. Neurorobots will one day solve real-world problems, thus filling a dual use as both a biorobot and an biologically inspired robot.

Neurorobots, an outgrowth of biorobots, may find important commercial applications in biologically inspired robots in the near future. For a comprehensive overview of biologically inspired robots, see [8].

1.1 Neurorobots and the Scientific Method

Can neurorobotics help us better understand the brain? Webb [36] has proposed a model for integrating the traditional view of the scientific method with modern technology, in particular, simulation. Referring to Fig. 1 the process of scientific investigation begins with the identification of a target system that we are interested in understanding. The scientist may theorize a mechanism that will explain the observed target behavior. The scientist may use various sources for inspiration for this mechanism. In biomechanics, it may be a cost function that we presume the human body optimizes. In the example cited by Webb, the idea of a Fourier transformation

may be used as a source for a hypothetical mechanism of how the cochlea transforms sound into neural impulses.

To understand the ramification of this hypothesis, we can create a simulation that takes similar input and gives similar behavioral output. A simulation can exist solely in a computer, or, in the case of a biorobot, the system can interact with the world. As computers are, of course, capable of fantastically faster mathematical computations than the human brain, one may think of a simulation as cognitive prosthesis that helps us think more effectively about complex problems that are reducible to a well defined system of equations.

A biorobot is a simulation paradigm for understanding behavior in animals in which a computer program is allowed to gather and process raw data from the world in real-time and produce an effect on the world. A biorobotic simulation is as legitimate a scientific tool as a self-contained computer simulation. Ultimately, the difference between a simulation that runs in a simulated world and a simulation that runs in the real-world is that we can compare our physical simulations with the target system in the same environment as the organism we are investigating. *For this reason, we might say that for understanding the neural basis of behavior, a biorobot can be a more legitimate and scientifically meaningful simulation, than a computer simulation.*

1.2 21st Century Robotics: Productizing Mythology

The idea of a robot is grounded in biological inspiration. It is a search for what constitutes the "spark of life" and what distinguishes the living from the non-living. It raises the question, "how can we create artificial beings?" Automata have been built for thousands of years to amaze and entertain and make promises of slave-like machines that will serve human-kind. It was only in the 20th century, with the advent of the stored program, digital computers that we could begin to realize practical, commercially viable robotic machines. Stored program digital computers enable the following capabilities in robots:

1. *The ability to create highly complex behavior*- Complex behavior is dependent on the ability to make memories. In the case of computer programs, memories can be constructed from bistable elements integrated on a massive scale onto inexpensive chips. Complex behaviors uses a large number of non-linear processes to make decisions, compute transformation and most importantly to derive percepts. A percept is an element extracted from a sensory stream that captures some invariant property of that sensory stream [26]. We distinguish this from a linear transformation of an input stream where the original sensory input can be recovered from the transformed elements. Many different sensory configurations can create the same percept. This processing is easily and reliably simulated on digital computer.
2. *The ability to alter behavior to achieve different tasks (i.e. to be reprogrammed)*- A stored program computer can be altered in its operation by controlling the

data it operates on, controlling the mode of the code that is executed and so on. These are properties that are exceedingly difficult to achieve without modern computers.

To date, the most aggressive uses of the ability to perceive have been seen in autonomous robots and factory automation. However, at the level of consumer robots, there has been marginal ability to perceive the environment, little ability to be reprogrammed, and relatively simple behavior. Thus, the consumer robots have not fully utilized the power of the modern computer. Why?

Productization of highly complex machines is highly problematic. In general, the *Occam's razor principle of productization is that the simplest device is the one that will most likely be built.* Thus, introducing complexity is contrary to the parsimony of productization.

From a product perspective the industrialist must answer several question:

1. What advantage does complexity give us? Does it make the product better or worse?
2. How does one test such a machine to know if it is working?
3. How can we tell if a learning system is learning?
4. If a system requires thousands of cycles to significantly change behavior, how can we test such a capability quickly?

The problem will be complicated even further in the case of a neurorobot. At the end of the factory line, tests must be done to confirm the behavior of the robot being built. Suppose that a robot system had the complexity of a human being. Much as the human mind is 'programmed' by a sequence of experiences, neurally based robots will be programmed as well by experience. Their resulting behavior will be difficult to predict. It might take 16 years to determine if the robot would be a juvenile robotic delinquent or is on track to be a brain surgeon! Debugging, at the system level, will require new automated tools that can analyze the system using invasive techniques, or perhaps an entire field or robot psychology.

For these reason neurorobotic technology will be very limited in market acceptation, or indeed other highly intelligent machines, until we can solve the practical problem of testing such systems, anddemonstrating that these systems will yield clearly superior performance— performance that can be achieved in no other way more simply. One exception to the rule on market acceptance has been the robots built by Mark Tilden. Tilden's first prototype of the 'Robosapien' line of robots was built at the Neuromorphic engineering workshop in Telluride, Co, in July, 2001, see Fig. 2. Some 5-10 million Robosapien robots and its derivatives have been sold. The essence of Tilden's idea was to use analog oscillators to generate the movement of bodies. His robots, while remarkably sophisticated as a consumer product also exhibit a *je ne sais quoi* quality of life-like movement that has not been duplicated by others. His prolific experimentation in the 1990's lead to the development of the first prototype Robosapien in July of 2001.

Tilden's robots were acceptable as products but had only limited perceptual capability due to the sever cost constraints of the toy market (not, indeed, by any limitation of Tilden's imagination).

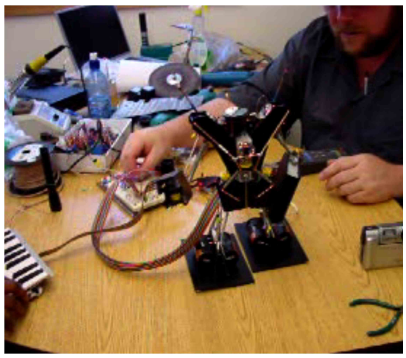

Fig. 2 The birth of a revolution in biologically inspired robots. (A) The first Robosapien prototype was created in July 2001 in Telluride Colorado by Dr. Mark Tilden. Tilden successfully incorporated many principles of biological motion into the a commercially viable product. (B) November, 2005 People's Republic of China, the second version of RoboSapien, RS-V2 is being assembled, tested, and readied for shipment. Between 5-10 million robots, based on this design, have been sold [37].

To date, Tilden's commercial efforts have been limited to the consumer entertainment market. It is evident that neurorobotics greatest impact will be both in science and in the service robotics market, or robots that perform useful work.

Productization of a myth is exceeding difficult, but progress is being made.

1.3 Computational Substrate

Neurorobots, by definition, must function in real-time. As a result, computational speed and I/O bandwidth constrain the level of complexity now achievable in neurorobotic brains. The substrate for computation is important. Computation used in Neurorobot may be based on computer simulation or may be directly implemented as Neuromorphic chips.

1.4 Neuromorphic chips

A neuromorphic chip is an analog VLSI device that implements a specific neural-system computation or class of computations. Historically, these neuromorphic chips differ from their digital cousins in two essential ways. First, these systems rely on analog computations that directly implement operations such as integration, differentiation, multiplication and addition with the purpose of efficiently implementing neuronal networks. Second, digital computers that we are familiar with

are stored program computers. They are reconfigurable and capable of running an extremely wide range of computations. Neuromorphic devices have lacked this capability. Neuromorphic chips trade speed and power efficiency for flexibility and development ease. Each chip may have only a handful of people in the world that understand how to use a given device. This limits their dissemination.

1.5 Graphics Processing Units

Another approach to computation in Neurorobots is to use conventional Central Processing Units (CPU) and Graphics Processing Units (GPUs) for processing. GPUs are leading a revolution in computing. GPU chip concentrate most of their real-estate on computation rather than memory as in traditional CPUs. This allows 16 computational cores to be placed on a single chip. What is more, these chips are inexpensive. The retail price for a state of the art Pentium CPU with 4 cores is roughly the same as a GPU card with some 240 cores. GPU processing have doubled in speed consistently every year [2]. Today, a GPU is the most important computational element in a high performance computer, well overshadowing the raw computational power of the CPU. As stated, GPUs are optimized for computation, not memory. This make GPUs an ideal host for complex biophysical algorithms. GPUs will allow the use of models with dynamic synapses, multiple compartments and other advanced features that, to date, have not been incorporated into neurorobotics due to real-time requirements. This revolution in GPU technology will have a major impact in neurorobotics, where computational dense models need to be constructed.

1.6 Purpose of this Chapter

It is clear that neurorobotics is on the cusp of a revolution due to the aforementioned technological advances. The dissemination of neurorobotic experimentation is limited due to the relatively small group of individuals in the world that are cross trained in robotics and neurocomputation. This article is meant to be a primer in neurorobotics, bringing together fields of robotics, neurocomputation, and computational technology.

It is the hope that this chapter provides the first integrated view of Neurorobotics. This chapter is based on a lecture given at the 2008 Telluride Neuromorphic Engineering Workshop sponsored by NSF.

2 Classical Robotics

We define classical robotics as the core topics featured in text books by Craig [13], Paul [31], or Asada and Slotine [4]. The classic topics consists of (1) configuration space (2) kinematics and inverse kinematics (3) Differential motion (4) Statics (5) Dynamics and (6) trajectory generation. In recent years Baysian probability has come to the forefront as well, and might be considered as apart of the core topics in robotics.

2.1 Configuration space

The starting point of classical robotics is configuration space. A robot is an articulate system with numerous rotational and prismatic joints. If the position of each joint is fixed, the system will be in a particular configuration. The set of all possible joint positions of the robot is the configuration space. Picking a point in configuration space defines the position of each point on the robot. You might consider a luxo lamp. By fixing the rotation of the base, the rotation of the light, and the joints in between, the robot has a well defined position or configuration. The configuration space point also sets the position of every point on the surface and in the interior of the robot. The former being important when considering collisions with the environment, and the latter when considering the mass property of the robotics system.

Often, we wish to know how each of these points correspond to a world frame of references, i.e. in Cartesian space. Cartesian space is a simple, generic way of specifying the location of any point on the robot. Cartesian space considers 3 mutually orthogonal axes that define a coordinate system in space. You might think of the corner of a room where the vertical seam is the z axis, and the horizontal wall-floor seams are the x and y axes. Kinematics can help us determine, systematically, the relationship between configuration space and Cartesian space.

2.2 Kinematics

Refer to the leg in Fig. 3. This leg is composed of two segments of length L_1 and L_2. The position of the foot relative to the hip can be computed if we also know the angle between the axis of each segment and the ground. These angles are given by θ_1 and θ_2.

Thus the hip's position can be computed as:

$$x_e = L_1 \cos(\theta_1) + L_2 \cos(\theta_2) \tag{1}$$

$$y_e = L_1 \sin(\theta_1) + L_2 \sin(\theta_2) \tag{2}$$

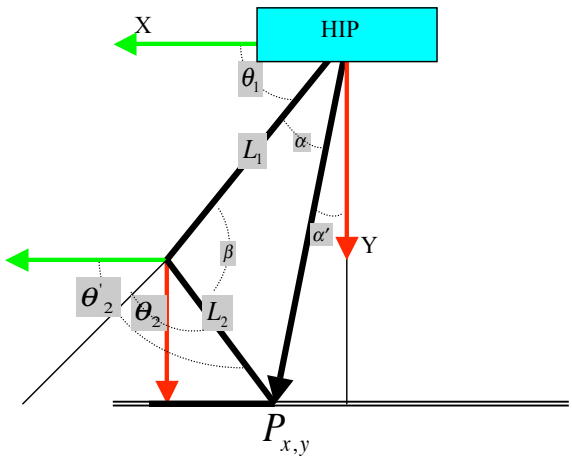

Fig. 3 Two degree of freedom leg. We consider the application of basic techniques in classical robotics to a simple, one legged robot.

The only subtly here is that we have purposefully placed joint angles θ_1 and θ_2 in the frame of reference of the foot. This makes the addition particularly easy. In the typical case, however, the robot may have a sensor on each joint indicating the angle of the one link relative to another link. For the current, planar case, we can make the substitution:

$$\theta_2 = \theta_1 + \theta_2' \tag{3}$$

where θ_2' is the angle of the second link as would be measured by a sensor on the robot. We make the simplifying assumption that the hip is always level.

Cartesian space has three axes and can be augmented with a rotation to create the pose of a portion of the robot. For any point in configuration space, there is a unique point in Cartesian space. Further, this map is smooth at almost every point. By systematic analysis of the geometry of a robot, we can compute this map. This is the robotics kinematics.

The magic of kinematics is that if we specify the configuration of our robot using standard notation and apply a standard set of rules, we can turn the metaphorical crank and determine solutions to forward kinematics, that is the mapping from configuration space to world coordinates.

The main point here is that *forward kinematics can be found by a well define procedures*. Inverse kinematics, or going from Cartesian coordinates to configuration space requires some insight. Referring again to Fig. 3, we can image that we are given a vector $P_{x,y}$, specifying were we would like to place the foot relative to the hip. The length of this vector is just: $p = \sqrt{x^2 + y^2}$. We now have a triangle with three sides of known length. Applying the law of cosines we have:

$$\alpha = \cos^{-1}\sqrt{\frac{L_1^2 + p^2 - L_1^2}{2L_1 p}} \tag{4}$$

$$\beta = \cos^{-1}\sqrt{\frac{L_1^2 + L_2^2 - p^2}{2L_1 L_2}} \tag{5}$$

$$\alpha' = \arctan 2(y,x) \tag{6}$$

$$\theta_1 = \frac{\pi}{2} - \alpha' - \alpha \tag{7}$$

$$\theta_2' = \pi - \beta \tag{8}$$

Thus we find that we can define the angles. In general, as more links are added the inverse kinematics solution becomes exceedingly difficult and requires an increasing level of skill to find a solution.

2.3 Differential Motion

The mapping from configuration space to Cartesian space is continuous and it is also differentiable. Thus may wish to calculate:

$$dx = f(d\theta) \tag{9}$$

This relates small changes in θ to small changes in x. The two are related by a *Jacobian Matrix*:

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} \frac{df_x}{d\theta_1} \frac{df_x}{d\theta_2} \\ \frac{df_y}{d\theta_1} \frac{df_y}{d\theta_2} \end{bmatrix} \begin{bmatrix} d\theta_1 \\ d\theta_2 \end{bmatrix} \tag{10}$$

This can be summarized as :

$$dx = J(\theta)d\theta \tag{11}$$

Using equations (1) and (2), we can compute as:

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} -(L_1\sin(\theta_1) + L_2\sin(\theta_1+\theta_2)) & -(L_2\sin(\theta_1+\theta_2)) \\ (L_1\cos(\theta_1) + L_2\cos(\theta_1+\theta_2)) & (L_2\cos(\theta_1+\theta_2)) \end{bmatrix} \begin{bmatrix} d\theta_1 \\ d\theta_2 \end{bmatrix} \tag{12}$$

What is the relationship between small changes in Cartesian coordinates and small changes in configuration space? It is simple:

$$J(\theta)^{-1}dx = d\theta \tag{13}$$

Well, it is simple if it exists. We can find the Jacobian inverse as long as the determinant of $J(\theta)$ is nonzero, i.e. $det|J(\theta| \neq 0$. That is, we are checking for the condition:

$$0 = L_1 L_2(\sin(\theta_1 + \theta_2)\cos(\theta_1) - \cos(\theta_1 + \theta_2)\sin(\theta_1)) \tag{14}$$

This happens when the robot's leg is held straight out, i.e.: $\theta_2 = 0$ and :

$$0 = L_1 L_2(\sin(\theta_1)\cos(\theta_1) - \cos(\theta_1)\sin(\theta_1)) \tag{15}$$

which is always true. Intuitively, when the leg is stretched out, we cannot make a small movement, $d\theta_1$ or $d\theta_2$ that will result in the end of the leg moving away from the hip radially.

2.4 Statics

Suppose we wish to find a relationship between the force produced at the foot of the robot, and the torques exerted around the knees and hip.

This relationship is given as:

$$J^T(\theta)\mathbf{F} = \tau \tag{16}$$

where F is a vector of forces at the robot foot, in our example, and τ is a vector of torques placed at the knee and hip. We omit the proof here, but the reader is referred to [4]. This equation might be useful for determining the expected torque on actuators given that the leg is in different configurations. Torque, τ is just:

$$\tau = \mathbf{r} \times \mathbf{F} \tag{17}$$

Of course we can find the inverse relationship as well:

$$F = (J^T)^{-1}(\theta)\tau \tag{18}$$

As noted above, the Jacobian will not have an inverse when the leg is straight out. The reader can verify that in this case we will not be able find a torque that will cause the leg to produce a force straight out from the hip. Intuitively, the reader knows this is true. Try standing on your heels with legs fully extended, and then try to jump! Your heel is supporting your weight, but that weight does not cause torques on hips or knees in this example.

2.5 Redundancy

The most often case in biologically systems is that there are a large number of ways of reaching the same point. As the reader can readily verify, for the simple case of two links, there are two possible configuration for knee and hip for the foot to reach a given point (even if one might represent a painful hyperextension of

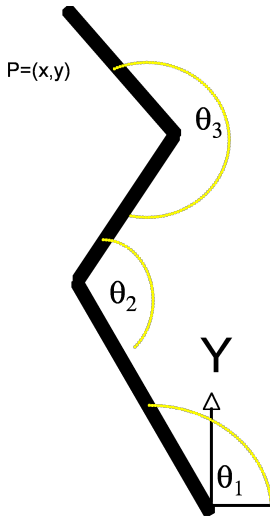

Fig. 4 Illustration of redundancy. In biological systems, redundancy is the norm. Anchor your feet and grab a door handle. You should be able to move your wrist, elbows, torsos in many different ways while keeping your feet anchored and still holding the door handle. Make sure no one else is around when you try this experiment.

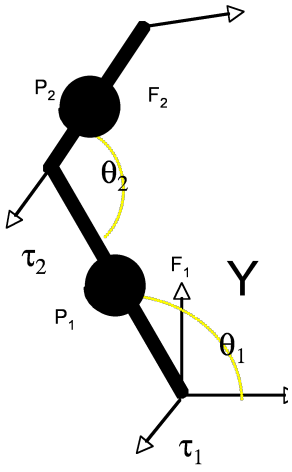

Fig. 5 Model of dynamic system. Here we assume that the mass are concentrated as point masses (illustrated as spheres). Note that when computing Jacobian, we are interested in the distance to and velocity of these masses, not the endpoint of the robot.

the knee). Figure 4 gives an example of a simple, redundant system where infinite configuration correspond to a given target point.

Humans take advantage of redundancy when they control they posture during movement. Depending on the task, a person might change the configuration of his/her body to be able to exert more force, or remain balanced etc.

2.6 Dynamics

Real systems have mass and evolve through time, that is, they have *dynamics*. While the dynamics are simple, only in the case of a 1 or perhaps 2 link system, the methodology that is used to write down closed form dynamics is common to both a 1 link as well as a Nth order system. Referring to Fig. 5, first we form the *Lagrangian*. The Lagrangian summarized the dynamics of a system.

$$L = KE^{TOTAL} - PE^{TOTAL} \tag{19}$$

That is, the total kinetic energy minus the potential energy.

The potential energy of a particle is:

$$PE = mgh \tag{20}$$

If you lift an object in a constant gravitational field, with magnitude g, one can use eqn (20) to compute the potential energy of this object.

If you have many objects, or in our case, more than one link, you can sum the potential energy. For a mass, we consider the height of the center of gravity (mass). The total potential energy is thus:

$$PE = g\sum_i height_i m_i \tag{21}$$

We may with to use some of the machinary developed above to help us compute the height of the mass above the ground, which we can do using forward kinematics, and the velocity of these point masses, which we can do by knowing the joint velocities and using the Jacobian to find the point mass velocities. Eqn. (11) implies:

$$\dot{x} = J(\theta)\dot{\theta} \tag{22}$$

Let us assume that the mass of each link of the leg can be considered as a single, point mass, then the kinetic energy is:

$$KE = \frac{1}{2}mv^2 \tag{23}$$

where m is the mass of the particle, and v is its velocity. One way of looking at the kinetic energy is that it summarizes the net work done on the particle up until that point.

The kinetic energy of the leg is given as:

$$KE = \frac{1}{2}\dot{\mathbf{x}}^T M\dot{\mathbf{x}} \tag{24}$$

which, using eqn (22) we can re-write as:

$$KE = \frac{1}{2}\dot{\theta}^T J(\theta)^T MJ(\theta)\dot{\theta} \tag{25}$$

here M is a matrix with diagonal elements equal to the mass of each link. In our case, M is to dimensional.

We can now compute the dynamics using a formula:

$$\tau = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} \tag{26}$$

This formula relates the torques at each joint to accelerations, and gravity's pull on each link. At low velocities, gravity is the dominate contributor to torque, far outweighing any other dynamic effects.

2.7 Trajectory Generation

We must move the limb from point A to point B. If we suddenly accelerate the limb as quickly as possible to the goal position, we will run into problems. First, the exact trajectory of the limb will be ill-defined. We may collide with the environment, or the robot may even colide with itself. Second, we induce a lot of wear and tear on the robot's joints and gears. It is better to choose a smooth trajectory that will allow us to avoid obstacles, and gently accelerate.

The basic idea is to generate a spline from a series of knot points. See Fig. 6. The spline has additional constraints placed on it including the position, velocity and acceleration of beginning and end points. The knot points can be positions to avoid obstacles. In classical robotics, the robot control system ensures that trajectories are tracked well.

Note that in classical robotics, we were concerned with controlling the movement of a manipulator. For tasks such as welding is critical that we can achieve good trajectory tracking.

To do this, certain design features are incorporated into the robots. First, we might use high gear reduction motors, perhaps 1:100 or more. This means the robot's joint might revolve 1 times per 100 revolution of the motor. High gear reduction has certain benefits. These benefits include good disturbance rejection: dynamic force perturbations are reduced dramatically as they are reflected back to the motor. Second, we can run the robot at high RPM. In general, electric motor achieve their greatest power output at high RPM. Finally, we would include a stiff feedback control loop. Feedback is the process of comparing the actual position of the a joint to the desired position of the joint and adding or subtracting torque to achieve a reduced error in the next control cycle. In general, disturbances of the environment are repressed.

In a walking machine, if a leg strikes an obstacle while it is in swing phase, two thing can happen: (1) The leg moves out of the way, around the obstacle or (2) the induced torque caused by the strike destabilizes the robot and causes it to stumble.

Nature uses two methods to keep from stumbling. First, in swing phase the leg is very complaint. That is, its feedback gain is very low. Thus, when it strikes, the strike causes a motion of the leg, not the entire robot. Second, animals have reflexes. These

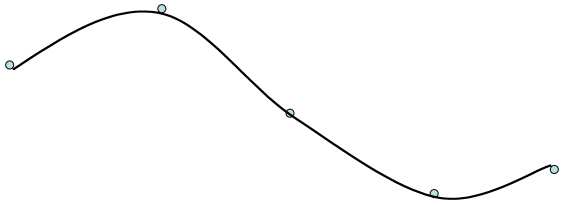

Fig. 6 Trajectory Generation. Trajectories can be specified by a series of 'knot points.' Trajectory generation is the process of finding a smooth path, a spline, through these knot points, while satisfying some endpoint criteria such as position, velocity and acceleration.

reflexes cause the leg to move back, up and over the obstacle. This is an automated response.

In classical robotics we do not consider reflexes except in the case of careful, slow movements,i.e. guarded moves. We do not alter the stiffness of the control loop during trajectory execution.

The next thing to notice about the classical approach to trajectory generation is that knot points are selected by task criteria and not by energy criteria. We have shown that a leg can generate its own trajectory by allowing passive movement of the leg during swing phase [28]. Thus the inherent dynamics of the system selects the best trajectory and takes a minimum energy path.

The final thing to notice is that trajectories generation by knot point do not explicitly take into account the cyclic nature of locomotion. Locomotion is highly periodic. We touch on Central Pattern Generator Theory below as an alternative to the classic robot knot-point paradigm.

2.8 A pause to reflect

The power of analytical methods are illustrated here by the ability to determine the overall endpoint of the robot given configuration of subcomponents, i.e. the links. We take can take advantage of some simple properties of vector spaces to allow the addition of subcomponents to find the overall forward kinematics, for example. For inverse kinematic mapping we cannot do this and it is therefore a more difficult in practice.

We note that the Jacobian is a particularly handy transformation. It allows us to tell when the system is capable of generating force at the foot by applying torques at the ankle. It can tell us if the arm has reached a singularity and is therefore not capable of motion in a certain direction.

We note that nowhere in the classical method did we explicitly represent time and synchronization between robot and the environment and the robot and itself.

This mapping problem is non-trivial when we consider tasks such as interaction of any point on the skin of the robot and the world. Here, the kinematic-inverse kinematic computations are so numerous that they cannot be calculated by hand. We must resort to massive computer computations. We note that much of what we uncovered are ways of creating mappings from one space to another. This can easily be learned using even traditional neural networks.

While the purest may argue that the geometric approach is sufficient, its appeal wanes as the complexity of problems grown. Learning methods, including those based on neural building blocks, will become indispensable tools in the roboticists toolbox. CPG methods offer an approach to the time synchronization problem. As CPGs become better understood, they to will become important, indispensable tools.

Artificial Neural Networks have been shown to be universal function approximators. [21]. The Universal function approximator is a powerful property that can allow us to subsume much of conventional, geometric based robotics. In the next section we discuss more complex models that allow us to go beyond just simple mapping to create system of reach temporal behavior.

3 Basic Neurocomputation

Cells are the basic building block of multi-cellular organisms. Special cells exist that are electrically excitable. These cells include muscles cells and in neural cells (Neurons).

The cell membrane (Fig. 7) can be thought of as a bag containing the cellular machinery that allows an imbalance in ion concentration to occur inside versus outside the cells. Computationally we do not care about the cells DNA, mitochondria, cell nuclei etc. We do care about this ion imbalance. The cell membrane itself has a very high resistance. On either side of the cell membrane is a conductive solution, thus the cell membrane forms the dielectric of a capacitor. Ion channels penetrate the cell bilipid (composed of two layers of fat molecules) membrane layer and are highly selective to specific ion species.

3.1 Information Flows into dendrites and out of axons

Neural cells have two types of processes that are attached to the cell body. The *axon* is a long process that can transmit *action potentials*, discussed below, from the cell body to other neurons throughout the brain. These axons synapse on the dendrites of other neurons. These dendrites collect information from thousands of neurons. Each synapse contributes a bit of current to the cell and results in the alteration of the cell membrane voltage. Each synapse may have a different coupling strength. A given cell can either make it more likely or less likely that a cell will generate an action potential, fire, depending on if it is an excitatory synapse or an inhibitory synapse.

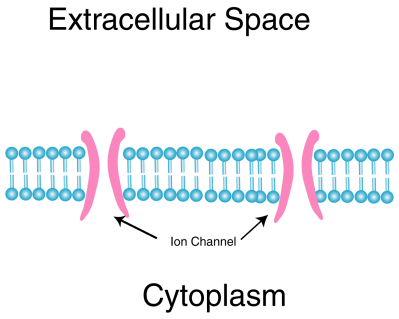

Fig. 7 The Cell membrane is composed of a bilipid insulating layer and isolate conductive electrolytes inside and outside the cell. Certain ion species, such as potassiam, calcium and sodium are particularly important to electrical activity in neurons. The cell membrane contains ion channels that are highly selective to specific ions and can be turned on and off by changes in voltage or by a special protein (ligand). The cell membrane also contains ion pumps that maintain a specific ion imbalance between the inside and outside of the cells. Opening ion channels take advantage of this imbalance to create current flows and hence alter membrane potential. Highly dynamic events called action potential encode voltage events and propagate this information to thousands of other cells.

3.2 The Neuron Cell is a Capacitor with a Decision Making Capability

The membrane/electrolyte system forms a capacitor. The electrical model of an ideal capacitor is:

$$C\frac{dV}{dt} = i \tag{27}$$

where C is the capacitance, i is a current flow, and V is the voltage potential measured relative to the inside of the neuron.

Ions flow is dirven by diffusion and voltage driving force. *Ion pumps* in the cell membrane maintain a concentration gradient for various ions. *Ion Channels* allow charged ions to flow across the cell membrane in a highly selective way. Given a ion gradient between the inside and outside cell, a diffusion gradient is set up that causes a net flow of ions from the higher to the lower gradient. As each ion particle is charged, this causes a current flow. If a voltage potential is maintained across the membrane, positive ions will be attracted to the negative side of the cell membrane. This leads to the build up of a concentration gradient.

The Nernst potential specifies the relationship between concentration gradients and the voltage potential across the membrane:

$$E = \frac{RT}{zF} ln \frac{[outside]}{[inside]} \tag{28}$$

where E is the potential, T is the absolute temperature, and R the gas constant, F is the Faraday constant. For real cells the $E_k = [-70..-90mv]$, $E_{Na} = [50mv]$, $E_{Ca^{2+}} = [150mv]$.

3.3 Neural Models capture the basic dynamics of the cell body and throw away some details

The difference in neural models comes down the to the equations specifying I in eqn (27). The trade off is between the complexity of the dynamics, the ability to analyze the systems and the ability to simulate a given model efficiently. The most well known model is the Hodgkin-Huxley model. Hodkin-Huxley empirically determined their equations [20, 19]:

$$i_m = g_L(V - E_L) + g_k n^4 (V - E_K) + g_{Na} m^4 h(V - E_{Na}) + I_s \tag{29}$$

h,n,m evolve according to complex equations, I_s are synaptic currents coming from other neurons. The equation can be understood in the following way: the values in parentheses $(V - E_x)$ are driving forces that will tend to drive the potential of the neuron to the Nerst reversal potential E_x. In front of each of these driving forces are weighting factors. These weighting factors determine which driving force will dominate the equation, and hence the equilibrium value for the cell membrane. If only g_l is active, the cell membrane will relax toward $V -> E_l$. When the cell begins to fire, after the cell membrane has increased in potential, the Na part of the equation dominates and the membrane potential shoots up toward E_{Na}. Next the potassium term turns on and resets the neuron.

In general the integration of this equation is difficult, and the analysis is extraordinarily difficult, so researchers have turned to simplified models which capture some of the features of this system. Here we focus on computationally efficient models.

3.3.1 Leaky Integrator

One of the simplest models is the so called leaky integrator. This model avoids the complex dynamics of firing altogether:

$$c_m \frac{dV_i}{dt} = -g_l V_i + I_i \tag{30}$$

$$u_i = f(v_i) \tag{31}$$

$$I_i = \sum_{i=1}^{N} w_{ij} u_j \tag{32}$$

where the V is the membrane potential, u is the average firing rate of the neuron over a small time window. $f(v)$ is a function that transforms membrane voltage into firing rate.

In steady state $\frac{dV_i}{dt} = 0$. This implies that $g_l V_i = I_i$. We can often assume that $g_l = 1$. This equation is an integrator because absent the $-V_i$ term, it is a perfect integrator. This equation is leaky because absent the input I_i, the voltage V_i "leaks" back to zero.

This model does not capture the essential spiking characteristic of neurons. All precise timing information is eliminated. The next step up in complexity is an integrate and fire model.

3.3.2 Integrate and Fire

$$c_m \frac{dV_i}{dt} = -g_L(V_i - E_L) + I_i \tag{33}$$

$$if(V > V_{thres}) -> V = V_{reset} \tag{34}$$

$$I_i = \sum_{i=1}^{N} w_{ij} s(u_j) \tag{35}$$

In the integrate and fire model, as the membrane approaches a fixed threshold, the cell 'fires' and generates a spike. It then resets the membrane voltage. This model is not sophisticated enough to build interesting models of networks controlling motors systems. We must introduce spike frequency adaptation:

$$c_m \frac{dV_i}{dt} = -g_L(V_i - E_L) + r_m g_{sra}(V - E_k) + I_i \tag{36}$$

$$\tau_{sra} \frac{dg_{sra}}{dt} = -g_{sra} \tag{37}$$

$$if(V > V_{thres}) -> V = V_{reset}, g_{sra} -> g_{sra} + \Delta g_{sra} \tag{38}$$

$$I_i = \sum_{i=1}^{N} w_{ij} s(u_j) \tag{39}$$

Here we have an adaptation terms. The variable g_{sra} functions much like a spike averager, averaging spikes over an exponential time window. As the number of recent spike increases, the $(V - E_k)$ term begins to dominate. This driving force is trying to shut the neuron off. Hence with more spikes, the neurons spikes less frequently.

3.3.3 Matsuoka Oscillator

The Matsuoka Oscillator [30] another popular oscillator which seems almost like a continuous time version of the integrate and fire model with adaptation and is given as:

$$\tau_i \frac{u_i}{dt} = -u_i - \beta f(v_i) + \sum_{j \neq i} w_{ij} f(u_j) + u_0 \tag{40}$$

$$\tau_i' \frac{v_i}{dt} = -v_i - f(u_i) \tag{41}$$

$$f(u) = max(0,u), (i = 1,2) \tag{42}$$

As can be seen, as the neuron fires more, the value v_i accumulates (almost like averaging spikes, only this neuron does not spike). As v_i increases, it introduces an inhibitory term suppressing the "firing rate" of the neuron.

3.3.4 Izhikevich Model

The Hodgkin-Huxley model is capable of a rich range of behavior, but is difficult to integrate. The leaky integrator, integrate and fire with adaptation, and the Matsuoka oscillator are easy to integrate by have less richness than the HH.

The best of all-possible-world model was discovered by Izhikevich[22] :

$$\frac{dv}{dt} = 0.05v^2 + 5v + 140 - u + I \tag{43}$$

$$\frac{du}{dt} = a(bv - u) \tag{44}$$

$$if (v > 30mv) v- > c, u- > u + d \tag{45}$$

See Fig. 8 for the range of firing behaviors that can be obtained from this model. These behaviors are controlled by parameters a,b,c, and d.

3.4 Numerical Integration

Next we turn to the practical matter of computing our equations. We can do so by numerically integrating the equations on a digital computer.

The equations for neurons that we have discussed so far are of the form:

$$\frac{dx}{dt} = f(x) \tag{46}$$

we can write:

$$dx = f(x)dt \tag{47}$$

$$\Delta x \approx f(x)\Delta t \tag{48}$$

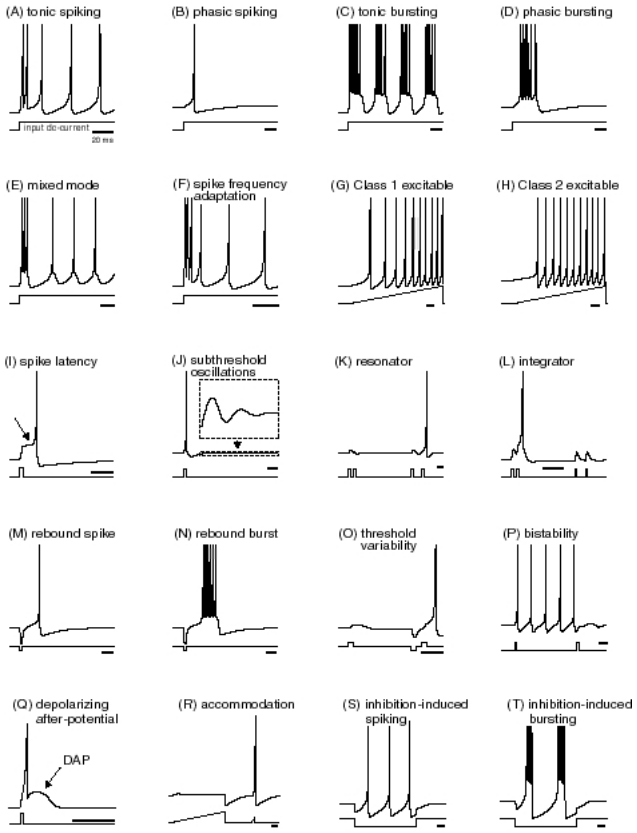

Fig. 8 Range of behavior available with the Izhikevich neural model. *Electronic version of the figure and reproduction permissions are freely available at www.izhikevich.com*

$$x_{n+1} = x_n + f(x)\Delta t \tag{49}$$

$$x_0 = x(0) \tag{50}$$

The use of eqns (49) and (50) is called Eulers Integration. Derivation and use of Euler Integration is straightforward, and they are completely adequate for most equations. However equations such as the Hodgkin-Huxely equation may benefit from a better integrator such as Runge-Kutta 4th order. The iterative equations are given as:

$$x_{n+1} = x_n + \frac{h}{6}(a + 2b + 2c + d) \tag{51}$$

$$a = f(t_n, x_n) \tag{52}$$

$$b = f(t_n + \frac{h}{2}, x_n + \frac{h}{2}a) \tag{53}$$

$$c = f(t_n + \frac{h}{2}, x_n + \frac{h}{2}b) \tag{54}$$

$$d = f(t_n + h, x_n + hc) \tag{55}$$

Runge-Kutta is more efficient than Euler Integration. While Runge-Kutta requires more evaluations per time-step than Euler Integration, we can accurately take much larger time steps using Runge-Kutta.

However, Euler is often fast enough for small networks and has the advantage that it implementation is trivial.

3.5 Building Neural Oscillators: Natures coordination and trajectory generation mechanism

Here we turn to the basic unit of movement generation in vertebrates: the oscillator. Oscillator circuits can be seen at the core of virtually all models of rhythmic motion generation. They are the basic building blocks of the so-called Central Pattern Generators, or CPG circuits. These circuits are groups of neurons (in vertebrates they are in the spinal cord) seen in animals capable of periodic movement including walking, running, swimming, flying etc.

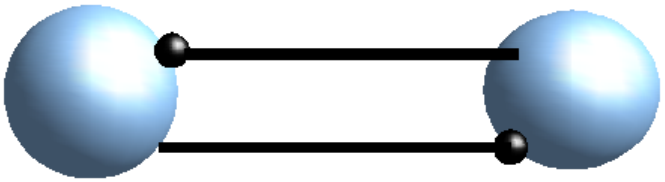

Fig. 9 The basic element of the central pattern generator is the Brown half-centered oscillators. Two neurons are coupled with mutual inhibition. As one neuron fires, it suppresses the other. Eventually, the firing rate wanes and the other neurons becomes active and the cycle repeats.

The basic neural unit is a network oscillator composed of two neuron in mutual inhibition, see Fig. 9. The basic idea of the half-centered oscillator was first proposed by Brown [11] and was the key element in a theory of central pattern generation, that is, the generation of rhythmic movement by spinal circuits without the need for a sensory trigger, or a detailed signal from the brain. While this idea was successfully repressed for period of time, in recent years it has become accepted

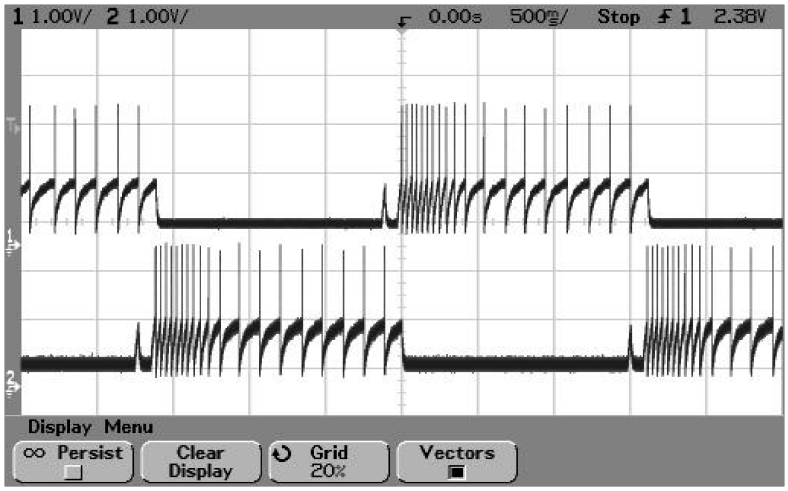

Fig. 10 A Brown half centered oscillator constructed using an integrate and fire neurons with adaptation implemented with discrete components (i.e. real resistors, capacitor etc., according to the equations given above).

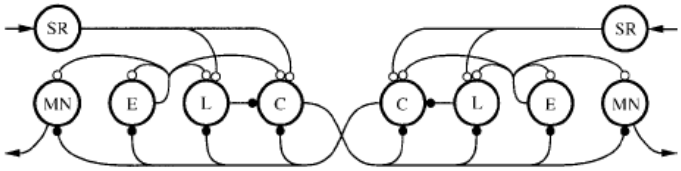

Fig. 11 Basic circuitry controlling locomotion in lamprey, an eel like animal. Adapted from [16]. The C neurons are coupled across the midline with mutual inhibition. They form the basis of the left-right bending of the lamprey.

since Grillner's seminal work [18, 15]. In Fig. 11 we see an illustration of the spinal circuits for locomotion in lamprey. The lamprey is a animal that first evolved some 550 million years ago and has been relatively undifferentiated since that time. It thus gives us insight into the earliest solutions of biology. Such circuits, especially with inhibition across the mid-line to enforce a 180 degree phase shift between the left and right half body has been highly conserved. Whenever you walk, you take advantage of the lamprey solution in the left and right alternation of your legs.

3.6 Reflexes and High Level Control

The CPG circuits are modulated by sensory stimuli to adapt the CPG to the animal and to its immediate environment. Thus, the CPG is not a rigorous general given orders to a low level control system. Rather it cooperates dynamically with the environment via reflexes, for short term adaptation, and descending control from the brain, for long range anticipatory control and to achieve high level goals.

For details on the biophysics of neurons, the reader is referred to [19]. For more details on neural models the reader is referred to [14]

4 Notable systems

Here we highlight some notable models in neurobotics/biorobotics by others

1. Case Western group– Beer and colleages have created neural models of locomotion. Starting from simulation, this group has built a robot controlled by CPG based on the cockroach. This work is highly instructive. See [12, 7, 17, 6, 5] for details on their work.
2. Neuromechanical model of swimming— Ekeberg has created a series of neuromechanical models of swimming in lamprey. What is interesting here is the integration of neurons, muscles, body and environment to achieve a complex behavior, swimming [16].
3. TAGA— Through simulation, Taga and colleagues have explored the relationship between dynamics, neural networks, and the environment. [35, 33, 32, 34]
4. Tekken— Tekken is a quadrupedal robot driven by Matsuoka oscillators. It is capable of rapid walking, and features highly innovative reflex integration with CPGs, see [25, 24].

5 GPUs

Central Processing Units have doubled in speed about every 18 months for the past 40 years [1]. Graphics Processing Units have doubled in speed about every 12 months over the past 8 years [2]. The result has been that graphics units are arguably the most important computational element in the modern computer.

NIVIDIA has opened their architecture to allow other to use their highly parallel architecture for general purpose simulation. GPUs are distinguished by devoting a great deal of chip real-estate to computation, instead of memory. They have a relatively small amount of high speed memory and also a large pool of They are therefore best suited for compute-bound processes. As an example, the Hodgkin-Huxley model is estimated to require 1200 flops per 1 ms of computation [23] while using only a handful of state variables. Further, neuronal models are inherently paralleliz-

able as they rely on very similar computation applied to different state variables. On the other, hand there are some elements of the neuron elements that high dimensional and require a relatively small amount of computation for 1 ms of simulation. An example of that is the simulation of axonal delay, which can be implemented as an individual delay for each connection between an axon and a dendrites. Given the small amount of high speed memory, it is not practical to port a large and complex delay line onto GPU processors. The inhomogeneity of computational versus state space dimension for different aspects of neural simulation seems to prescribe a certain mapping of the neuronal network simulation onto a combination of GPU and CPU elements.

A neural simulation for neurons, the network connectivity, input and output provisions, and learning rules. The neuron simulation is composed of a (1) Simulation of the decision to fire, (2) Simulation of axonal delays (3) simulation of synapses (4) dendrite simulation. The learning rule is typically confined to the synapse weight. However, there is in principle no reason why axonal delays might not also be changed or altered as they might play an important role in storage of activity patterns.

The raw processing power of a GPU is staggering. The NVIDIA Telsa card with 128 core of processors is capable of nearly 500 GigaFlops of computation. A new version, to be released in late 2008 will have double that power.

Recent experiments in our lab have achieved rates of 10 million Izhikevich neurons in real-time. Currently this preliminary figure does not include long range synapses. The results are promising. Within a year, we anticipate that a computer with 4 NVIDIA GPU cards will be capable of integrating about 80 million neurons in real time, or nearly 10^8 neurons in a single desktop computer. If GPU maintain their rate of speed increase, doubling each eary, (which is not guaranteed), within a little more than a decade it may be possible to simulate as many neurons as their are in a human brain in real-time and to control a robot.

Problems still exist for the GPU paradigm including the problem of getting information to the computational cores in an efficient manner. Yet even more insurmountable problem exist in taking advantage of the processing power we have today.

It is clear that one of the key innovations in neurorobotics will lie in harnessing GPU processing power for real-time neural computation

6 Conclusion

Neurorobotics is a paradigm which has evolved out of the desire to understand computation in the human brain. Due to advances in the theory of neural computation, and the dramatic increase in processing power, Neurorobotics may prove to be capable of creating highly complex behaviors in life-like machines. The market acceptance of such machines is complicated and remains uncertain, but promising.

Acknowledgements The authors wish to thank Kiwon Sohn for his careful reading of this manuscript and useful comments. The authors acknowledge the support of the University of Arizona startup fund to MAL. This chapter is derived from a lecture given by MAL at the Telluride Neuromorphic Engineering Workshop in 2008.

References

1. Moore's law. http://en.wikipedia.com/wiki/Moore's_law
2. Nvida cuda compute unified device architecture programming manual. http://nvidia.com/cuda
3. Arkin, R.: Behavior-Based Robotics. The MIT Press, Cambridge (1998)
4. Asada, H., Slotine, J.: Robot Analysis and Control. John Wiley and Sons, New York (1986)
5. Beer, R., Chiel, H.J., Quinn, R., Ritzmann, R.: Biorobotic approaches to the study of motor systems. Current Opinion in Neurobiology **8**, 777–782 (1998)
6. Beer, R., R.D., Q., Chiel, H.J., Ritzmann, R.: Biologically-inspired approaches to robotics. Communication of the ACM **40**(3) (1997)
7. Beer, R.D., Chiel, H.J., Gallagher, J.: Evolution and analysis of model cpgs for walking ii. general principles and individual variability. Journal of Computational Neuroscience **7**, 119–147 (1999)
8. Bekey, G.: Autonomous Robots: From Biological Inspiration to Implementation and Control. Blackwell Scientific Publications, The MIT PRESS (2005)
9. Brooks, R.: Achieving artificial intelligence through building robots. Tech. rep., Massachusetts Institute of Technology (1986)
10. Brooks, R., Stein, A.: Building brains for bodies. Autonomous Robots **1**, 7–25 (1994)
11. Brown, T.G.: The intrinsic factors in the act of progression in the mammal. Proc. of the Royal Soc. Lond., Ser. B. **84**, 309–319 (1911)
12. Calvitti, A., Beer, R.D.: Analysis of a distributed model of a leg coordination i. individual coordination mechanisms. Biological Cybernetics (1999)
13. Craig, J.J.: Introduction to Robotics: Mechanics and Control. Printice Hall (2004)
14. Dayan, P., Abbott, L.F.: Theoretical Neuroscience: Computational and mathematical Modeling of Neural Systems. The MIT Press, Cambridge, MA (2001)
15. Delcomyn, F.: Neural basis of rhythmic behavior in animals. Science **210**, 492–498 (1980)
16. Ekeberg O Grillner, S.: Simulations of neuromuscular control in lamprey swimming. Philos Trans R Soc Lond B Biol Sci. **354**(1385) (1999)
17. Gallagher, J., Beer, R.: Evolution and analysis of dynamic neural networks for agents ingtegrating vision, locomotion, and short-term memory. In: Genetic and Evolutionary Computation Conference (1999)
18. Grillner, S., Zangger, P.: On the central generation of locomotion in the low spinal cat. Exp. Brain Res. **34**, 241–61 (1979)
19. Hille, B.: Ionic Channels of Excitable Membranes. Sinauer, Sunderland (1984)
20. Hodgkin, A.L., Huxley, A.F.: A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. **117**(4), 500–544 (1952)
21. Hornik, K., White, M.S.H.: Multilayer feedforward networks are universal approximators. Neural Networks **2**, 359–366 (1989)
22. Izhikevich, E.: Simple model of spiking neurons. IEEE Transactions on Neural Networks **14**(6), 1569 – 1572 (2003)
23. Izhikevich, E.: Which model to use for cortical spiking neurons? IEEE Transactions on Neural Networks **15**(5), 1063 – 1070 (2004)
24. Kimura, H., Fukuoka, Y., Hada, Y., Takase, K.: Adaptive dynamic walking of a quadruped robot on irregular terrain using a neural system model. In: ISRR, pp. 88–97. Lorne, Australia (2001)

25. Kimura, H., Fukuoka, Y., Konaga, K., Hada, Y., Takase, K.: Towards 3d adaptive dynamic walking of a quadruped robot on irregular terrain by using neural system model. In: IEEE and RSJ IROS 2001. IEEE, Hawaii (2001)
26. Lewis, M.A.: Perception driven robot locomotion. Robot Society of Japan Journal. (2002)
27. Lewis, M.A., Etienne-Cummings, R., Hartmann, M., Cohen, A.: Toward biomorphic control using custom avlsi chips. In: 2000 International Conference on Robotics and Automation. IEEE, San Francisco (2000)
28. Lewis, M.A., Etienne-Cummings, R., Hartmann, M., Xu, Z.R., Cohen, A.H.: An in silico central pattern generator: Oscillator, entrainment, motor neuron adaptation & biped mechanism control. Biological Cybernetics **88**(2), 137–151 (2003)
29. Lewis, M.A., Nelson, M.E.: Look before you leap: Peering behavior for depth perception. In: Simulation of Adaptive Behavior. Zurich (1998)
30. Matsuoka, K.: Mechanisms of frequency and pattern control in the neural rhythm generators. Biol. Cybern **56**, 345–353 (1987)
31. Paul, R.P.: Robot Manipulators: Mathematics, Programming, and Control. The MIT Press, Cambridge (1982)
32. Taga, G.: A model of the neuro-musculo-skeletal system for human locomotion. i. emergence of basic gait. Biological Cybernetics **73**, 97–111 (1995)
33. Taga, G.: A model of the neuro-musculo-skeletal system for human locomotion. ii. real-time adaptability under various constraints. Biological Cybernetics **73**, 113–121 (1995)
34. Taga, G.: A model of the neuro-musculo-skeletal system for anticipatory adjustment of human locomotion during obstacle avoidance. Biological Cybernetics **78**, 9–17 (1998)
35. Taga, G., Yamaguchi, Y., Shimizu, H.: Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. Biol. Cybern. **65**, 147–159 (1991)
36. Webb, B.: Can robots make good models of biological behavior? Behav. Brain Sci pp. 1033–50 (2001)
37. Yanofsky, P.: President wowwee robotics. personal communication (2005)