0

Real-Time Control in Robotic Systems

Alex Simpkins
University of Washington
USA

1. Introduction

Robotic systems are beginning to interact with humans and are increasing in dimensionality, control complexity, task difficulty, and dynamic capability. It is therefore becoming increasingly imperative that control be performed in real-time (i.e. on or faster than the timescale of the system dynamics), intelligently making decisions on demand, and then executing them in a dependable fashion. In addition, high-dimensional bio-inspired systems are becoming more common (Butterfab & et al., 2001; Movellan et al., 2005; Simpkins et al., 2011; Todorov et al., 2010; Vandeweghe et al., 2004; Wilkinson et al., 2003), and are based upon a different strategy for control (and therefore control system design) than traditional engineered systems - coordinated, compliant, and often coupled control rather than orthogonal, stiff, and independent control of joints. As a result, issues related to real-time capability, more critical now than ever, must be addressed with a coherent, measured approach.

In order to deal with this challenge appropriately, a number of sub-problems must be addressed and will be discussed in this chapter. These are timing, dimensionality, computational capabilities, system bandwidth, and safety. There are fields devoted to each problem independently, and the background of each will be given as each one is discussed. Rarely does a researcher or practitioner in one field work in any of the other fields, let alone all of them. However, ignoring any of these sub-problems will likely lead to a failure of a complex real-time system. Therefore, familiarization with these issues and methods will help the researcher and practitioner design, implement, and troubleshoot a real-time system most effectively. This chapter will present current solution methods for each component of the problem and show how they can be integrated together effectively.

1.1 Key points and contributions

- A rare integrative approach between real-time control theory and practical hardware issues which are closely linked with the functionality of these theories.
- A novel means of semi-redundant integrated systems to provide improved safety measures for human-robot interaction.
- Strategies which effectively deal with high dimensional systems and complex control strategies in real-time.
- The concept of real-time is different in different fields. Here we discuss a variety, how they relate to each other, and how each can enhance and integrate with the others.

- This unifies and expands upon approaches in the fields of real-time control systems, control engineering, mechatronics, dynamics, robotics, embedded systems, and computer science to address new design challenges.

1.2 Chapter organization

The rest of this chapter is organized as follows. Section 2 breaks down the real-time problem into sub-problems, and Section 3 presents how each problem is addressed. Each sub-problem is discussed in a way which exposes how they are all related, why it is important to address all aspects of the real-time problem, and how this can be done. Section 5 presents a set of results for a real-time system which has been designed with the methods presented in this chapter, demonstrating its effectiveness. Section 6 closes the chapter with a review, discussion of the problem and approach, and suggests further directions in research.

2. The sub-problems

By defining the sub-problems we can begin to address the overall issue of real-time control in robotic systems. The timing problem is centered around what is defined as real-time for a particular application. For example, an energy-producing chemical process may occur on the order of several hours, or the coordination of joints required for locomotion may occur on the order of milliseconds. Microprocessors are digital devices, and commands to the actuators are updated periodically. Thus timing issues become guaranteeing that sample time is short enough, that delays in the system are minimal (or properly addressed), that samples are not missed (or if sample time is variable, this is accounted for), and that information bottlenecks are avoided or part of the control strategy. Timing leads to issues of computational capability required versus what is available. This defines what dimensionality and control complexity is possible, and limits the overall system bandwidth. Traditionally designers have considered processing power versus requirements as an afterthought, with little concern over the criticality. This is due to the fact that, for simple problems, most modern processors are more than powerful enough. In complex systems, however, this quickly becomes a poor assumption. When one considers the actual realities of controlling a fifty degree of freedom (DOF) system, for example, where each DOF possesses multiple sensors and actuators, the required information flow bandwidth becomes large quickly. A system with such limitations can effectively lose control of some or all DOFs under dynamic load, though it may be guaranteed to be controllable in theory from a dynamical systems perspective. When considering these issues, and that in the future robots will be interacting directly with humans more often in everyday activities, safety becomes a major consideration. A robot which can exert large forces, is not back-drivable, and/or possesses significant mass is inherently unsafe for human interaction. Making the robot small and weak is not the only or best solution as this still does not guarantee that the system will not be dangerous. There are ways to minimize risk of human injury in the case of loss of control, even during a catastrophic failure of the entire primary control system.

3. The solution approach

Though there are a number of sub-goals for approaching real-time control in robotic systems, an overall and most significant approach is that the various design aspects of the system must be constrained such that each solution does not violate another's constraints. For example,

when setting sample rate for the control algorithm, one possible constraint could be that this must not cause a need to communicate data more rapidly than the data bandwidth allows.

3.1 Timing issues

Timing problems can be mitigated by ensuring that traditional real-time control issues as well as new issues arising in complex robotics are addressed. If sample time is unavoidably significantly variable, some mathematical assumptions in discrete control do not hold (Nilsson, 1998; Wittenmark et al., 1995), and various equations in the dynamics will have variation over ones with the precise sample time assumption. However, a constant delay can be modeled, following (Wittenmark et al., 1995), as a standard state-space system,

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t-\tau) \tag{1}$$

$$y(t) = Cx(t) \tag{2}$$

then the sampled-data system becomes

$$x(kh+h) = \Phi x(kh) + \Gamma_0 u(kh) + \Gamma_1 u(kh-h) \tag{3}$$

$$y(kh) = Cx(kh) \tag{4}$$

where

$$\Phi(h) = e^{Ah} \tag{5}$$

$$\Gamma_0(h,\tau) = \int_0^{h-\tau} e^{As} ds B \tag{6}$$

$$\Gamma_1(h,\tau) = e^{A(h-\tau)} \int_0^{\tau} e^{As} ds B, \tag{7}$$

and delays larger than a sample h can be dealt with fairly simply, by adding the relation

$$\tau = (d-1)h + \tau', \;\; 0 < \tau' \leq h \tag{8}$$

with an integer d. Now our system becomes, after replacing τ by τ' in Γ_0 and Γ_1,

$$x(kh+h) = \Phi x(kh) + \Gamma_0 u(kh-dh+h) + \Gamma_1 u(kh-dh). \tag{9}$$

Variable delays can be dealt with by simply computing time varying system matrices. Then the model described in Equation 3 is still valid. As long as the variation in sampling is less than a delay in duration, the model described is still valid, as the control signal does not vary with the delay. If the delay varies longer than a sample period, then a different model may be derived which causes the control signal to not be held constant, but rather to vary along with the delay. This allows the control signal to be scaled appropriately, and mitigates instabilities. Excessively large variations in sampling period can lead to unavoidable (or rather 'very difficult to avoid') instability due to violation of controllability guarantees. In other words, when the system finally processes the delayed sample, too much time may have passed to react properly, and a large error may already have been incurred from the previous control action. In the case that this type of problem is likely, it is possible to create robust controls, which guarantee certain bounded behaviors even in the event of large errors due to sample time variability, which may be lumped into disturbances. Another way to effectively alter

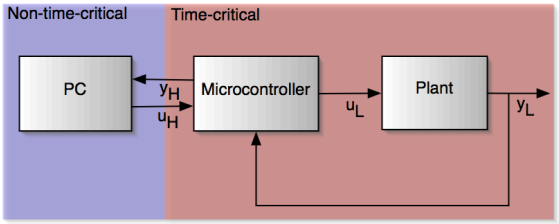


Fig. 1. For complex control systems, it is effective to split time-critical aspects of control from non-critical aspects. In this way a complex algorithm can be computed in near real-time, but may not be guaranteed, while the critical aspects which require such guarantees can be designed at the low level to handle variable updates to its reference. L stands for 'Low level' while H stands for 'High level.' y represents output, and u represents control input.

control parameters to match the delay is to include the delay as a parameter of the control equation (Åström & Wittenmark, 1990).
Thus we can attempt to model delays by measurement, and then compute an appropriate control, but that does not tell us everything that might happen, and how do we, in the context of complex systems, measure for every possible permutation? Additionally, it may not be trivial to make measurements of some features needing to be measured! That leads us to the next section regarding computational capabilities.

3.2 Computational capability issues

Computational capabilities required depend on required overall update rates, complexity of the control algorithm, number of peripherals involved and dimensionality of the system (thus the size of data passed around during code execution). There are a number of ways to ensure code will be processed on time and that computational requirements and capability are matched. With highly complex control algorithms required for artificial intelligence and complex coordinated movements, a very effective approach to ensuring real-time capability is to split the time-critical aspects of control from the computationally intensive aspects, as in Figure 1. Then a hierarchical but highly interconnected framework is formulated. This parallels the human brain's sensorimotor system structure, which can be thought of, most definitely, as the most intricate and advanced control system on the planet. Biological systems are also excellent at solving the types of control problems now beginning to be addressed by roboticists. The high level would consist of powerful personal computers or clusters of computers which are excellent for massive floating point computations but are limited for time-critical applications, while the low level would consist of embedded systems which are excellent at time-critical applications but less adept at complex calculations. This design approach has been applied and is detailed in (Simpkins et al., 2011). The challenge then becomes communication between the levels. There are a number of communication strategies and protocols which address this issue well. Additionally, the high level operating system must be carefully selected. Most common operating systems are not real-time.
GPU parallel processing is a powerful tool which is significantly under-applied in control applications at this time. It is possible to perform complex algorithms rapidly on thousands of processors in parallel on such devices, and the current state-of-the-art simulators are beginning to take advantage of such capabilities (such as introduced in the appendix of (Erez et al., 2011), which takes advantage of such an engine for speed). They are low cost (especially

compared to cluster computing) and highly effective. This approach is useful not only to run single-thread algorithms, but also to run multiple model predictive algorithms in parallel. In this way, several potential control decisions or trajectories can be explored very rapidly, and the one with the best outcome selected.
Measuring execution time for all the computations required, including communication delays, and delays from peripherals is an important ability to have. This is not a trivial problem, however.

3.2.1 Worst case execution time

Worst Case Execution Time (WCET) is the time it takes to execute the longest branch of code, not including hardware peripherals. This is important to at least have a reliable estimate of, since a designer needs to know whether a task can finish in a set amount of time for real-time systems. Traditionally, with fixed code and no caches, this was not as difficult a problem as it is in the context of more advanced (often multicore) processors, infinite loops, complex pipelines, and dynamic code length.
The traditional methods of measuring WCET comprise an entire field and will be mentioned here fairly briefly, as the main contribution is when and how to use these tools in conjunction with all the other subproblems that need to be addressed (The interested reader is referred to (Ermedahl, 2003; Hansen et al., 2009; Wilhelm et al., 2008). This field is moderately mature, though advances continue steadily. Timing analysis is the process of determining estimates or bounds on execution times. (Wilhelm et al., 2008) provides an excellent overview of timing analysis and the WCET problem in general, as well as tools which are available.
There essentially are two approaches for WCET prediction. These are static and dynamic (Grob, 2001) approaches. Static prediction approaches measure small pieces of code which do not change, do not have recursion, and are otherwise limited (Puschner & Nossal, 1998), while dynamic approaches (sometimes called measurement-based methods) measure an entire execution pathway, running on a simulator or the actual hardware for a set of (probably limited) inputs.
The static method may not provide realistic estimates when code snippets are combined together, and scheduling complex algorithms which combine many snippets together may break down the estimates. In addition, some processors have speculative components, where the processor determines an expected path, begins to load items in memory, and performs some processing ahead of time in anticipation. However, these speculations can of course be mistaken, and then all the predicted data must be deleted and the proper execution performed, which is actually slower than just waiting then branching when the call happens, without prediction.
Modern processors provide means of measuring code performance in a variety of ways via peripherals built into the chip, but some of these methods are imprecise at best. They do give a measure of how many cycles a task takes, however performing the measurement itself takes some computation. This violates the basic principle of measurement - that the action of performing the measurement should have such a small effect on the system measured that no significant change occurs in the process, but it does give a great deal of information, and the effect is fairly small. These measures can be used to create an estimate of best and worst case execution times. In practice this is very helpful, though difficult or impossible to provide hard guarantees in all cases.

Two effective approaches which can be used on complex real-time robotic control systems, of the techniques discussed are either to split the time-critical components from the non-critical components, or to use a measurement technique which will not necessarily guarantee perfect timing, but may provide fairly good estimates on bounds of timing. In safety-critical applications it is suggested that having a low level which has timing guarantees is more appropriate than bounded estimates, in combination with a watchdog (discussed in Section 3.5). In some experimental robotic systems it may be sufficient to simply design controllers which deal with temporal variability without instability, but write code which minimizes delays. This latter method may be effective for rapid prototyping of new designs and testing experimental theories, while the hierarchical structure with some components real-time may be a more effective general strategy, as long as communication between levels is designed carefully, such as in (Simpkins et al., 2011). Standard timing analysis tools should be used at the programming stage to test algorithms and ensure their real-time performance. Even in the best cases, there are always limitations, and in complex high dimensional systems we must consider carefully what is the maximum capability available, and how much is required.

3.3 Dimensionality and bandwidth

The dimensionality of a system may technically be very high, but can be reduced in practice depending on the task. Or alternatively, simple coordination can be performed at a low level (akin to reflexes in biological systems, which are thought to be processed in the brain stem or lower levels, depending), leaving higher levels free to coordinate a few degrees of freedom (DOF) in more complex patterns. This reduces and efficiently distributes the processing load. For example, though human beings have over twenty-two DOFs in their hands, during the majority of grasping and manipulation tasks, they tend to use as few as two DOFs for a variety of common tasks. The individual DOFs are coupled together into movement 'synergies,' which makes the computational task significantly simpler.

3.3.1 Advantages of synergies and evidence to support their uses

Biological systems are almost unilaterally agreed to be capable of superior complex manipulation tasks in terms of complexity and variety of skills. Interestingly enough, several studies have been performed of humans undergoing complex manipulation tasks, and dimensionality of the movements have been explored using principle components analysis and canonical correlations analysis. See (Anderson, 2003) and (Mardia et al., 1992) for good discussions of these techniques. The results essentially state that biological systems use synergistic movements to achieve goals. It is likely that there are a number of reasons for this, but one advantage of synergistic control is that the number of individual dimensions controlled is reduced significantly over the total state space. Of the many DOFs available, many manipulation tasks require less than *three* (Santello et al., 1998). This has led some researchers to claim that this is a maximum for control but it appears that there is a sliding dimensionality. However dynamic complex movements require more training than lower dimensional, less dynamic ones. One theory is that when learning a new skill, humans go through stages of learning, where initially many dimensions are frozen, then gradually are unfrozen as learning takes place(Bernstein, 1967). One possible explanation for this strategy is that, in order to free bandwidth as much as possible, coordinated movement is encoded at as low a level as possible, freeing up the high level to devote bandwidth to complexity of the strategy, rather than simply giving individual joint commands. In fact, the manifold of

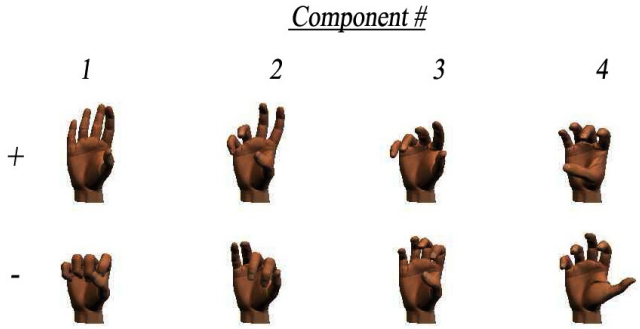


Fig. 2. First four principle components of postural movement data for humans undergoing a 'variability maximizing experiment,' scaled appropriately for joint angles and applied such that the size of the component for a joint determines its deviation from a neutral posture. It is interesting to note that the components, which represent the dimensions that were most significant in terms of accounting for the variability in the data, and the direction of variability, comprise certain basic components of manipulation - gripping with all fingers, using a wavelike pattern of all fingers or using the first few primary fingers and thumb, precision manipulation with all fingers in fine patterns, and finally the thumb dominating the movement. This may suggest that combining these synergies, or using subsets of fingers, many manipulations can be performed.

the subspace is quite dynamic, and changes depending on the task. In other words, joints are coupled together in a fairly optimal fashion for each task(Todorov & Ghahramani, 2003; 2004). In fact, the author has performed a series of studies attempting to maximize the dimensionality of hand movement control, and there is still a notion of synergies(Simpkins, 2009). The maximum dimensionality is approximately ten DOFs. The obvious advantage is that the less individual trajectories that must be computed, or more generally, control decisions to be individually made, the faster those decisions can happen, and the more complexity can go into computing what exactly those decisions will be. So couple joints together when possible into synergies such as Figure 2 and compute a global trajectory for them to follow, such as open vs. closed hand (See Figure 3), and this can be done quite quickly and efficiently computationally. A number of approaches can be taken to build control systems upon this general methodology, but optimal control has been found to effectively model much biological movement control data, and provides an intuitive means of building control systems. Optimal control (Stengel, 1986) is a methodology which makes decisions based upon a notion of optimality for a task. This notion comes in the form of a cost (or reward, hereafter referred to only as cost for simplicity) function. Behaviors are created by specifying the dynamics of the system as constraints, then actions are chosen which minimize the cost computed by the cost function. There is evidence that the sensorimotor system ignores task-irrelevant deviations (Todorov, 2004; Todorov & Jordan, 2003), which tends to reduce dynamic load and uses less system bandwidth to compute controls than actively controlling everything individually.

Bandwidth is a term which is used in many fields for different purposes. In control theory it relates the ability to track a reference signal (Franklin & et al., 1994); in communications and

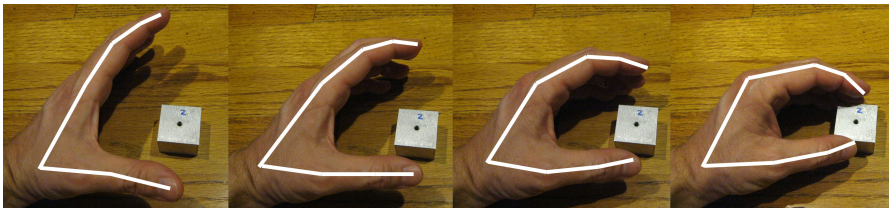


Fig. 3. Images of a hand gripping an object. Though there are many degrees of freedom in the hand, it is clear that there is a low dimensional synergy in this case. We can control the entire hand from this synergy for this task (open-close). In that way, more complex algorithms can be computed in real-time if needed. A sliding complexity (in terms of dimensionality and dynamics) can be used to guarantee real-time control even for a complex system.

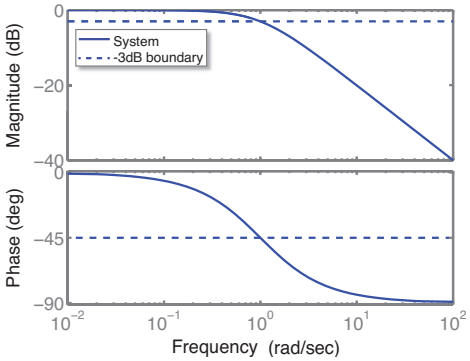


Fig. 4. The bandwidth cutoff for a system tracking a sinusoidal input is considered to be the point at which the amplitude of the output drops below -3dB, along with a corresponding phase shift. This is the bode plot for $G(s) = 1/(s+1)$.

networking it refers to how much data throughput is possible. However, the general concept is very similar in all instances. We will use as the following as the definition of bandwidth:
Definition: ***Bandwidth*** *shall be defined as the rate at which a system, open-loop or closed-loop, may successfully track a reference. The boundary between successful tracking and unsuccessful tracking is the point at which the amplitude of the output drops below -3 Decibels (dB) relative to the input signal in the frequency domain, such as Figure 4.*
The bandwidth requirement for a system is also reduced by synergistic movement control. It is also significant that a designer consider not what the maximum possible bandwidth of the overall system can be, but more what the maximum bandwidth *should* be. In other words, what is required (along with a factor of safety) to succeed at all tasks the system is designed to accomplish? Too high of a (dynamic) mechanical bandwidth may lead to instability, high forces, and other problems affecting system safety. Too high of a data bandwidth leads to higher probabilities of lost samples, delays, and related issues. It is also possible to have a sliding bandwidth which varies depending on computational complexity versus dynamic timing of a particular task. In fact, this appears to be closer to what biological systems do -

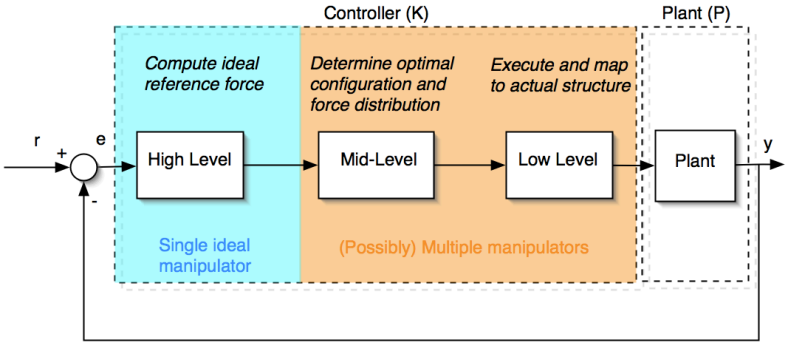


Fig. 5. Block diagram of hierarchical control scheme. This approach breaks a single complex problem into several smaller and more tractable problems.

use some subset of features of the complex system for any given task, but have the capability for a variety of tasks (Todorov & Jordan, 2002).

3.4 Control complexity

As more intelligent control is applied, the computations required to carry out these algorithms tend to increase. For example, the number of mathematical operations per second to run model-predictive nonlinear stochastic optimal control is far higher than proportional integral derivative control.

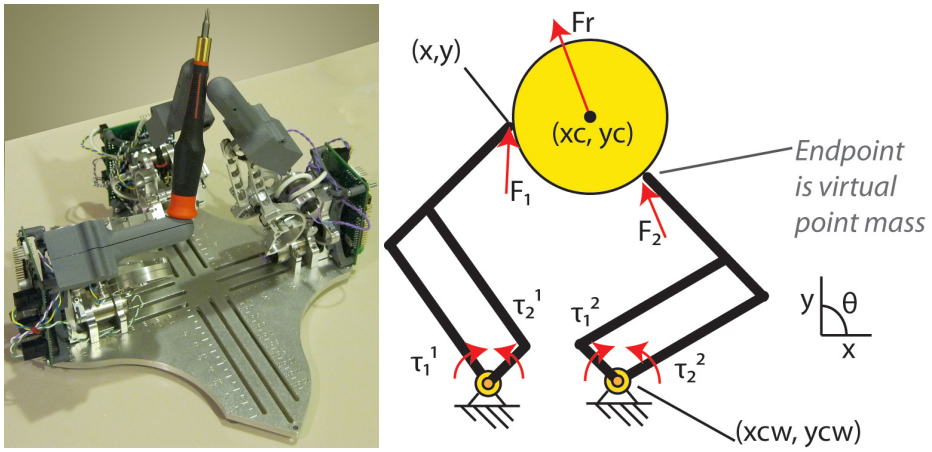
An approach which is promising is to break the one complex problem down into simpler components with a hierarchical control approach (See Figure 5). Then different components of the hierarchy, as discussed in Section 3.2, can address either time-critical or computationally intensive aspects of the controller. This will be explored through results from a system developed by the author (in collaboration with Michael S. Kelley and Emanuel Todorov) for complex manipulation and locomotion (Figure 6(a)).

Finally, some aspects of the control can be computed or adapted offline or on a slow timescale. This is parallel to some learning processes in humans. The resulting solutions (such as pre-computed trajectories) can be accessed from memory rapidly when required.

3.4.1 Model and approach

The basis for this approach is presented in (Simpkins & Todorov, 2011) for the robot in (Simpkins et al., 2011).

The goal of these robotic fingers was to develop a robotic system which parallels certain key aspects of biological systems for manipulation (and similarly for locomotion). The reason to do so is that most robots are not designed to behave in a similar fashion to biological systems. They are often solving different problems, such as how to follow a trajectory in order to weld two parts together or to cut away material in a prescribed pattern with a milling bit. In these standard cases it is very important for the robot to track a specific trajectory regardless of outside disturbances from the world. Biological systems such as a human being walking across a room, or selecting a key from a keychain full of keys in his or her pocket are solving a completely different problem. They require a bidirectional information flow with the world at an actuator level - many movements are more like a dance with the outside world as one



(a) (b)

Fig. 6. (a) A modular bio-mimetic robotic system developed by the author at the University of California, San Diego and the University of Washington in the Movement Control Laboratory. (b) A simplified 2-dimensional representation of the robot and object, used in the algorithm described in this section.

partner, and the human being as the other. The human acts on the keys by applying forces to the keychain, the keychain changes state, and by nature of the contacts with the human's hand, alters the state of the human, who gains information about the state of the keys. This also helps simplify the control problem in various ways, as the human hand conforms to the shape of the key, rather than the hand attempting to move *in spite of* the key.

Creating a robot which interacts with the world in this fashion involves solving a number of new design challenges without compromising on any one specification. For example, in order to dynamically manipulate objects, a robotic finger must be able to move rapidly, be completely compliant and back-drivable, communicate at a high rate with a host processor, be capable of implementing complex control in real-time, be sensitive, measure appropriate variables (such as force, velocity, and position), and it must be sufficiently robust. This is to the author's knowledge the first design which solves all these constraints and incorporates modularity (each finger is completely self-contained and has wireless capabilities, with provisions to be attached to any base with the appropriate hole pattern). Controlling such a device to perform dynamic manipulation in the way previously described is a new type of control problem, and requires a significantly new approach.

The goal is to apply forces to an object to cause it to behave in a desired fashion. This can be tracking a trajectory, or any other goal of control. An important note is that the two problems (manipulation and locomotion) are related in that we either have a small object relative to manipulators such as a hand, or the ground, an object of infinite size (in either case, it is often that one side of the problem is significantly larger than the other side - manipulators or object manipulated), as in Figure 7.

Forces are applied to the object by manipulators, described as linkages, as shown in Figure 6(b), which are a simplified model of the physical robots developed by the author, shown in Figure 6(a) and described in detail in (Simpkins et al., 2011). Torques (τ_i) are applied to either

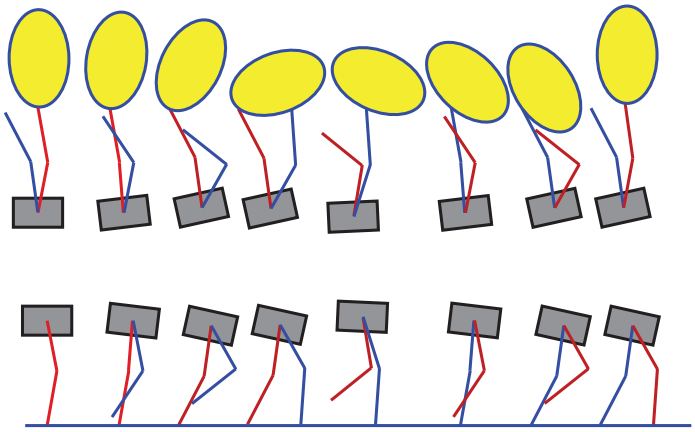


Fig. 7. Image sequences showing manipulation of an object (top) and locomotion (bottom). Note the parallels between the two. In fact, the two can be thought of as essentially the same problem - this is a significant insight.

of the two links shown for a particular manipulator in order to generate an output force at the endpoint. This is similar to human joints in terms of the way tendons pull on joints to cause an output force. However, here for design simplicity and robustness, the complex inner musculoskeletal structure is reduced to a four bar linkage model. The motors which produce the torque are coupled to the joints through a cable drive system which has ultra-low friction and little if any backlash - essential components for back-drivability. The cables act as the tendons would in biological systems in the sense of a pull-pull system, as well as providing some spring characteristics. As much mass as possible is kept at the base, and decoupled from the output, which is another reason for the linkage design. This helps keep inertia small relative to the objects being manipulated, and relative to the output force capabilities of the finger. This is significant because the finger needs to be capable of dynamic movement (rapid accelerations and decelerations), and must avoid wasting most of the force output capability of the motors in overcoming the structure's inertia (imagine going through your day with an extra 10kg of weight strapped to your wrists; it would indeed be very difficult to perform simple tasks because nearly all of your strength would be used to overcome the weight and inertia of the additional mass). Large inertia also is very detrimental to back-drivability.

The concept of these devices is not to attempt to duplicate the structure of a biological system. Rather, the concept is to capture the capabilities which are significant for developing controllers that solve the problems of manipulation and locomotion similarly to biological systems. Each finger is capable of producing a few Newtons of peak output force in each axis, and requires a fraction of that to be backdriven (frictional and inertial loads are small), enough for fine dynamic manipulation. Each can perform closed loop control at over 1kHz, communicating between all fingers and a higher level processor (such as a personal computer) all relevant data (position, velocity, force, actuator states, etc).

Consider for simplicity the manipulation or locomotion problem within a 2D context. Objects are represented by center of mass, position and angle, mass (m), inertia (J), and a boundary ($d()$), which may be described by splines, allowing any shape to be represented, convex or concave. The sum of forces on the object (F_x, F_y, and M) are given by

$$\begin{Bmatrix} \sum F_{x,o} \\ \sum F_{y,o} \\ \sum M_o \end{Bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -d_y(\theta) & d_x(\theta) \end{bmatrix} \begin{Bmatrix} \sum_i f_{x_i} \\ \sum_i f_{y_i} \end{Bmatrix} - \begin{bmatrix} m_o & 0 & 0 \\ 0 & m_o & 0 \\ 0 & 0 & J \end{bmatrix} \begin{Bmatrix} a_{x,o} \\ (a_{y,o} + g) \\ \ddot{\theta}_o \end{Bmatrix} \tag{10}$$

Where f_i representing the force applied by manipulator i at the surface location relative to the object center of mass as a function of angle (θ) relative to horizontal in the global coordinate system ($d(\theta)$), g representing gravitational acceleration, a representing acceleration of either manipulator i's endpoint or the manipulated object, depending on the subscript, and $()_o$ representing 'with respect to the object being manipulated.' Time derivatives are represented by an over-dot, for example, the time derivative of x has the notation $\dot{x}$.

Contact dynamics can be complex to simulate accurately, and can bring about bottlenecks for otherwise lightning fast algorithms (important in real-time contexts). Though with the rigid body assumption it is true that accurate contact dynamics appear difficult to 'get right,' real objects are not infinitely rigid. This provides a means by which contacts are smooth, soft, and damped. The reader is encouraged to consider his or her hands as an example- note the flexibility of the tissue. In general we, as human beings, do very little manipulation of rigid objects with rigid objects. When was the last time you attempted to pick up and manipulate a coffee cup or other fairly rigid body while wearing rigid steel armor which has no padding over your hand? It is unlikely that you ever have, or ever will. In fact, it is important to have damped interactions to avoid damaging the manipulators and objects - consider shoes, which prevent joint and limb damage due to shocks over time. This makes the problem significantly simpler, and we can make strong assumptions which do not necessarily reproduce contact dynamics of rigid bodies perfectly, but instead assume that objects interacting with each other do so with damping. In addition, the concepts we are presenting here do not necessarily depend upon the contact algorithm, being a modular method, and so the physics and contact model can easily be switched for others, even on the fly. And it makes sense to have a more flexible strategy for simulating dynamic interactions than making one assumption and basing the entire methodology upon this such that the algorithm becomes 'brittle' to simple changes, or becomes computationally infeasible due to complexity.

Given this discussion we create a constraint for the form of the algorithm presented here (Simpkins & Todorov, 2011). We assume that there is no slip during contacts, effectively creating a sticky contact algorithm. The advantage here is that it allows a constraint on acceleration of the manipulators relative to the object,

$$\begin{Bmatrix} a_{x,i} \\ a_{y,i} \end{Bmatrix} = \begin{bmatrix} 1 & 0 & d_x(\theta_i) \\ 0 & 1 & d_y(\theta_i) \end{bmatrix} \begin{Bmatrix} a_{x,o} \\ a_{y,o} \\ \ddot{\theta}_o \end{Bmatrix} \tag{11}$$

where a represents the acceleration of the object o or point mass i, and $\ddot{\theta}_o$ is the angular acceleration of the object. This does create an instantaneous change of velocity when a contact occurs, which can be modeled as an impulsive force that causes the change in velocity which is required. There is also an assumption that the point masses are small enough in mass relative to the object that their contribution to the momentum of the system when in contact is insignificant.

When not in contact, the point masses experience accelerations due to a force which is applied by the robot fingers. The entire robot finger assembly is modeled as a point mass with external

forces applied to the point mass in order to cause accelerations. This assumption is made because the rotor inertias of the motors dominate the overall finger inertia. It is also makes the system more simple to model and control than one with complex nonlinear dynamics, but this would be possible as well and is a trivial extension. The affine point mass dynamics for mass i are given by

$$\begin{Bmatrix} a_{x,i} \\ a_{y,i} \\ 1 \end{Bmatrix} = \begin{bmatrix} \frac{1}{m_i} & 0 & 0 \\ 0 & \frac{1}{m_i} & -g \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} b_{x,i} \\ b_{y,i} \\ 1 \end{Bmatrix} \tag{12}$$

and the object dynamics, after rearranging the terms in Equation 10 are given by

$$\begin{Bmatrix} \sum_i f_{x,i} \\ \sum_i f_{y,i} \\ -\sum_i f_{x,i} d_y(\theta_{i,p}) + \sum_i f_{y,i} d_x(\theta_{i,p}) \end{Bmatrix} - \begin{Bmatrix} m_o a_{x,o} \\ m_o a_{y,o} \\ J\ddot{\theta}_o \end{Bmatrix} = \begin{Bmatrix} 0 \\ m_o g \\ 0 \end{Bmatrix} \tag{13}$$

and the end effector dynamics when in contact with the object can be similarly derived to give

$$\begin{Bmatrix} f_{x,i} \\ f_{y,i} \end{Bmatrix} + m_i \begin{Bmatrix} a_{x,o} \\ a_{y,o} \end{Bmatrix} + m_i \begin{Bmatrix} \ddot{\theta}_o d_x(\theta_i) \\ \ddot{\theta}_o d_y(\theta_i) \end{Bmatrix} = \begin{Bmatrix} b_{x,i} \\ b_{y,i} + m_i g \end{Bmatrix}. \tag{14}$$

We combine the object and end effector dynamics into a single equation, which allows us to solve for the unknown forces and accelerations in one operation

$$Aw = b, \tag{15}$$

where A is given by

$$A = \begin{bmatrix} 1 & 0 & -m_o & 0 & 0 \\ 0 & 1 & 0 & -m_o & 0 \\ 0 & 0 & 0 & 0 & -J \\ I & \mathbf{0} & m_i & 0 & m_i \ddot{\theta}_o d_x(\theta_i) \\ \mathbf{0} & I & 0 & m_i & m_i \ddot{\theta}_o d_y(\theta_i) \end{bmatrix} \tag{16}$$

w is given by,

$$w = \left\{ f_{(x,i)}, f_{(y,i)}, a_{(x,o)}, a_{(y,o)}, \ddot{\theta}_o \right\}^T \tag{17}$$

and b is given by

$$b = \left\{ 0, g, 0, b_{(x,i)}, b_{(y,i)} + m_i g \right\}^T. \tag{18}$$

Then, we can compute a solution, or an approximation to the solution quite quickly using the solution method of choice. One method that works well is simply to use the pseudoinverse of A, which will yield the forces the point masses apply on the object and thus the forces on the point masses in response, and the acceleration of the object.

$$w = A^+ b \tag{19}$$

The A matrix changes dimensions as needed depending on which manipulator is in contact, on the fly. This is how we can easily compute an approximation to the contact forces and resulting acceleration.

The overall problem, now that the physics are laid out, can be presented as the following:

Compute the optimal forces and contact locations to track the reference if in contact, and if not, compute the trajectories for those manipulators not in contact in order to move them to new contact points, or into a useful position (if, for example, the object moves out of the particular manipulator's workspace).
These two trajectory classes form a pattern, which could be referred to as a gait for manipulation or locomotion.
There are several sub-problems to consider in order to not only solve the above overall problem, but to do so in a useful way - for example, manipulate objects without dropping them, make a robot walk without falling over. Additionally, it should be noted that most manipulation problems involve redundancy - usually one has more fingers than one needs to for any given task, so how does one decide what to do with each finger? There are many solutions (Simpkins & Todorov, 2011). Here we will state that this can be formulated within the optimal control framework, and solved using the method described below. In that way the solution which 'best' solves the problem at hand is selected from the set. It remains to further describe what 'best' means, and how the set of possible actions is generated.
It is possible to formulate this problem within the optimal control framework as a single level problem, however it is nonlinear, high dimensional, and stochastic. We are interested in real-time solutions, and it is difficult to solve these problems with an unprescribed length of time and computational capabilities, let alone limited resources and time. There is also support for the notion that the human brain solves such problems in a hierarchical but interconnected fashion. Thus we draw from nature to address this problem - break the one complex problem down into several more simple and tractable problems, organizing challenging aspects of the problems into groups which sum to the solution of the more difficult problem.
The three levels are the high level, which assumes one can apply an ideal force to the object directly at its center of mass, a mid-level, which takes as input the requirement for the ideal force, and decides how to break the force up into individual forces to apply with the manipulators, and where these forces should be applied, and finally a low level which deals with mapping endpoint forces to apply with manipulators to joint torques. The mid-level also, in computing optimal configurations, weighs out the difference between remaining in the same contact posture and breaking contact to create a new grip.

3.4.2 High-level

Another advantage of this approach is that the high level can employ more complex algorithms, as it will be lower dimensional overall. In fact, predicting individual trajectories of the manipulators would be prohibitively computationally expensive, however implemented in this way it is not.
To implement this high level controller, one can implement any desired strategy: optimal (Stengel, 1986), robust (Zhou & Doyle, 1998), risk-sensitive, model-predictive (Erez et al., 2011; Li & Todorov, 2004), classical (Franklin & et al., 1994), and adaptive (Krstic et al., 1995), to list a small subset of possibilities.
In an opposing notion, it is simpler to implement classical control using ideal forces, and the lower level controls may be implemented similarly as, for example, Proportional Derivative (PD) control which, in its discrete form, is given by, with ΔT the time increment, k the current time-step, K_d the differential gain and K_p the proportional gain, e the error, y the output state

feedback signal, and finally r the reference,

$$u_k^{PD} = K_p(e_k) + \frac{K_d}{\Delta T}(e_k - e_{k-1}) \tag{20}$$

where the error is given by

$$e_k = r_k - y_k \tag{21}$$

3.4.3 Mid-level

Given the required force to apply to the object, the mid-level splits this force into (given the number of manipulators) what forces each manipulator should apply and where they should apply them. This level deals also with workspace limitations, what to do if the object moves outside the manipulator workspace, and contact breaking is handled as well, all in one equation.

This is done differently, depending on which of the manipulators are in contact and which are not. Given a particular contact grouping, the overall ideal force is distributed among the in-contact fingers with a fast quadratic constrained optimization (forces constrained to only have a negative surface normal component).

The forces (f_x) acting upon the object (at location specified by $d(\theta)$) are in the x, y, and angle θ coordinates, and are linear forces and the resulting moments, given in matrix form (at timestep k) , for manipulator i acting on the object as

$$f_{t_i,k} = \begin{bmatrix} f_{x_i} & f_{y_i} d_x(\theta_i) - f_{x_i} d_y(\theta_i) \\ f_{y_i} d_x(\theta_i) - f_{x_i} d_y(\theta_i) & f_{y_i} \end{bmatrix} \tag{22}$$

The objective $J(f_i)_k$ is to minimize the error between the ideal force and the sum of total forces acting on the object, which tends to keep any force from being excessively large,

$$J(f_i)_k = \left[\left[f_{r,k} - \sum_i f_{t_i,k} \right]^2 + \sum_{(x,y),i} f_{i,k}^2 \right] \tag{23}$$

We can pose this as a constrained optimization problem by stating (with n representing the normal to the surface of the object at the point of contact),

$$\left(f_i^{opt} = \underset{f_i}{\operatorname{argmin}} \left[J(f_i)_k \right], s.t. \left[f \bullet n < 0, \right] \right) \tag{24}$$

So given the manipulators in contact, we can extremely quickly determine a set of forces which sum (as closely as possible) to an equivalent of the ideal desired force.

The behavior of the fingers that are not in contact (and the contact breaking behavior) can be encoded in a field equation which combines several different elements to create a pattern of action. This is easier to build from scratch or data than a cost function, since it is more direct and actions can be easily shaped. This is similar to directly constructing a policy - imagine a virtual force acting on a finger, so when you hold a coffee cup in your hand and rotate it continuously in one direction, there is a moment where you feel an increasing urge to change your grip (in such a way that you do not drop your cup). A virtual force field translates well to constructing these types of behaviors from observations. In the case presented here it is desirable to change grip when a manipulator is nearing the boundary of its workspace (i.e.

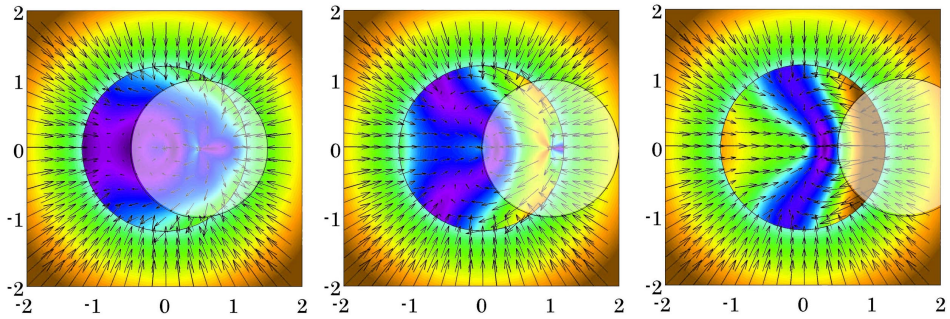


Fig. 8. Virtual force field rendering. Here one sees the forces acting on the manipulator endpoint. Note how the object (white) affects the field as it enters the workspace - the further the object is from the center of the workspace, the narrower the contact region becomes. This is a fast way of generating trajectories for the endpoint, and is very flexible for a variety of behaviors or adapting behaviors, as it is simply a matter of altering parameters of an equation.

apply a force which pulls the finger away from the object near the boundary of the workspace), and find a new contact which is closer to the middle of the workspace, in order to maximize potential for varied actions. This force should fall off near the line between the center of the workspace and the center of the object.

$$F_e = K_e \frac{(x - x_c)}{1 + ||x - x_w||^2} \tag{25}$$

Equation 25 pulls the manipulator toward the object, but falls off the further the object is from the center of the workspace - in other words if the object is too far away, the term which pulls the manipulator towards the center of the workspace (Equation 26, the spring element, dominates (here x_w represents the center of the workspace, x_c represents the center of the object, x the endpoint of the manipulator, K_e and K_s gains, and a is a constant determining the fall-off of equation 26 as a function of distance from the object, surface normal n,

$$F_s = K_s n ||x - x_w|| e^{-a(x - x_c)^2} \tag{26}$$

and finally F_t the total virtual force field equation.

$$F_t = F_e + F_s \tag{27}$$

Outside of the workspace, biological system tendons and joint assemblies act as springs, pulling toward the center of the workspace, as in Equation 28.

$$F_t = K_k(x - x_w) \tag{28}$$

These equations can be visualized as in Figure 8. Here it is clear that as the object enters the workspace, the virtual force field acting on the manipulator endpoint correspondingly alters. In the case just described we use (for clarity) circular objects and workspaces, but different shapes can be used by replacing the simple subtraction by a function of any shape.

3.4.4 Low-level

There are advantages to working in endpoint coordinates (Simpkins & Todorov, 2011), one of the most notable being the flexibility in actual structures the control is solving the problem for. In addition, the more challenging parts of the problem are solved for dynamics which are linear.

We here implement a bounded Newton's method (Ferziger, 1998) to compute the optimal feasible torques to apply which map as closely as possible to the desired forces at the endpoints. Each state is initialized with the current state rather than randomly in order to be near the goal initially, and this is quite effective and efficient in practice. When implemented on a physical system, accuracy is further improved by implementing a low level PD controller to track these joint torques more closely, and deal with issues of friction and un-modeled dynamics of the structure or system.

3.5 Safety and real-time control

Human and robot interaction can pose a number of dangers. This is especially true when the humans are interacting with dynamic or powerful robots. Not only do the robots present danger to humans directly, but the unpredictable nature of human behavior is such that they can introduce disturbances into the system which are difficult to account for ahead of time. There are a number of ways to mitigate hazards. One approach is to make the entire robotic system passive - based on inertia, back-drivability, and brakes. Then the human applies all the energy to the system. Though this is powerful, the human-robot system can still create accelerations which can cause injury to themselves or others, and the system is no longer autonomous. The human *must* be in the loop for the system to act. Making the entire robot small and weak, or to mechanically decouple the motors from the human is another approach. This may be effective in some cases, but has limitations as then the robot cannot perform many tasks. In the mechanical coupling case it may be difficult to keep disconnected components from moving unsafely.

In all cases it is beneficial to have a secondary monitoring system which is connected with all the necessary sensors and can control either braking or power to actuators. This 'watchdog' system performs checks on the control system to make sure it does not exceed safety specifications, that all components operate properly, and that the high level control has not crashed. If anything does fail checks, the watchdog can disable the power and stop the system safely. A small processor is all that is required to perform these duties, requiring minimal space and power. The overall architecture for this watchdog approach will be discussed, along with demonstrations of a robot with such a system integrated into the design.

3.5.1 Failure modes of a standard system

Consider a standard engineered control system, as shown in Figure 9(a). The system consists of a mechanical system, sensors, actuators, power input/supply, and some sort of computer system.

Case 1: Sensor failure

Assume this is a feedback system, such as a robotic arm used for high energy haptics. The robot is aware of its current position by a series of position sensors coupled to key joints. This allows the robot to follow a reference trajectory, as in the image sequence in Figure 12. Now consider what happens if one sensor suddenly discontinues to function (Figure 9(b)), or begins

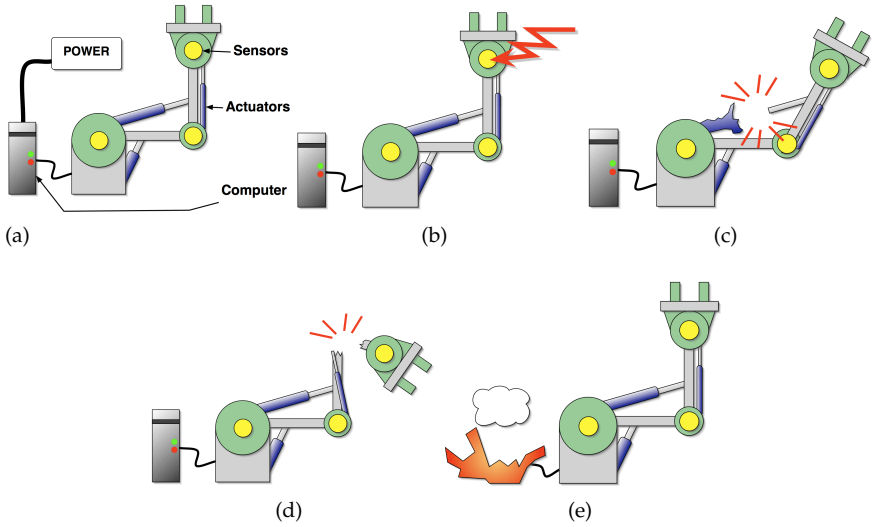


Fig. 9. (a) A typical engineered control system. It consists of a mechanical structure, sensors, actuators, power input/supply, and some computer system. (b)-(e) Some examples of failures of the typical system, and how they can be classified into one of the four cases (power supply omitted for clarity). (b) Case 1 : sensor failure - a sensor stops functioning properly. (c) Case 2 : actuator failure - in this case an actuator breaks, but it could be that it has an internal electrical failure. (d) Case 3 : mechanical failure - here we see that a main mechanical component has dislocated completely. (e) Case 4 : Computer failure - here it seems that the computer has been destroyed, but this type of failure also can be a software crash.

to function abnormally (a wire breaks, it wears out, shock destroys it, a bad ground, etc). The robot control algorithm assumes it is still being fed sensor information, and acts accordingly. This will likely cause a disastrous acceleration until a boundary is struck with maximum force, and the user may be injured.

In the case of a haptic system, it is better to have a motor turn off rather than accelerate uncontrollably, however how does one integrate available information to react properly? There actually is information available to tell us that there is a hardware failure, even if the sensor failure itself is literally undetectable.

Case 2: Actuator failure

There are a number of ways an actuator can fail (Figure 9(c)). An electrical actuator may fail such that it becomes passive, unresponsive, sluggish, or lacking in power output. It may also lock or jam due to some component failure. The internal circuit of the actuator may fail such that it shorts (which tends to make the actuator act as a viscous brake) or has an open circuit, which leads to a non-responsive, passive actuator. A hydraulic actuator may lock, or it may leak fluid which leads to pressure loss and either stiffness or loss of power, depending on where the leak occurs. It may also suffer a compressor failure (which may lead to a total failure of the actuator) or component wear (which may lead to inefficiency or lack of functionality as well). Pneumatic actuators may experience a lock-up, but due to gas compressibility this is less common. Often the failure of pneumatics lead to rapid functionality loss as the gas

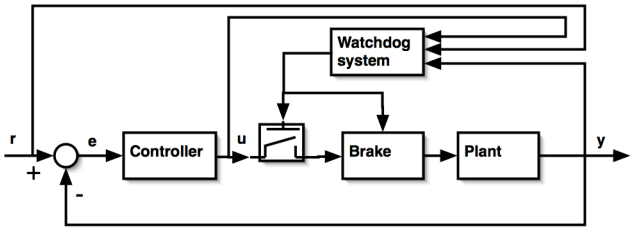


Fig. 10. One possible safety configuration for a watchdog system. In this system there is a simple microprocessor overseeing all safety-critical functions of the main device - sensors, actuators, and boundaries. If anything is incorrect, it can disable the motors and enable a brake directly.

pressure loss may be more rapid than for a fluid-based system. Many pneumatic, electrical, and hydraulic systems include integrated sensors which may fail, as previously discussed in Case 1.

Case 3: Mechanical failure

A mechanical failure is defined by the author as a failure in one or more mechanical aspects of the system. This can be a component breaking or yielding (deforming such that it does not return to its original shape), a mechanical jam, or a physical disconnection which is undesirable, as in Figure 9(d).

Case 4: Computer or electronic failure

A computer failure may occur through a software crash, electrical short or disconnection, a software bug or error, memory corruption, or computational insufficiency, as in Figure 9(e). Any of these can lead to the control system not acting as designed.

Achieving the goal regardless

Ultimately, many failures can be detected and 'handled' in such a way that potentially devastating effects are mitigated. It may be convenient to include as a safety device a redundant low level microprocessor-based 'watchdog' which is independent of the standard system, as in Figure 10. It is connected to all the sensor information and actuators, and has a programmed set of boundaries to maintain. Its job is to oversee all the components of the system and ensure they are functioning. The watchdog and main computer system ping each other frequently in a simple way such as with a pulse train. If there is a failure in either system component (main or watchdog) the system, depending on its function, may be shut down. The watchdog may be connected to the actuators such that it may disable the power output and engage some braking action if desired. It is also generally a high bandwidth system (since it is not performing many computations). If the system is mission-critical, such as in a military application or in a flight control or other situation where it may be dangerous for the system to be suddenly disabled entirely, it may be more effective to determine the safest course of action which still achieves the objective. In the case of a walking robot, where a Case 1-4 failure has occurred in the leg, it may still be possible to walk. Consider humans with an injured joint - they can often offload much of the dynamic load to anther leg or joint. It is simply a matter of an adaptive control algorithm which recomputes a solution for a different system configuration. Consider sending a robot into a situation where it will be gradually melting - a radioactive disaster or extreme conditions. The robot needs to be capable of adapting to loss of actuators, sensors, and some computational capabilities. An algorithm such as discussed

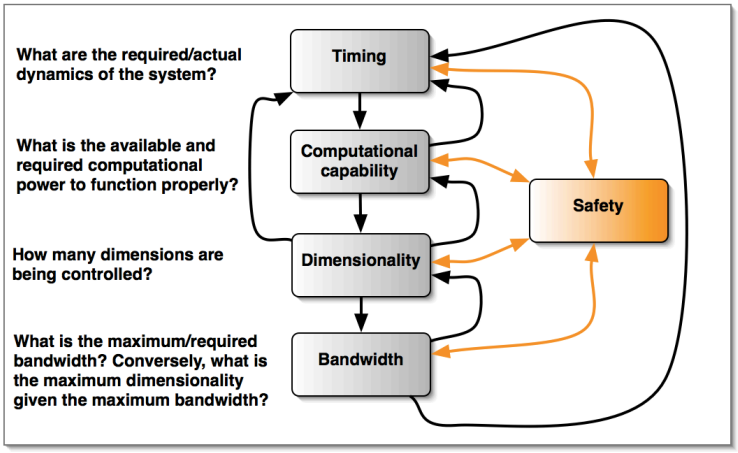


Fig. 11. Suggested flowchart for real-time system design. Note that safety is to be part of each stage of the design process. In addition, bandwidth, dimensionality, and timing are somewhat interdependent, which is the reason that feedback is incorporated in this chart. The first thing to start with as a question is 'what are the dynamics of interest?' This leads to the computations outlined in Section 3.1. Second, one must determine just how many degrees of freedom the system needs to perform all its tasks (Section 3.3), and how complex the control algorithms need to be (and can be, depending upon hardware available). This determines dimensionality, which affects the timing. Bandwidth limitations impose constraints on both dimensionality and timing (slower timing allows for higher dimensionality, and vice versa for faster timing).

in Section 3.4.1 can easily handle these sorts of challenges. Identification of the failure is a challenge. This can be achieved through the integration of the watchdog. A watchdog can be used to compare the measurements with expected outputs - for example a sensor with a zero voltage output which measures velocity of a motor normally, while a motor is being commanded to a high velocity, and is currently not drawing very much current (meaning it probably is not stalled), and has significant back-emf being generated can be a flag for a sensor or actuator failure. All the components of a system work together in unison, and the dynamics of a system express themselves through all the sensors and actuators. Expected values can be integrated into the watchdog in the form of a classifier which can output error states very quickly and efficiently, if well-trained and carefully coded. The watchdog and computer compare error signals, and can eliminate (from the control algorithm), freeze, or otherwise remove from or adjust the dynamics as much as possible to minimize failure impact. Then the control algorithm is simply updated with a new number of actuators or in the case of something similar to locking an injured joint, a new number of degrees of freedom, and it continues to solve the problem in the optimal fashion given the new system. Since everything is solved online this is possible.

4. Putting it all together

Methods have been presented which, when combined, effectively deal with the real-time control problem for complex robotic systems. Since some of the components of the design

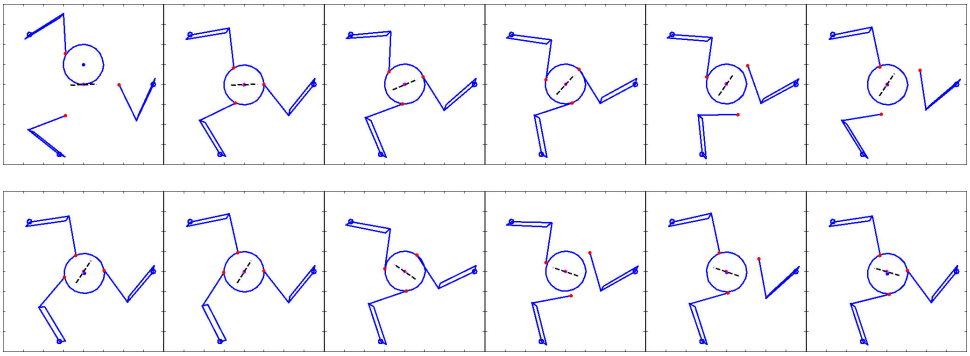


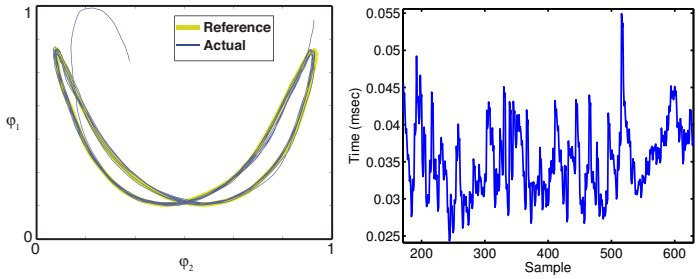
Fig. 12. Animation sequence of three manipulators tracking a rotating stationary trajectory (Frames advance starting from top-left in sequence to the right in the same fashion as normal english text). Note the multiple contact breaks in order to allow the object to be continuously rotated.

depend upon others, Figure 11 suggests a sequence which is typically effective. Each component is interdependent with each other component, however an effective starting point is to define the problem, or class of problems first. This gives a sense of the requirements for timing. Then the designer can compute required bandwidth and dimensionality of the system (as well as constraining the design by the available computational power), or generate something like the sensorimotor system of a human, with a high mechanical dimensionality, but limitations on overall bandwidth and dimensionality of control, leading to optimal actions for a particular task. At each stage of the process, safety engineers or the primary designer can design checks such as a secondary safety system and impose safety-based constraints.
Real-time control itself, in order to be most effective, should not be decoupled from the design of the actual system. Otherwise the system may be unnecessarily limited due to designed-in bottlenecks. However, being aware of the concepts presented in this chapter, a control engineer can effectively maximize the capabilities of a system which already exists and must merely be controlled.

5. Results and application

We begin with a description of the problem, originally described in (Simpkins et al., 2011), and discussed previously in Section 3.4.1. In order to study biological systems, and how they interact with their environment through manipulation and locomotion, it is useful to have a robotic device which mimics certain key aspects of biology (those systems capable of manipulation and dextrous locomotion), such as output-to-actuator backdrivability[1], low endpoint inertia, high dynamic capabilities, and sensitivity. After carefully determining the requirements for manipulation and locomotion, the author and collaborators drew upon biological systems in terms of timing and dynamic capability to define requirements. In addition, there was a desire to create a modular system which could ultimately be assembled quickly into a variety of configurations. There are a variety of requirements determined, and a summary is a 1kHz maximum bandwidth for the control of five fingers through one

[1] Application of force at the output can about a change of state at the input of a back-drivable system.



(a) (b)

Fig. 13. (a) The robot, controlled hierarchically from a PC running matlab and communicating over bluetooth, with a local low level PID controller, is capable of tracking a cyclical motion very rapidly. The timing of the loop for this plot is approximately 1msec, with some variability in timing. (b) In this plot, the loop (which here includes contact dynamics and a more complex simulation) was deliberately executed in a non-optimal fashion in order to vary the timing of the sampling and updates, for demonstration purposes. When such variability is handled properly, as described in this chapter, the system can be built to remain stable.

bluetooth or USB connection, ability to apply approximately 1N of endpoint force, minimal friction and endpoint inertia, resolution above one-tenth of a degree, and measures of force in addition to position. These requirements pose a challenge as communicating position, force, and current for each joint of each finger for five fingers at 1kHz becomes 9000 variables per second, and at 12-bits of resolution, for five fingers, this becomes 540kbps. If the robot is receiving, for example, no more than three variables for command of either position, velocity, or force output, then the most data must be sent from the low to the higher levels, as the serial interface used can simultaneously send and receive data.

There have been a few biologically-based robots recently developed, however they all suffer from some limitation imposed by a lack of an integrative real-time approach such as discussed in this chapter. It is not unusual for a well-designed hand mechanism to have a maximum bandwidth of one-tenth of a Hz to move from open-to-closed. Consider attempting most daily tasks where every motion could be performed in no shorter time than ten seconds. Dynamic manipulation would indeed be impossible! We omit the discussion of the design itself here, shown in Figure 6(a), instead focusing on how to maximize the real-time capabilities.

Figure 12 demonstrates a simulation based upon the real-time concepts discussed in previous sections. This is an implementation of a real-time hierarchical optimal control of a number of manipulators tracking a reference trajectory, and can be applied to the physical robots. In this case the high level is a simple proportional-derivative controller tracking a 'rotate continuously' command. It is clear that the algorithm is capable of multiple contact breaks required in order to allow continuous rotation.[2]

This simulation runs entirely in real-time, with no offline components. With code executing in Matlab R2008b for Macintosh upon an Intel Core 2 Duo processor at 2.33GHz with 2GB of

[2] Further results and animations can be found at the author's home page, http://casimpkinsjr.radiantdolphinpress.com/pages/papersv2.html. In addition, this is the location for all the author's publications, with most available for download.

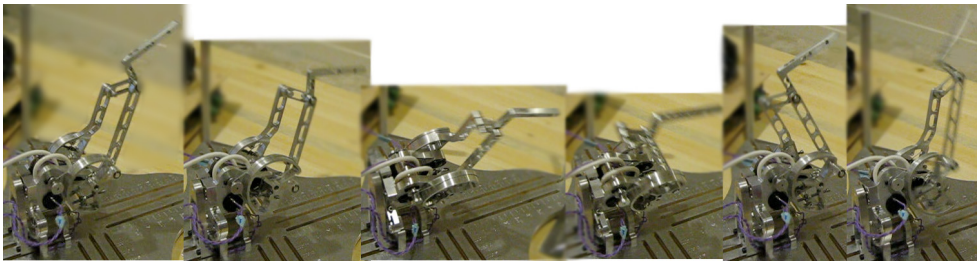


Fig. 14. Image sequence of a bio-mimetic robotic finger designed by the author undergoing real-time control as described in this chapter. The finger is tracking a cyclical shape in 3-Dimensional space at approximately a 10Hz frequency. The robot is controlled wirelessly, through bluetooth, and is communicating 9x12bit values for position, velocity, and coil current at 1kHz, and assuming it is receiving the same amount of data as it is sending (as a worst-case), required bandwidth is 9x12x1000x2 = 216,000 bits per second, or 216kbps. The bandwidth of the connection is 2e6bps, or 2Mbps, which is nearly 10 times greater. In this case the algorithm controlling the finger is not computing any contacts, and so executes in an average of less than 1msec at the PC side.

RAM, each loop takes on average 30msec, as measured from within Matlab (See Matlab help on tic-toc), which is fast enough for real-time manipulation. Human beings have an update rate perceptually of approximately 30Hz, and dynamically of only a few Hz (Nigg & Herzog, 1994). Additionally, it would be straightforward to optimize this code to execute far faster by implementing it in C, for example.

The fully implemented control algorithm including hardware executes in approximately the same time, as the hardware is provided the command computed by the simulation, and the actual object states are fed back into the system to adjust for the difference between simulation and reality. Essentially including the hardware in the loop in parallel with the model adds a model reference control element to the hierarchical optimal control strategy. The majority of the processor time is spent on the contact dynamics computations, and could be further improved to minimize some calculations without changing the algorithm significantly. Therefore when some manipulators are not in contact (as in Figure 14), computation time is reduced dramatically. The 30Hz or even a 200Hz sample rate are well within the 1kHz bandwidth of the robot manipulator and wireless communication system. It is notable that, in Figure 13(b), the variability in timing is less than one sample, and so the state space representations described in Section 3.1 hold. In the case that the manipulator is tracking a trajectory, as in Figure 13(a), we see that this hierarchical strategy is capable of smooth rapid control of the manipulator. If the current Matlab implementation were to be used, and the system were required to perform manipulation at a faster rate than 30Hz, the concepts in Section 3.3 could be more explicitly included in the control strategy of Section 3.4.1 by adding a cost on the number of simultaneous manipulator control inputs. A lower dimensionality means the algorithm will execute faster and be capable of a higher bandwidth, while still being able to increase dimensionality on demand.

6. Conclusion

Modern robotic systems are becoming more and more complex, and requiring far more sophisticated controllers to be designed and implemented than ever before in order to deal with the demand for them to perform more complex tasks. These changes have raised a new challenge for engineers and roboticists. This chapter has presented a first step toward a unifying theory which combines and expands upon modern real-time design control techniques. It broke the problem down into sub-problems which each have a significant body of literature, while describing the fact that these seemingly separate fields are in fact intertwined, are part of a whole, and complement each other. By being at least aware of this method of organizing the problem, and the potential pitfalls associated with the sub-problems, it is possible to improve the end result of a design by making revisions and requirements which are more appropriate far earlier in the process than otherwise possible.
A number of results were presented for the control strategy discussed, as well as an implementation on a real dynamic bio-mimetic robot. Currently, many of the Matlab functions associated with this robot are being implemented in C and made available as MEX files which Matlab can still make use of. This should improve the execution time significantly of the simulation, as well as all other aspects of the computations. These techniques will be part of the development of new skills for these dynamic manipulators. The two dimensional representation has been extended to three dimensions, and will be part of a series of papers to be released soon by the author.
These new challenges require a different overall strategy for problem solving than ever before in engineered systems, but have the potential to truly revolutionize all aspects of human experience. By developing and refining methodologies, we will transition from individual designs which take years of iterative slow improvements with many dead ends to clear paths with fewer roadblocks, and new tools to illuminate the way.

7. References

Anderson, T. W. (2003). *An Introduction to Multivariate Statistical Analysis*, John Wiley and Sons.

Åström, K. & Wittenmark, B. (1990). *Computer controlled systems - theory and design*, 2nd edn, Prentice Hall, Englewood cliffs, NJ.

Bernstein, N. (1967). *The coordination and regulation of movements*, Pergamon, London.

Butterfab, J. & et al. (2001). Dlr-hand ii: Next generation of a dextrous robot hand, *Proc. of the IEEE International Conference on Robotics and Automation* pp. 109–114.

Erez, T., Todorov, E. & Tassa, Y. (2011). Fast model predictive control for complex robotic behavior, *Robotics Science and Systems (RSS'11)* .

Ermedahl, A. (2003). *A modular tool architecture for worst-case execution time analysis*, PhD thesis, Uppsala University.

Ferziger, J. (1998). *Numerical Methods for Engineering Application*, 2nd edition edn, John Wiley and Sons, New York, NY.

Franklin, G. F. & et al. (1994). *Feedback control of dynamic systems*, 3rd edn, Addison Wesley, Reading, MA.

Grob, H.-G. (2001). A prediction system for evolutionary testability applied to dynamic execution time analysis, *Information and Software Technology* (43): 855–862.

Hansen, J., Hissam, S. & Moreno, G. (2009). Statistical-based wcet estimation and validation, *9th International Workshop on Worst-Case Execution Time (WCET) Analysis* .

Krstic, M., Kanellakopoulos, I. & Kokotovic, P. (1995). *Nonlinear and Adaptive Control Design*, John Wiley and Sons.

Li, W. & Todorov, E. (2004). Iterative linear quadratic regulator design for nonlinear bio-logical movement systems, *Proc. of the 1st International Conference on Informatics in Control, Automation and Robotics* **1**: 222–229.

Mardia, K., Kent, J. & Bibby, J. (1992). *Multivariate Analysis*, Academic Press.

Movellan, J., Tanaka, F., Fortenberry, B. & Aisaka, K. (2005). The rubi/qrio project: Origins, principles, and first steps, *IEEE International Conference on Development and Learning* **1**: 80 – 86.

Nigg, B. & Herzog, W. (1994). *Biomechanics*, John Wiley and Sons, New York.

Nilsson, J. (1998). *Real-time control systems with delays*, PhD thesis, Dept. of automatic control, Lund institute of technology, Sweden.

Puschner, P. & Nossal, R. (1998). Testing the results of static worst-case execution-time analysis, *Proceedings of the 19th IEEE Real-Time Systems Symposium (RTSS98) Madrid*.

Santello, M., Flanders, M. & Soechting, J. (1998). Postural hand synergies for tool use, *Journal of Neuroscience* (18): 10105–15.

Simpkins, A. (2009). *Exploratory studies of sensorimotor control and learning with system identification and stochastic optimal control*, PhD thesis, University of California, San Diego, La Jolla, CA.

Simpkins, A., Kelley, M. & Todorov, E. (2011). Modular bio-mimetic robots that can interact with the world the way we do, *Proceedings of the IEEE International Conference on Robotics and Automation* .

Simpkins, A. & Todorov, E. (2011). Complex object manipulation with hierarchical optimal control, *IEEE Symposium of Adaptive Dynamic Programming and Reinforcement Learning, IEEE Computer Society, Paris, France*.

Stengel, R. (1986). *Stochastic Optimal Control: Theory and Application*, John Wiley and Sons.

Todorov, E. (2004). Optimality principles in sensorimotor control, *Nature Neuroscience* **7**: 907–915.

Todorov, E. & Ghahramani, Z. (2003). Unsupervised learning of sensory-motor primitives, IEEE, *In Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*.

Todorov, E. & Ghahramani, Z. (2004). Analysis of synergies underlying complex hand manipulation, IEEE, *Proceedings of the 26th Annual International Conference of the IEEE EMBS, San Francisco, CA, USA*.

Todorov, E., Hu, C., Simpkins, A. & Movellan, J. (2010). Identification and control of a pneumatic robot, *Proc. of the fourth IEEE RAS / EMBS International Conference on Biomedical Robotics and Biomechatronics* .

Todorov, E. & Jordan, M. (2002). Optimal feedback control as a theory of motor coordination, *Nature Neuroscience* **5**(11): 1226–1235.

Todorov, E. & Jordan, M. (2003). A minimal intervention principle for coordinated movement, *Advances in Neural Information Processing Systems* **15**: 27–34.

Vandeweghe, J. M., M.Rogers, M.Weissert & Matsuoka, Y. (2004). The act hand: Design of the skeletal structure, *Proc. of the IEEE International Conference on Robotics and Automation*.

Wilhelm, R., Engblom, J., Ermedahl, A., Holsti, N., Thesing, S., Whalley, D., Bernat, G., Ferdinand, C., Heckmann, R., Mitra, T., Mueller, F., Puaut, I., Puschner, P., Staschulat,

J. & Stenström, P. S. (2008). The worst-case execution time problem - overview of methods and survey of tools, *ACM Transactions on Embedded Computing Systems* **7**(3).

Wilkinson, D. D., Vandeweghe, J. M. & Matsuoka, Y. (2003). An extensor mechanism for an anatomical robotic hand, *Proc. of the IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 238 – 243.

Wittenmark, B., Nilsson, J. & Torngren, M. (1995). Timing problems in real-time control systems, *In Proceedings of the 1995 American Control Conference. Seattle, WA* .

Zhou, K. & Doyle, J. (1998). *Essentials of Robust Control*, Prentice Hall, Upper Saddle River, NJ.