Towards methods for robotic systems capable of human-level dexterity in manipulation and locomotion

Alex Simpkins[1]

[1]*University of Washington, AC101 Paul G. Allen Center, Box 352350, 185 Stevens Way, Seattle, WA 98195-2350*
……

Abstract: This paper presents a method for control of robotic systems that can potentially fill the need for autonomous systems that are more capable than the current state of the art. It extends prior work by the author and discusses how this method can not only solve a single control problem for a single task, but also address the challenge of creating a system that will work for a variety of tasks without reprogramming or redesigning. The entire problem can be solved in real-time, which allows for flexibly adding or removing manipulators, changing strategies, and more. Several experiments are described that demonstrate the effectiveness of this method for manipulating objects using any number and design of manipulators.

Keywords: Manipulation, locomotion, hierarchical control, robotics, mechatronics, optimal control, compliance

1 Introduction

1.1 Problem description

This paper presents a method for control of robotic systems that has the potential to produce the breadth of behaviors necessary for autonomous systems to seamlessly integrate with society, as well as address challenges currently not possible with technology. Presently there is a serious need in mechatronics, control, neuroscience, and robotics to develop algorithms that are capable of controlling complex robotic systems in unstructured environments and for poorly defined problems. This requires a solution that addresses issues of control, computational complexity, uncertainty, combinations and switching of behaviors, and cooperation between possibly multiple manipulators. The system may have nonlinearities, stochasticity, and changing dynamics.

1.2 Background

Automatic control systems and mechatronics became popular in the 1970's with the successful application to manufacturing using industrial robot arms and pre-programmed trajectories. Robots have been used for a variety of repetitive operations such as cutting, gluing, welding, basic assembling in very specified ways, lifting, moving, and positioning of objects on assembly lines. However, such methods fail in the case of poorly controlled environments with uncertainty or for problems that are difficult or complex to break down into very specific pre-planned steps. It is important to address these failures because in the near future there is a large portion of the population that will be approaching retirement age, and not enough skilled people to replace them, and because manufacturing does not just consist of single products produced consistently for a long time but a more dynamic and changing product line. In addition, there are many people with need of effective artificial limbs that behave more like the original limb, and people with brain injuries

who cannot effectively control their bodies.

With many recent advances in computational capability, control theory (in terms of optimal, robust, model predictive, nonlinear, and stochastic control, as well as high dimensional multi-input-multi-output systems), advances in sensing and actuation, there is hope for addressing these problems.

The computer graphics field has some recent advances in locomotion and manipulation that may translate to mechatronics and robotics technology. Specifically the work of [25][10] which involves dynamic controllers that are not necessarily (or have a goal to eventually not be) based upon motion capture. Several earlier methods and parallel branches develop controllers by performing motion capture on humans and extracting controllers[8][2] which mimic the goals of the humans, though the extracted cost functions are not likely to be the same functions as the solution is non-unique. More recently some groups have applied and extended methods originally developed for the chemical process industry several decades ago based upon model predictive control (MPC). This tends to produce effective trajectories in simulation, though it is not yet clear how well this will transfer to real systems since the simulated systems typically lack many limitations inherent in real systems such as force capability, range of motion, complex friction functionals, power density for actuation, issues of digital control (sampling, aliasing, bias, noise, real-time computation, nonlinear physics). Careful comparisons between simulated contacts and true contacts do not appear to have been performed either for these methods, though they eventually will be and the methods adjusted. Typically real systems also have significant damping, so the algorithms may overcome the previously mentioned limitations and transfer well. The question is whether the available hardware is capable enough, which is why the author has been developing hardware [21] that can effectively apply algorithms developed here.

There are several other limitations of current approaches. Those are approaches to dynamic manipulation and locomotion where current approaches can, for example, walk in a straight line along a pre-programmed trajectory, or within certain limited uncertainties, however sensing is a challenge and typically these algorithms either do not work in real-time or require perfect perception, which is an impossibility. The work of [25] and [13] must be computed on a powerful grid for several hours if not days until a viable solution is found. Biological systems tend to mix perception and action in very intelligent ways, and so it is reasonable to assume they are not necessarily separating measurement from control. Additionally, one can effectively manipulate objects and perform locomotion without visual feedback (most 'perfect perception' paradigms used today are motion-capture-based, which is not without its own problems and expenses).

Learning is also a challenge. Many times human beings encounter new situations or environments with unknowns that they must address when attempting to achieve a goal[19]. This can be very difficult, and there are some adaptive and learning algorithms, but the capability is still in its infancy, nor is there a general 'learner.'

High dimensional systems, which manipulation paradigms typically encompass, are very difficult to solve in real time especially when systems are coupled. This is partly due to the large number of variables. For example, a discretizing approach using dynamic programming for the problem presented in this paper would have to be, even for only three manipulators, 24-Dimensional. This means, if the state space were only discretized (quite coarsely) into 10 bins per dimension, and 32-bit variables are used, the memory required to solve the problem would be no less than 2.3e25 bits. That is 1.25e14 GB of memory storage. Not only is this much more than is reasonable presently by far (this would encompass more desktop systems than currently exist on the

planet and consume more than the entire energy consumption of the earth currently – 8e3 TW, with total estimates of energy consumption in 2008 of approximately 15TW), but that much data could never be transferred, and even if it could there is no computer or super computer that comes close to being capable of processing that many computations. The algorithm presented in this paper can run in real-time on a five-year-old laptop, far from the most powerful computer system.

It appears that coupling degrees of freedom together in a dynamic fashion can actually help simplify the control algorithms rather than add complexity. It is important not to attempt to cancel coupling but rather to use it[16][18][15]. In fact, some researchers are developing hardware to study how the human musculoskeletal structure's coupling works, and thus far they are finding the coupling in biological systems is very useful[24]. The methods presented here can make use of coupling in a system, though this is not the present focus here.

Most current approaches do not address stability very well, if at all, and this is a very serious problem. Often stability is only defined within the context of something appears to work or does not appear to work. Such risks cannot be taken when, for example, a system is disarming a dangerous explosive device, performing a medical procedure, driving a car autonomously, helping an elderly person walk, or manipulating very expensive and rare components for space vehicles. Computational guarantees of stability are necessary and generating algorithms that can find a guaranteed feasible stable solution will help make these methods applicable to manufacturing and areas where safety can be a concern. This paper begins to address this issue by using, at each stage, methods that have theoretical guarantees of stability that are well developed in the literature.

Grasping itself, without manipulation has a wide body of literature, and many approaches[12][6]. Again this is rarely a real-time-solved problem, and once a grasp is completed, the object and hand typically act as a rigid body, which is not as significant here, since such problems can be solved 99% of the time with a one degree of freedom gripper, and dynamically controlling an object's trajectory (potentially independently of the arm trajectory) allows for a more rich interaction with the world than hand-object rigid body motion alone (rigid grasping is also significant). The work presented here can be used for both rigid body type grasping *and* dextrous manipulation.

In general most solutions in the literature to manipulation and locomotion are hand-crafted (in fact, it is referred to often as a 'black art' because it takes tremendous skill and intuition to do this successfully), and though this may be an effective means of solving a single problem, these methods are very limited when it comes to addressing a generalized problem of manipulation or locomotion where many problems must be solved simultaneously or mixed, and it may not be clear ahead of time *what* must be solved. There is room for handcrafted solutions, for example humans learn new skills that are specific to that skill's paradigm. Humans also appear to have a general capability for manipulation and locomotion that may not be optimal for each task individually, but can be applied to many tasks to achieve a goal. The method presented here can not only seamlessly integrate specific hand-crafted skills, but can also be one of the first approaches to a generalize-able approach. In this way we present a methodology that does not tear down but rather provides a means by which the best approaches can be integrated into one system, producing a superior dexterity to any one narrow method.

1.3 Contributions

The contributions of this paper are summarized below.

- This paper expands upon, and contributes a new compact representation of the hierarchical optimal control (HOC) strategy for the locomotion and manipulation problems presented in

[20].

- Other methods can be substituted anywhere within the hierarchical strategy without redesigning the entire system.
- The concept of using more complex control at the high level allows for the higher dimensional lower levels to be solved with a virtual force field (described in [20]) or other control strategies which are more conducive to real-time implementation on microprocessors than processor intensive iterative local methods such as iterative linear quadratic gaussian or differential dynamic programming[9] without loss of the benefits of complex, possibly predictive behaviors.
- A novel means of assuring the object is not dropped (or that the system remains balanced) is presented.
- This control approach will allow for successful manipulation and locomotion with minimal external sensing (parallels the process of proprioception in biology). Current approaches (which have had limited success) rely on direct state feedback regarding the object or ground, including shape, mass, inertia, and position parameters. Typically this requires an expensive and complex motion capture system such as a VICON or Phasespace system, limiting activity to a laboratory.
- An improvement over the original algorithm upon which this is based. An orthogonal velocity term is added to the force field functions, improving the optimality of the contact point selection after a contact break, and stability of the contact breaking behavior.

1.4 Outline

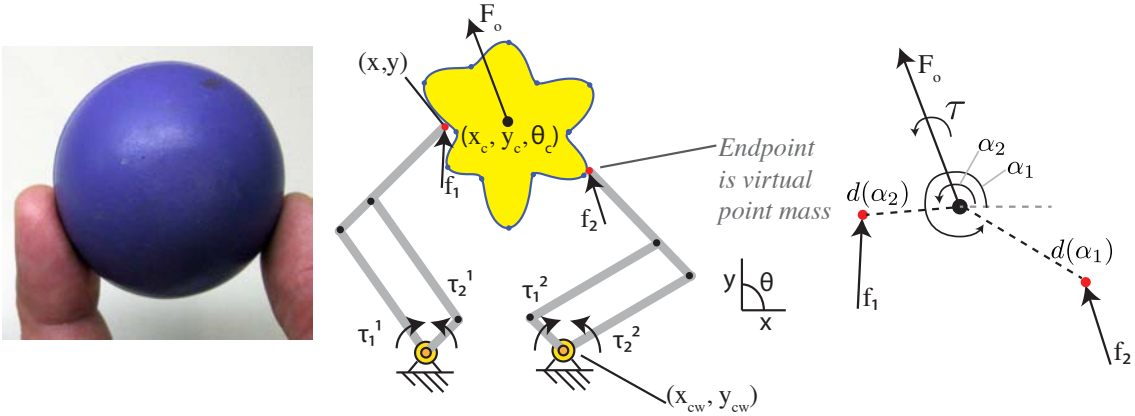
The rest of this paper is organized as follows. Section 2 describes the overall model of the system dynamics in this approach, as well as the control strategy and several experiments. Section 3 discusses the results and an analysis of said results. Finally, Section 4 presents a conclusion, evaluation, and description of the future of this work.

2 Methods and Materials

2.1 Dynamic Model

The system shown in Figure 1 is modeled by discrete shapes that are represented by B-spline curves[14] (which has the advantage of a fast convex hull computation for the contact algorithm). Each object has mass and inertial properties, a center of mass, and orientation relative to the coordinate system origin. The robotic manipulators are modeled by point masses at the endpoints, as in [20]. This has several advantages, not the least of which are linear dynamics and simpler collision detection.

The model and dynamics computations are direct extensions of the two dimensional model originally introduced in [20]. The linear point mass dynamics significantly simplify the contact computations.



(a) (b) (c)

Figure 1: (a) Two human fingers grasping an object during manipulation. (b) 2-Dimensional representation of the system. (c) Diagram of the system, showing forces due to manipulators, contact locations, and resultant force.

2.1.1 High-level dynamics

The high level dynamics are simply given by the sum of forces acting about the center of mass of the object (Figure 1c),

$$\begin{Bmatrix} \sum F_{x\rho} \\ \sum F_{y\rho} \\ \sum M_o \end{Bmatrix} = \begin{bmatrix} W & 0 \\ 0 & W \\ -d_y(\alpha) & d_x(\alpha) \end{bmatrix} \begin{Bmatrix} f_x \\ f_y \end{Bmatrix} - \begin{bmatrix} m_o & 0 & 0 \\ 0 & m_o & 0 \\ 0 & 0 & J \end{bmatrix} \left(\begin{Bmatrix} \ddot{x}_o \\ \ddot{y}_o \\ \ddot{\theta}_o \end{Bmatrix} - \begin{Bmatrix} 0 \\ g \\ 0 \end{Bmatrix} \right) + \nu \begin{bmatrix} \dot{x}_o \\ \dot{y}_o \\ \dot{\theta}_o \end{bmatrix}, \tag{1}$$

where F represents force, m_o represents object mass, $d(\alpha)$ represents contact locations for those manipulators in contact, W is a row vector of ones length equal to the number of manipulators, f_x and f_y the manipulator applied force vectors each of length equal to the number of manipulators, the last term is a small velocity damping to increase simulation stability. x, y, and θ represent position, and the respective time derivatives velocity and acceleration. At the high level, all forces are ideal, there are no manipulators, and so Equation 1 reduces to,

$$\begin{Bmatrix} F_{x\rho} \\ F_{y\rho} \\ M_o \end{Bmatrix} = \begin{bmatrix} m_o & 0 & 0 \\ 0 & m_o & 0 \\ 0 & 0 & J \end{bmatrix} \left(\begin{Bmatrix} \ddot{x}_o \\ \ddot{y}_o \\ \ddot{\theta}_o \end{Bmatrix} - \begin{Bmatrix} 0 \\ g \\ 0 \end{Bmatrix} \right) - \nu \begin{bmatrix} \dot{x}_o \\ \dot{y}_o \\ \dot{\theta}_o \end{bmatrix}. \tag{2}$$

This equation is re-arranged to have acceleration as the output, and force as the input,

$$\begin{Bmatrix} \ddot{x}_o \\ \ddot{y}_o \\ \ddot{\theta}_o \end{Bmatrix} = \begin{bmatrix} m_o & 0 & 0 \\ 0 & m_o & 0 \\ 0 & 0 & J \end{bmatrix}^{-1} \left(\begin{Bmatrix} F_{x\rho} \\ F_{y\rho} \\ M_o \end{Bmatrix} - \nu \begin{bmatrix} \dot{x}_o \\ \dot{y}_o \\ \dot{\theta}_o \end{bmatrix} \right) - \begin{Bmatrix} 0 \\ g \\ 0 \end{Bmatrix}. \tag{3}$$

This facilitates the state space formulation developed below.

Deterministic with no learning

Let the continuous state be given by

$$q = \begin{bmatrix} x & y & \theta & \dot{x} & \dot{y} & \dot{\theta} \end{bmatrix}^T. \tag{4}$$

Then after some algebra, we can arrive at a state space representation of the system, with matrices

$$A = \begin{bmatrix} 0^{3x3} & I^{3x3} \\ 0^{3x3} & 0^{3x3} \end{bmatrix}, M = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & M^{-1} \end{bmatrix}, u = \begin{bmatrix} 0 & 0 & 0 & F_x & F_y & F_\theta \end{bmatrix}^T. \tag{5}$$

Here we drop the small o, since the high level only refers to the object dynamics. The closed

loop continuous state equation (with an as-yet undetermined controller $\boldsymbol{u}$) is then given by

$$\dot{q} = Aq + Bu\,. \tag{6}$$

The discrete counterpart is given by

$$q(k+1) = \Phi q(k) + \Gamma u(k)\,, \tag{7}$$

where

$$\Phi = I + AT\Psi, \quad \Gamma = \Psi TB, \quad \Psi = I + \frac{AT}{2!} + \frac{A^2T^2}{3!} + ..., \tag{8}$$

which we truncate at a second order approximation, since T becomes approximately zero after raising to the third power. We can also approximate the continuous system with an Euler (or other) approximation, however this relies on a fast sample rate that may not be possible in the case of a high dimensional system running MPC and solving the entire problem in real-time. Ideally we can compute the various matrices exactly without assumption.

Observations are provided by, with C in the general case an appropriately sized identity matrix,

$$p = Cq\,. \tag{9}$$

Stochastic systems

One can also derive a stochastic system at this level and instead of using an LQR, use an LQG with a Kalman filter[7][22] to estimate the state. This is left for a future publication.

2.1.2 Low-level dynamics (including external ideal point mass 'manipulators')

The balance of forces for the manipulators represented in Figure 2, is given by,

$$\sum F = b_i - m_i g - f_i = m_i \ddot{q}_i\,, \tag{10}$$

which can be simplified to become

$$\ddot{q}_i = \frac{1}{m_i}\left(b_i - f_i\right) - g, \tag{11}$$

where the variables (f_i for resultant force from the contact, b_i for the external force applied to the point mass, and $\ddot{q}_i$ the double-differentiated position vector of the point mass) are all 2-D vectors in the x and y directions, and g is a vector for gravitational acceleration pointing in the negative y direction. When not in contact, the resultant force due to contact (f_i) will be zero. We apply a no-slip condition, as in [20], which provides a simplifying constraint,

$$\begin{bmatrix} a_{x_i} \\ a_{y_i} \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x(\theta_i) \\ 0 & 1 & d_y(\theta_i) \end{bmatrix} \begin{bmatrix} a_{x_o} \\ a_{y_o} \\ \ddot{\theta}_o \end{bmatrix}. \tag{12}$$

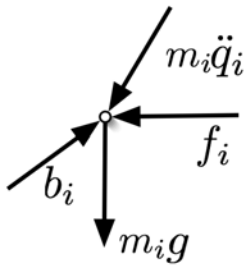


Figure 2: Dynamic balance of forces for point mass representation of manipulators.

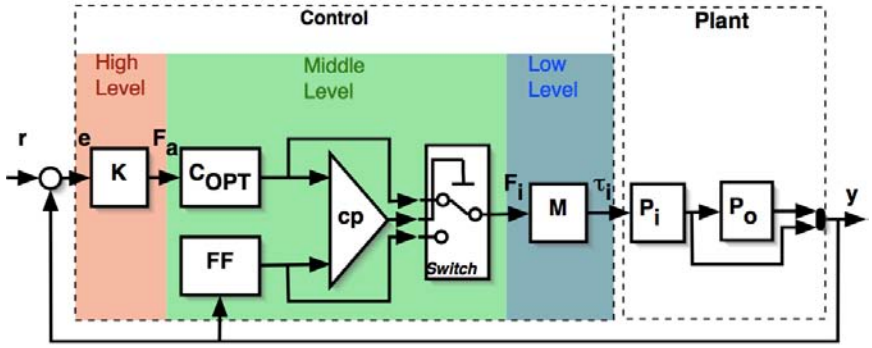


Figure 3: Overall hierarchical optimal control scheme. The high level computes the ideal reference force, the middle level determines the optimal configuration and force distribution, and the low level executes and maps the force to the actual structure. The plant here consists of the dynamics of the manipulators and object, whether they are in contact or not.

The mid-level system dynamics are found by augmenting the object dynamics with the manipulator dynamics, the latter given by,

$$f_i + m_i a_o + m_i \ddot{\theta}_o d(\alpha_i) = b_i + m_i g\,. \tag{13}$$

Then the entire system can be reduced at each time step to a least squares problem in standard form,

$$Nf = b, \tag{14}$$

with N defined as

$$N = \begin{bmatrix} 1 & 0 & -m_o & 0 & 0 \\ 0 & 1 & 0 & -m_o & 0 \\ 0 & 0 & 0 & 0 & J_o \\ I^{m\times m} & 0^{m\times m} & m_i & 0 & m_i d_x(\alpha_i) \\ 0^{m\times m} & I^{m\times m} & 0 & m_i & m_i d_y(\alpha_i) \end{bmatrix}, \tag{15}$$

the unknown forces and accelerations f,

$$f = \begin{bmatrix} f_{x_i} & f_{y_i} & a_{x_o} & a_{y_o} & \ddot{\theta}_o \end{bmatrix}^T, \tag{16}$$

and external forces b given by,

$$b = \begin{bmatrix} 0 & g & 0 & b_{x_i} & b_{y_i} + m_i g \end{bmatrix}^T. \tag{17}$$

The solution is then computed by

$$f = N^+ b, \tag{18}$$

using the `pinverse()` command in Matlab.

This representation is used to compute the update to all the dynamics of the system in an efficient simultaneous fashion. The matrices change size as appropriate depending on which manipulators are in contact to save computation when possible.

2.2 Control

The control structure is a hierarchical optimal control scheme, shown in Figure 3. This has the advantage of modularity. Any control strategy can be inserted at any point. Additionally, the simplified high-level dynamics can employ a more complex algorithm such as a model predictive controller that plans into the future. This controller need not concern itself with the low level computations, thus making it simpler to implement this scheme on real-time systems.

One might argue that one needs to build a controller which generates trajectories at the lower levels as well in an MPC sense, however, we suggest that MPC is only required at a high level. It is also possible to use MPC at all levels, however it is very computationally expensive and not yet

practical for implementation on real control systems that need to run on compact equipment. With the advances of computer systems this will likely be possible eventually, however there is little evidence that biological systems approach control this way, and there is tremendous support for a hierarchical approach. At times everything can be done in feedback mode only – in fact the results in this paper are for feedback mode. A complete work carefully comparing the two paradigms is forthcoming.

2.2.1 High level

The high level control, which has reduced dimensionality (in the case of a 2D system, only 6D state space), is simple enough for complex real-time control strategies. Here we compare feedback and predictive control. The feedback controller is an optimal infinite horizon Linear Quadratic Regulator [23], which is given in continuous time as the controller that minimizes the cost function for the high level system previously defined as

$$J = \int_0^\infty \left(x^T Q x + u^T R u \right) dt \,, \tag{19}$$

with the controller

$$u = -Kx \tag{20}$$

The optimal LQR controller is a well-established control method, so the interested reader is referred to [23] for good background information. The important concept to understand is that the controller makes decisions (maps states into actions) such that for each control action, the cost function defined in Equation 19 is minimized.

To create a model predictive control strategy, one iteratively runs the above algorithm faster than real-time between each time-step, simulating the system into the future based upon actions that the controller will take. Then a more optimal choice can be implemented than if the controller were to act assuming its action would be perfect. While this is useful and has been implemented very successfully in the chemical process industry (which has time scales on the order of hours or even days, and fairly low dimensionality), it has only recently begun to be applied to systems with faster dynamics and more complexity.

The high level system can incorporate low level position information, as well as an estimate of the object state; however in the examples developed in this paper, this is not necessary. In fact, the simpler approach is quite effective in many cases.

2.2.2 Mid-level: Virtual force field and distributing contact forces via fast convex optimization

The mid-level control has two parallel components – a policy-space virtual force field, and a formulation of convex optimization. Then they 'compete' with each other in order to produce a behavior – apply force to the object or pull away from the object in order to find a better contact location.

Convex optimization

Given the number of manipulators in contact, a maximum force capability, and the fact that manipulators can only apply forces to the object pointing toward the object interior, not away from it, we can formulate a convex optimization over the forces each manipulator should apply. By formulating this problem within a convex optimization framework, the problem can be solved with minimal computation time. Another advantage is that a convex optimization guarantees a globally optimal solution. The criteria to minimize are

$$\zeta\left\|\sum F\right\|^2, \quad \beta\left\|\sum F - F_a\right\|^2, \tag{21}$$

which balance the minimization of the sum of the forces (first term) with the minimization of the error between the ideal high level force and sum of forces (second term). Another possible criterion to add is to minimize the maximum force, but this does not appear to be necessary to compute a useful set of forces. Since the sum of two convex sets is also a convex set, this remains a convex optimization problem for multiple criteria. The minimization problem becomes, in standard form [1],

$$\begin{aligned} \min \quad & \varepsilon = F^T G F \\ & F \bullet n < 0 \\ & F_i < F_{\max}, \quad 0 < i \le m \end{aligned}, \tag{22}$$

where G is given by, with $I^{m\times m}$ representing the identity matrix, m the number of manipulators, $W = (1)^{1\times m}$ a vector of ones, the same constants ζ and β introduced above, and the vectors of contact positions relative to object center *dx* and *dy* of all manipulators in contact,

$$G = \begin{bmatrix} \zeta I & 0 & 0 & 0 & 0 \\ 0 & \zeta I & 0 & 0 & 0 \\ \beta W & 0 & -\beta & 0 & 0 \\ 0 & \beta W & 0 & -\beta & 0 \\ -\beta dy & \beta dx & 0 & 0 & -\beta \end{bmatrix}. \tag{23}$$

At this point detail can be added to the model to account for time delays inherent in any physical system. This will serve to smooth out the forces computed by the optimization, and serves to broaden stability margins, but is not entirely necessary unless stability cannot be achieved with a very aggressive action. In this approach we favour simplicity as much as possible, so this is not included here, but if one were to include such an addition, it would be a series element after the force is computed (essentially a low pass filter).

Policy-space virtual force field

The concept of a virtual force field was presented in detail in [20], but will be briefly described here for clarity, along with some expansion. The concept is derived from an empirically based approach. Consider manipulating an object – place on your hand some normal everyday object you interact with, such as a cellular telephone. Rotate the phone with your fingers while holding your arm stationary. At some point your fingers must change position in order for the phone to rotate continuously. When you initially pick up the phone with the intent of manipulating it with your fingers, you grasp the phone in a fairly even pattern, spacing your fingers such that they may most effectively use their individual workspace (to produce as broad a workspace as possible). As you rotate the phone and approach a boundary of the workspace of your fingers, one can model the desire to change finger contact position with the phone as if there were a force field acting on your fingers attempting to either pull the finger away or toward the object. By behavioural observation, a virtual force field can be constructed which produces both approach and contact breaking behaviours with a single equation. This behaviour can also be modelled by optimal control, the comparison between approaches can be compared as the force field approach being a direct policy-space approach versus the optimal control approach being indirect, instead deriving behaviours from the cost function terms. It is well known that tuning cost functions is more difficult than direct policies because of the indirect effect of changes in cost terms.

The important terms must address the following behaviours:

a. *Drawing fingers toward an object to grasp it at an appropriate location*
b. *Breaking contact with the object when necessary (near workspace boundary)*
c. *Ensuring stability of object grasp during contact breaking*

The term necessary for (a) is given by,

$$F_e = K_e \frac{(x - x_c)\beta}{1 + \|x - x_w\|^2}, \quad \beta = \min\left\{ e^{\left[b\left(\|x_c - x_w\|\right)\right]^{-1}}, \ 1 \right\}, \tag{24}$$

which draws the manipulator toward the object center. The denominator assists with avoiding the tendency of the manipulator to reach toward an object completely out of range (this term is brought to zero as the object moves far away from the workspace). Thus if an object which was, for example, thrown into the air in order to be caught again, the manipulators will position themselves near the center of the workspace and only begin to reach toward the object as it begins to approach a reachable location. This models biological behavior, as well as being an effective means of dealing with objects coming into and out of controlled space. It is also not only a waste of energy to reach toward an object that is out of reach and follow it, but also more likely to cause damage in the event of a collision – some slight bending of the joint initially prevents joint lock and higher impact forces (those readers who have ever 'jammed' a finger and then been injured for several days as a result understand this from experience).

A second term (velocity damping in the direction orthogonal to the line between center of manipulator workspace and object center of mass) avoids overshooting during rapid make-and break-contact trajectories.

$$F_d = -K_d \left(\frac{\dot{x} \bullet (x - x_w)}{(x - x_w) \bullet (x - x_w)} \right) (x - x_w). \tag{25}$$

The velocity-dependent term is actually optional but increases robustness of the algorithm.

Criterion (b) is satisfied by

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

and the terms are summed to determine the total force vector acting on a particular manipulator (note that each manipulator can conceivably have a different equation, or the relation can be changed during manipulation. Thus, each manipulator can perform different functions, such as a thumb in a hand, which may behave slightly differently than the other fingers),

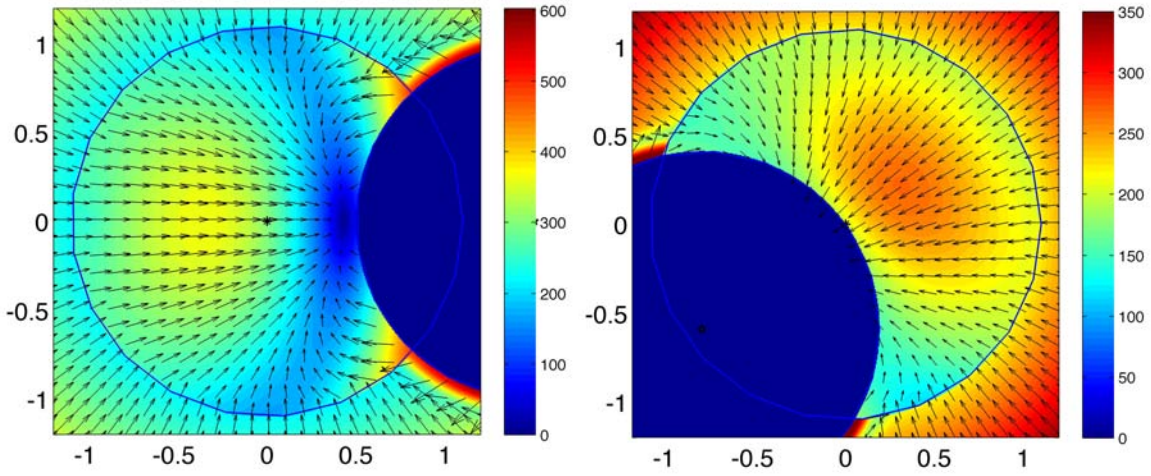
$$F_t = F_s + F_e + F_d. \tag{27}$$

The workspace range of motion of the manipulator is enforced, similar to biological systems, by a simple spring force that pushes a manipulator back into the workspace,

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

The field is visualized in Figure 4 for a single manipulator with an object to be manipulated in Figure 4, along with a vector field displaying the resultant virtual force direction and amplitude.

Criterion (c) need not be explicitly defined for successful manipulation, however it can add to stability by ensuring that contact breaks occur at a stable point (such that the other manipulators are in control of the object). Though the force field relation has been constructed empirically, it can be directly identified from data using system identification [11], inverse optimal control, or similar procedures.



(a) (b)

Figure 4: Visualization of force field function. The two axes are the x- and y-axes, respectively. The center of the image is the center of the workspace, which is outlined. The solid colored shape is the object entering the space. Note the direction of the arrows, which represent the force vector. The color scale represents the amplitude of the virtual force. The manipulator is drawn toward the zero point in the force field. In the case of the left sub-figure, if the manipulator were near the left workspace boundary (at x={-0.75, 0.0}, for example, it would clearly be drawn toward the zero force area, at roughly x={0.5,0.0}.

In order to release an object, the term pulling the manipulators toward the object can be made zero by making the gain K_e zero for that term, along with a gain on the convex optimization output (or the switch can be forced to select contact breaking, which saves some computation time as the convex optimization can be bypassed). Then the force field will pull the manipulator away from the object no matter the state of the system if the fingers are in contact. If one desired fine control over such behaviors, one could augment the state with the force field gains. This would allow complex sliding behaviors (if more complex friction models were used than are presented here).

Non-circular workspaces and object shapes can be addressed by replacing distance terms with functions of the variables. So, for example, Equation 24 can be adjusted to function optimally for very different shapes by replacing the numerator with

$$F_e = K_e \frac{\phi(x,x_c)\beta}{1+\psi(x,x_w)}, \quad \beta = \min\left\{ e^{\left[b(\|x_c - x_w\|)\right]^{-1}}, \quad 1 \right\}. \tag{29}$$

The force field will also work for a variety of shapes using Equation 24, but adding a functional for shape ensures a more optimal contact breaking behavior and contact location selection, especially for very exaggerated shapes.

2.2.3 Low level: Cartesian space feedback and inverse kinematics

The equations relating the Cartesian space endpoint location relative to the base and the link lengths and joint angles for the structure in Figure 1 are given by, with lengths L, angles θ, and position relative to the manipulator base *X,*

$$X = \begin{bmatrix} \mathrm{L}_1\cos(\theta_1) + \mathrm{L}_5\cos(\theta_2+\theta_5) \\ \mathrm{L}_1\sin(\theta_1) + \mathrm{L}_5\sin(\theta_2+\theta_5) \end{bmatrix}. \tag{30}$$

Given an initial guess (initialized from the previous solution) for the angles, we use Newton's method [3] to quickly iterate and converge to a set of angles within the joint constraints. The robot modelled has no singularities, and so each joint angle set is a unique solution. Due to this and the good initialization, the method converges in two to three iterations each time. Then the required two-element torque vector for manipulator *i* is computed directly once the angles are known by,

$$\tau_i = \begin{bmatrix} 0 & F_{i_x} \\ -F_{i_y} & 0 \end{bmatrix} X_i. \tag{31}$$

This provides the required torques for the actuators of the manipulator to generate for the particular configuration applied here. Note that one can apply the Cartesian forces to any appropriate structure, which allows for diverse morphology in terms of application of this control approach. In other words, the actual manipulator can have two joints or twenty, the desired force will still be computed in the same way, and the only thing that changes is the torque mapping.

2.3 Experiments

The experiments performed with this HOC algorithm need to determine several things:

a. *That the manipulators can grasp an object from a variety of initial configurations*
b. *The object can be made to follow an arbitrary trajectory without loss of stability and with sufficient performance*
c. *The closed loop system is an improvement over the open loop system*
d. *The modularity concept in terms of number of manipulators is functional*

2.3.1 Experiment 1: Algorithm grasping from a variety of initial conditions

In this experiment the system is initialized with the object in a variety of locations, and the manipulators in a variety of randomly selected positions. The manipulators in this case are not learning, and assume the shape is a circle. The shape parameters are radius 1m, mass 1kg. The object center position varies from values of approximately [-0.7,0.7] meters/radians (as appropriate). The simulation time-steps are 1msec, integrated with the Symplectic Euler method [5][4], a balance between a stable and simple integration scheme.

2.3.2 Experiment 2: Tracking, algorithm manipulating known shape, high-level feedback

In this experiment the objective is to track an arbitrary reference trajectory for the object. There are two references provided, the first a randomly varying trajectory, which is driven (at the acceleration level) by the double integration of zero-mean Gaussian noise with a covariance equivalent to the time difference between samples, in this case Δ_r. Therefore, the reference signal is generated by the following system (Figure 5), with an initial condition of zero, and independent noise processes,

$$\begin{aligned} x_r(k) &= \sin[\gamma], \quad \gamma = \mathrm{N}_1(0,\Delta_r) \\ y_r(k) &= \sin[\varphi], \quad \varphi = \mathrm{N}_2(0,\Delta_r) \\ \theta_r(k) &= \sin[\phi], \quad \phi = \mathrm{N}_3(0,\Delta_r) \end{aligned} . \tag{32}$$

The addition of unity feedback of the output to the random input tends to keep the system within the workspace (in fact, a double integrator with unity feedback in the Laplace domain, when converted back into the time domain is a sinusoid, so the random noise is filtered through a sinusoid, thus the tendency for the reference to remain bounded). This generates a non-repeating nearly oscillatory motion that has a fairly wide bandwidth of frequency injection, but is smooth enough to track.

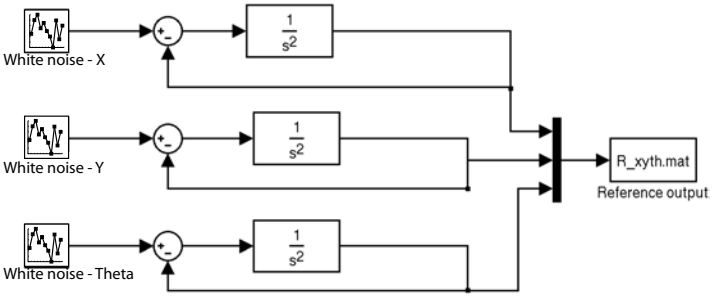


Figure 5: Reference generator for randomized reference. Each white noise consists of a random number generator. This is fed through a double integrator with unity feedback, then output. The result is a fairly smooth but rich reference signal.

The second reference goes through a variety of motions – translation of the object without rotation, rotation without translation, combinations, step inputs, sinusoids, and ramps. Both sets of trajectories (shown in the results section) are designed to cover the workspace of interest.

2.3.3 Experiment 3: Stability and performance tests

These experiments consist of subjecting the algorithm to random disturbances during trajectory tracking, a frequency response analysis, step and impulse response tests. Though performance requirements are somewhat arbitrary, we desire performance capabilities similar to humans, which means a bandwidth of 30Hz or less, and a moderate amount of damping.

The step and impulse response tests consist of applying a step, in one instance, and a single time-step impulse in the other, of approximately 1% and 100% amplitude, respectively, of the object workspace at a time of 250msec (giving the manipulators time to grasp the object prior to the reference change). The steps and impulse test applies the pulse to all dimensions. This is performed in open and closed loop for comparison.

2.3.4 Experiment 4: Flexible cooperation - Manipulation with 2 to 10 manipulators

This experiment repeats the tracking experiments with differing numbers of manipulators. The object reference tracking capability and force output of each manipulator are evaluated.

3 Results and Analysis

Animations of the experimental results are available on the author's website in the papers section[17].

3.1 Experiment 1 results

It is clear from Figure 6 that the HOC is capable of grasping an object given a random initial condition. For all tests, the grasps were successful and achieved steady state at nearly the same time. Due to the fact that for these tests the high level control is an LQR, and that the manipulator forces sum to very nearly the ideal forces computed at the high level, typical regulator behavior for this type of feedback controller is displayed. Though it would be possible to increase gain or damping, the LQR control was designed to balance performance and stability with similar capabilities to human levels. The controlled object settles to the reference trajectory within 5-600 msec, with acceptable overshoot.

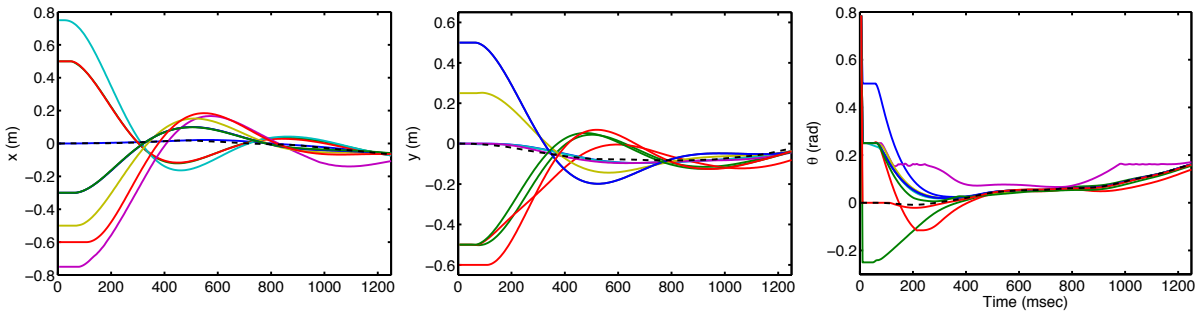


Figure 6: Experiment 1 results. The manipulators (3 finger configuration) grasp the object and cause the object to converge quickly to the reference trajectory for a variety of initializations of object and manipulator positions.

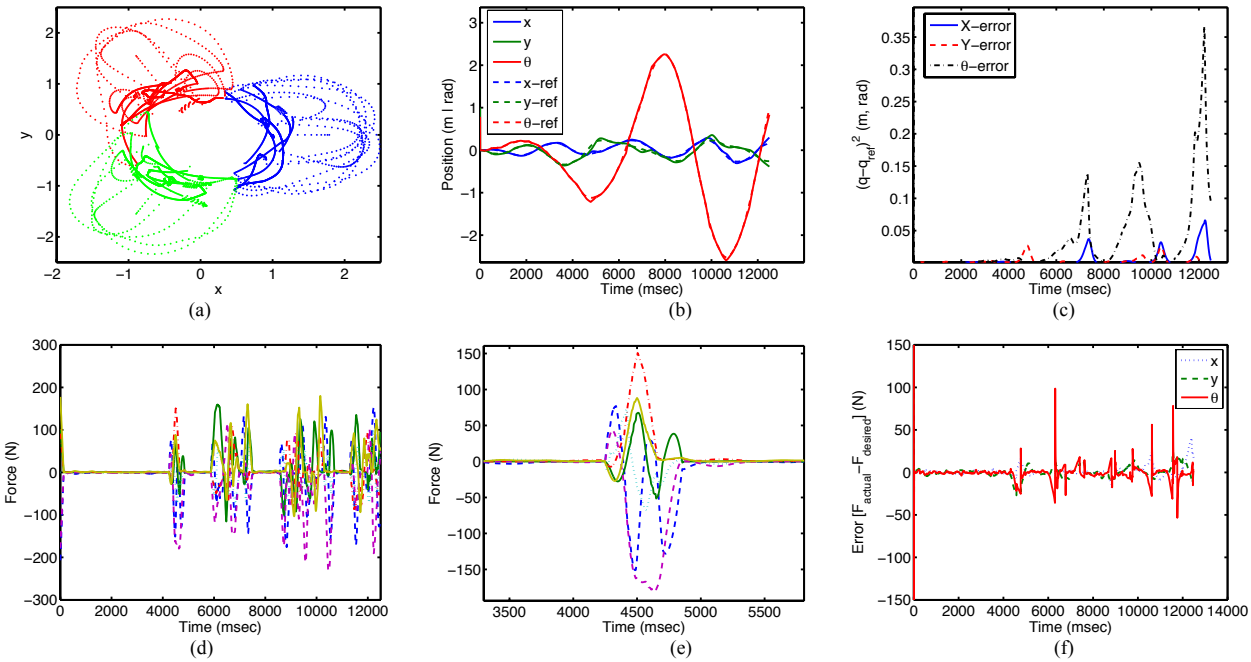


Figure 7: Experiment 2 results. (a, b) We see the algorithm can track with random acceleration of the reference. (c) The square of the tracking error for position and orientation. The velocity increases as the experiment progresses, and the algorithm can still track, even with a moderate gain. (d) Forces applied at the endpoint of the manipulator (to the virtual point mass), note large forces occur when contact is broken and the manipulator is seeking a new contact point. (e) A zoomed-in view of a contact break. (f) Error in actual applied force versus the ideal force computed by the high level. There are spikes during contact breaks.

3.2 Experiment 2 results

We see in Figure 7a, 8a that the manipulators move through a series of motions and perform a series of contact breaks and repositioning of the manipulators on the object surface. Figure 7b, 8b shows the reference trajectory and the actual object trajectory. That figure shows the closeness of the trajectory tracking, even with the random accelerations to track. Figure 7c shows the squared error between the trajectory and reference. In general the tracking is excellent, near the end of the ever increasing velocity there are more and more points where contact breaking must occur and large accelerations at those time instants yield tracking errors, but still acceptable ones. Figure 7d,8d shows the forces applied at the tips of each manipulator in x and y. The individual forces are not the key here, but rather a few points of interest, which will now be described. The forces when a manipulator is in contact are much smaller than when the manipulator is seeking a new contact point and is freely moving through space, as the algorithm is attempting to get to the new contact point as quickly as possible. Therefore the early part of the trajectory when no contact breaking occurs shows all forces quite small (below 5N).

Figure 7e shows a close-up of these endpoint forces just before, during, and after a contact break. When a manipulator breaks contact, those still in contact must compensate, and there is a dynamic force fluctuation during that process. This is why we see spikes in force errors (Figure 7f, 8f) – those are contact break-and-make forces. In a paradigm with soft contacts these forces will be damped significantly, but will likely still be present. These spikes cause larger spikes in torque tracking errors since both x and y components contribute to torque, and spikes typically are represented in both those axes.

It may be a significant point that there are some manipulation paradigms where keeping these contact spike amplitudes small would be significant, and this could be directly incorporated in the force field function, adding a term to dampen the trajectory just before contact, similar to modern hard disk control algorithms, which slow down the read-write head during a large seek operation just before reaching the goal location in order to reduce acoustic noise from the hard disk.

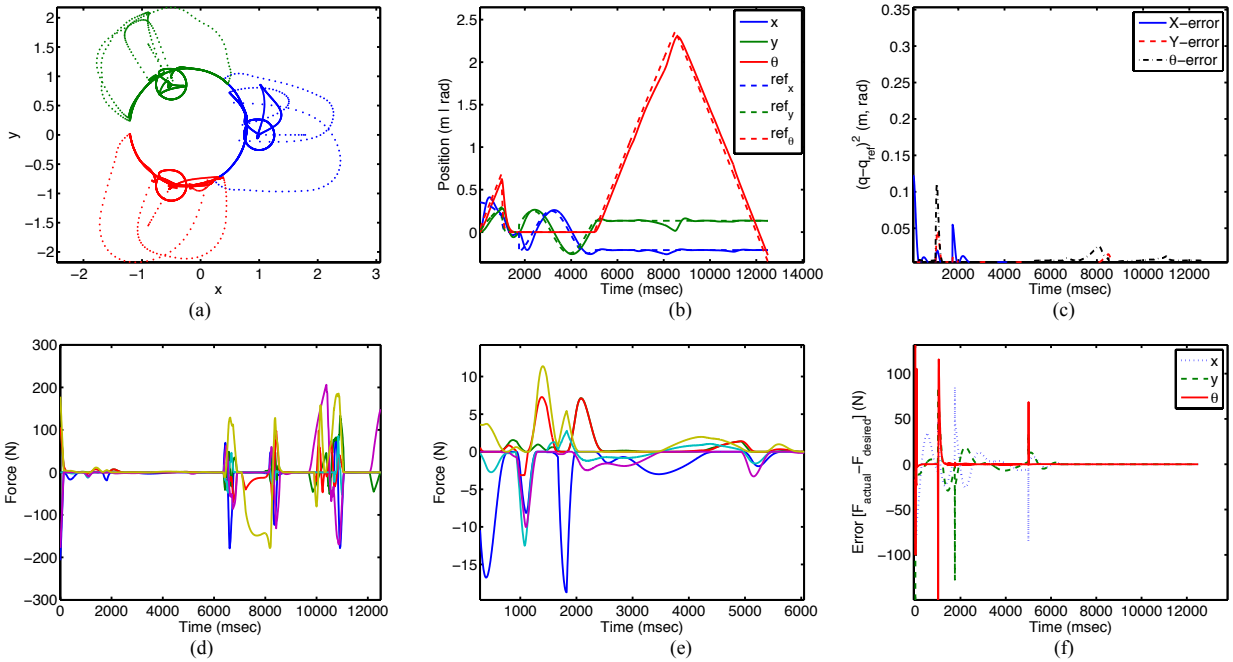


Figure 8: Experiment 2 results. (a, b) Here it is clear that with steps, sinusoids, ramps, and more the HOC can still track successfully. Since the acceleration of the reference is smooth we see in general the tracking error (c) is smaller. (d, e) The forces are also more smooth, and the force tracking error (f) is quite small unless there is a step or impulse change, as happens early in the first few seconds of reference.

However, when the manipulators have built-in physical damping such as rubber or foam tips, it is not clear whether this need be included at the algorithmic level or not.

It is clear from Figure 8 that the HOC can successfully track during step, sinusoids, and ramp reference shapes. Since this reference has mostly smooth accelerations as can be seen in Figure 8e (not the random sine-filtered acceleration of the reference for Figure 7, in general this trajectory, forces are smoother and errors smaller. Large forces and errors are experienced during step changes in reference, which is to be expected. In all references created, the HOC maintained stability and sufficient tracking performance.

3.3 Experiment 3 results

The eigenvalues of the open loop system are

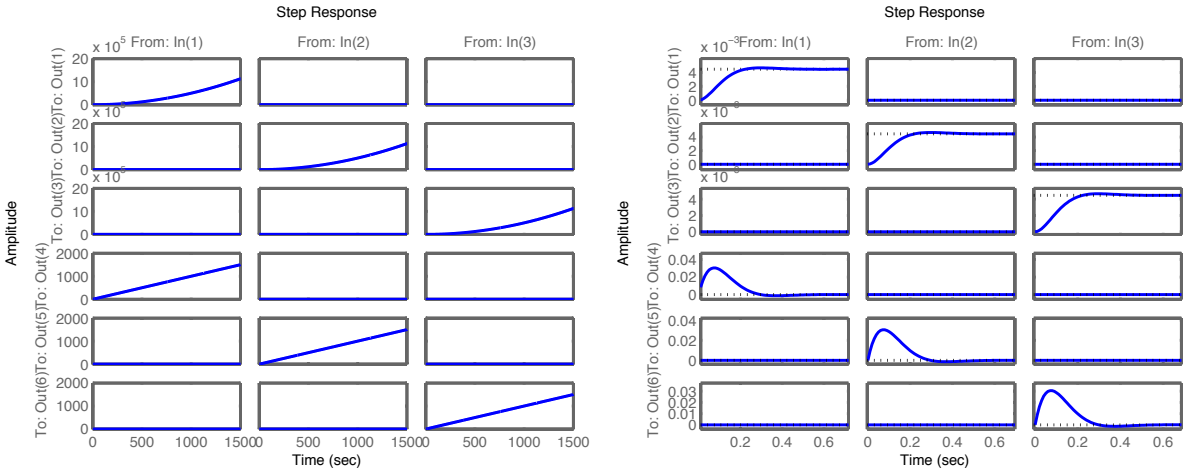
$$\lambda_o = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T, \tag{33}$$

which is highly undesirable for a control system. The eigenvalues of the closed loop system on the other hand, computed by the command LQR in matlab with the matrices as specified in [REFERENCE ABOVE EQUATIONS] and with the Q and R matrices found by loop-shaping to be

$$Q = \begin{bmatrix} 500 & & & & & 0 \\ & 500 & & & & \\ & & 500 & & & \\ & & & 0 & & \\ & & & & 0 & \\ 0 & & & & & 0 \end{bmatrix}, \quad R = \begin{bmatrix} 0.01 & & & & & 0 \\ & 0.01 & & & & \\ & & 0.01 & & & \\ & & & 0.01 & & \\ & & & & 0.01 & \\ 0 & & & & & 0.01 \end{bmatrix}, \tag{34}$$

are given by

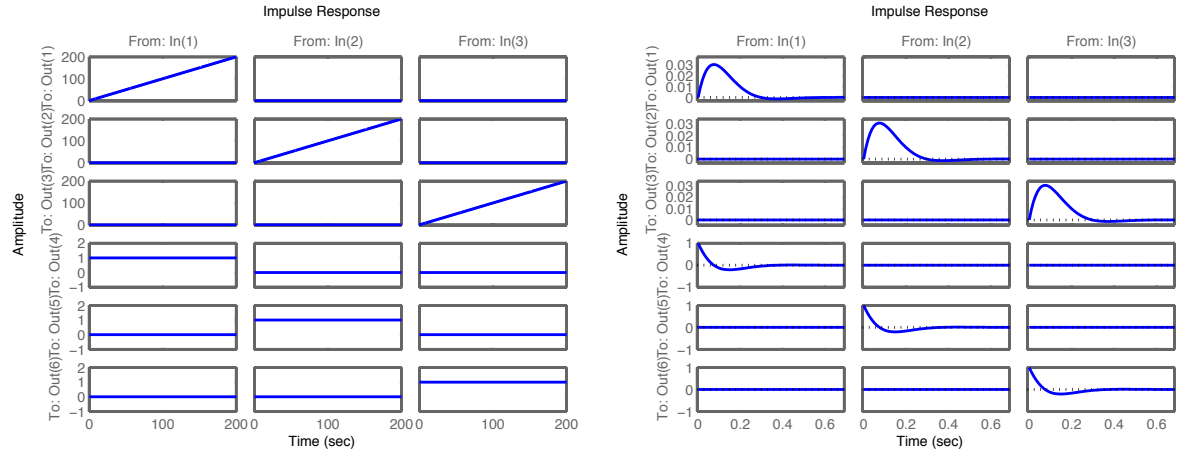
$$\lambda_c = \begin{bmatrix} -10.5737 + 10.5737i \\ -10.5737 + 10.5737i \\ -10.5737 + 10.5737i \\ -10.5737 - 10.5737i \\ -10.5737 + 10.5737i \\ -10.5737 - 10.5737i \end{bmatrix}. \tag{35}$$



(a) (b)

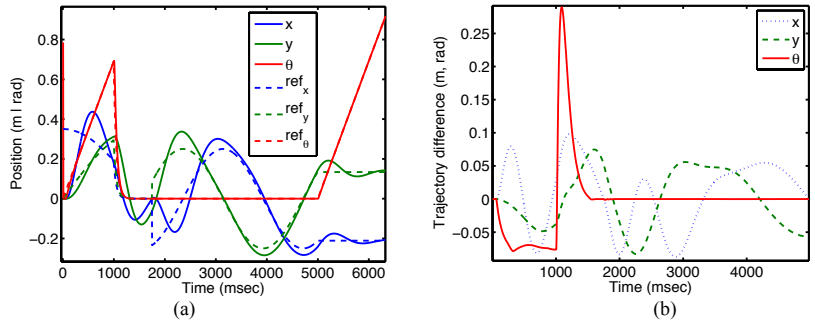
Figure 9: Here we see that the closed loop system is stable to step disturbances, settling in under 0.2 sec, while the open loop system is clearly not, diverging toward infinity. The matrix of plots display the input to output, as labeled (so, for example, if we are interested in determining the effect of input one on output one, we look at the [1,1] location in the matrix). Position and velocity are shown since the states include both position and velocity.

The closed loop eigenvalues are all negative in their real part (they are in the left half of the complex plane) and so the closed loop high level system is stable (see Figure 9 and 10). In fact, if one disables the manipulator portion, and applies computed high level ideal forces to the tracking problem of the object, indeed the object can be made to track, as seen in Figure 11. One may also note that, Figure 11, which is the difference between the manipulated and high-level trajectories, shows highly similar behaviors (the manipulators are indeed capable of generating the behaviors we would like to perform by ideal forces).



(a) (b)

Figure 10: Impulse response plots for the open and closed loop system. Again it is clear that the open loop system is not stable, while the closed loop system is stable.



(a) (b)

Figure 11: Ideal force control only. (a) Here we see that for the step-sinusoid-ramp trajectory the algorithm can track. (b) Shows the difference between the ideal force-driven trajectory and that created by using manipulators. It is clear that the difference is very small.

3.4 Experiment 4 results

The HOC functioned correctly regardless of the number of manipulators employed. These ranged from 2-10. Figure 12 displays trajectories for a variety of configurations. It is clear that similar patterns emerge since the object is moved through nearly the same trajectory. The reader's attention is now drawn to Figure 13c where we see that with more manipulators contributing to the object movement, each individual manipulator's force contribution drops. A second important point to note is that there appears to be an optimal number of manipulators in terms of accuracy for a particular trajectory and LQR control. With more manipulators recruited for a movement, trajectory tracking is slightly less accurate (Figure 13a,b).

There are a number of potential explanations for this phenomenon, however the most plausible is that, assuming during a particular motion the majority of manipulators are in contact, all fingers are contributing to the object grasp, as well as increasing the overall system mass being moved. With a different mass the optimal high level control can be recomputed in order to optimally track. In fact, in a model predictive setting and a learning setting this issue should disappear, since the optimal control is recomputed at each time step. Though ideally each manipulator would compensate for its added inertia to the system during contact, in reality each manipulator has limited bandwidth, and so during rapid accelerations those limits become evident. This demonstrates the limitation of force feedback to compensate for friction and inertia in a system – bandwidth is never infinite. However as long as the system need not move more rapidly than its capability in order to compensate, this is still a useful, if imperfect approach. That is why it is more challenging to dynamically manipulate objects which are very light relative to the manipulators – the object dynamics can be very fast relative to the capabilities of the manipulators, and thus easily 'fly' out of the controlled workspace during light grasps and contact changes.

A somewhat expected result is that, though manipulation is possible with two manipulators, it is not as optimal during contact breaking type complex movements. This can be explained by the limitations of the manipulator force capabilities and the fact that with two fingers there is a lack of redundancy: the moment one finger breaks contact the object is underactuated unless the second finger happens to be in an optimal location. In the case of two manipulators, a model predictive high level control would produce optimal results, as optimal grasp locations can be anticipated before the contact breaks occur.

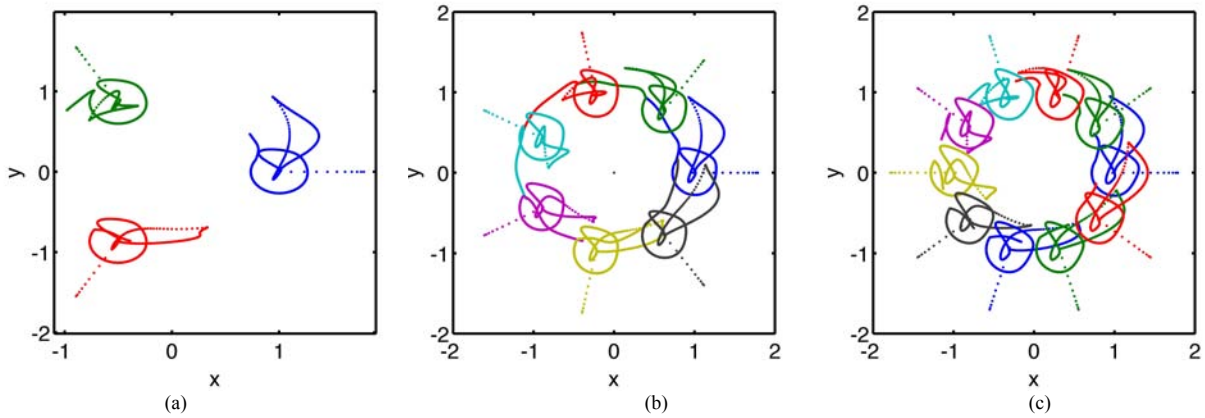
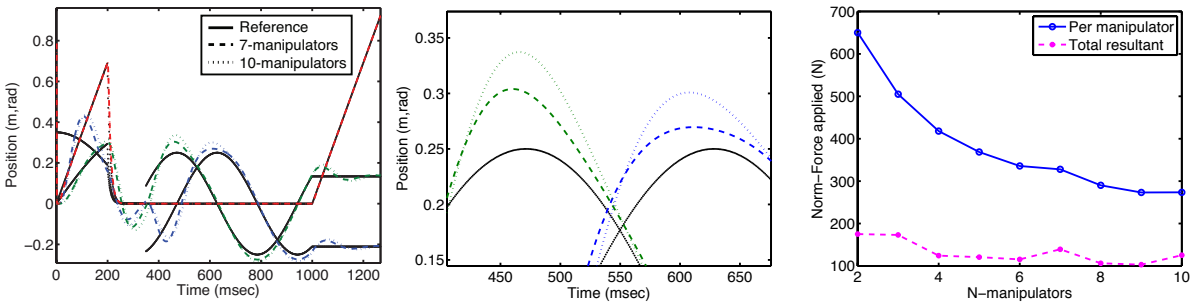


Figure 12: Experiment 5 results. A shorter portion of the trajectory is plotted here, for clarity. The manipulation is successful for three, seven, and ten manipulators.



(a) (b) (c)

Figure 13: (a) Trajectory tracking with differing numbers of manipulators. Note that with more manipulators there is a tendency to overshoot slightly more during rapid accelerations. (b) The right figure shows the same data zoomed in to highlight the overshoot. Each color is a different axis (red- θ , blue–x, green-y). (c) Individual manipulator force and resultant forces (over all manipulators) versus number of manipulators during 6 seconds of the same reference. The total force applied remains approximately constant (decreases slightly) as number of manipulators change, however the individual force is inversely proportional to the number of manipulators.

4 Conclusions

In this paper the author presents an approach to control of robotic systems that can potentially fill the need for a more adaptable, intelligent, and broad-spectrum approach to control of high dimensional, potentially uncertain, and cooperative robotic systems than previous methods have addressed. This is due to its modularity, where components of the algorithm can be replaced at any level, even on the fly during control actions. It is this modularity that will allow the breadth of behaviors necessary to interact with unstructured and uncertain environments, and varied or poorly defined tasks.

The experiments performed with the HOC algorithm demonstrate that this is an effective means of rapidly computing solutions in real-time for the challenges of controlling objects with external manipulators. The manipulators are capable of meeting performance and stability requirements, and adapting to changing numbers of manipulators and parameters. There is very little difference between the desired forces and those applied by the manipulators for all the variety of references generated.

There are several components to explore and add to this work for the future. First it will be extended to three dimensions, more complex contact algorithms, and learning (which will employ passive Kalman filter-type learning and active learning such as in [20]) in order to reduce uncertainty at various levels such as the manipulator level. Essentially manipulators are most useful when mobile, and so the HOC will be expanded to include a robotic arm that can move the manipulators and thus alter the workspace center. The algorithm will be tested by performing complex tasks with unstructured environments such as assembling a machine with varied shapes of components with little instruction and large or changing uncertainty. It will be applied to a variety of mechatronic hardware that as of the time of this writing is just completing testing phases. Force field functions built directly from human data, essentially ‘teaching’ the algorithm to perform certain tasks specifically, will be explored. Though a general function that performs well in a variety of tasks works, there are times where having a function or trajectory specifically optimized for a task makes sense to employ. This is where the modularity of the HOC is a tremendous advantage.

In the long term it is algorithms and methods such as this that will help the world to address issues such as recovery from brain injury, developing effective artificial limbs, improved intelligent automation in manufacturing and emergency situations, and exploration and navigation of hostile environments. It is the author’s hope that this work presents a step in the direction of an answer.

Acknowledgement

This work is supported by the U.S. National Science Foundation.

References

[1] S. Boyd and L. Vandenberghe. Convex Optimization, volume 25. Cambridge University Press, 2010.

[2] M. de Lasa, I. Mordatch, and A. Hertzmann. Feature-based locomotion controllers. ACM Transactions on Graphics (Proc. SIGGRAPH), 29(3), 2010.

[3] J. Ferziger. Numerical Methods for Engineering Application. John Wiley and Sons, New York, NY, 2nd edition, 1998.

[4] N. Giordano, N. Hisao (July 2005). Computational Physics (2nd edition). Benjamin Cummings. ISBN 0-13-146990-8.

[5] Ernst Hairer, Christian Lubich, and Gerhard Wanner. Geometric numerical integration illustrated by the stoïrmer/verlet method. Acta Numerica, 12:399–450, 2003.

[6] S. Jagannathan and G. Galan. Adaptive critic neural network–based object grasping controller using a three-fingered gripper. IEEE Trans. on Neural Networks, 15(2):395–407, March 2004.

[7] R.Kalman. A new approach to linear filtering and prediction problems. Transactions of the ASME–Journal of Basic Engineering, 82(Series D):35–45, 1960.

[8] Y. Lee, S. Kim, and J. Lee. Data-driven biped control. ACM Transactions on Graphics (Proc. SIGGRAPH), 29(4), 2010.

[9] W. Li and E. Todorov. 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, August 2004.

[10] C. K. Liu. Dextrous manipulation from a grasping pose. ACM Transactions on Graphics (Proc. SIGGRAPH), 28(3), 2009.

[11] L. Ljung. System Identification: Theory for the User. Prentice Hall, PTR, Upper Saddle River, NJ, 2nd edition, 1999.

[12] A. Moon and M. Farsi. Grasp quality measures in the control of dextrous robot hands. IEEE Colloquium on Physical Modeling as a Basis for Control, pages 6/1–6/4, Feb. 1996.

[13] I. Mordatch, M. deLasa, and A. Hertzmann. Robust physics-based locomotion using low-dimensional planning. ACM Transactions on Graphics (Proc. SIGGRAPH), 29(3), 2010.

[14] L. Piegl and W. Tiller. The NURBS book. Springer-Verlag, London, UK, 1995.

[15] M. Santello, M. Flanders, and J. Soechting. Patterns of hand motion during grasping and the influence of sensory guidance. The Journal of Neuroscience, 22(4):1425–1435, February 2002.

[16] A. Simpkins. Exploratory studies of human sensorimotor learning with system identification and stochastic optimal control. PhD thesis, University of California at San Diego, La Jolla, CA, 2009.

[17] A. Simpkins. http://casimpkinsjr.radiantdolphinpress.com, 2012.

[18] Alex Simpkins (2012). Real-Time Control in Robotic Systems, Robotic Systems - Applications, Control and Programming, Ashish Dutta (Ed.), ISBN: 978-953-307-941-7, InTech, Available from: http://www.intechopen.com/books/robotic-systems-applications-control-and-programming/realtime-control-in-robotic-systems

[19] A. Simpkins and E. Todorov. Optimal tradeoff between exploration and exploitation. American Control Conference, IEEE Computer Society, 2008.

[20] A. Simpkins and E. Todorov. Complex object manipulation with hierarchical optimal control. Paris, France, March 2011. IEEE Symposium of Adaptive Dynamic Programming and Reinforcement Learning, IEEE Computer Society.

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

[22] H. W. Sorenson, editor. Kalman Filtering: Theory and Application. IEEE Press, 1985.

[23] R. Stengel. Stochastic Optimal Control: Theory and Application. John Wiley and Sons, 1986.

[24] J Michael Vandeweghe, M.Rogers, M.Weissert, and Yoky Matsuoka. The act hand: Design of the skeletal structure. Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA '04), May 2004.

[25] J. Wu and Z. Popović. Terrain-adaptive bipedal locomotion control. ACM Transactions on Graphics (Proc. SIGGRAPH), 29(4):72:1– 72:10, 2010.