Complex object manipulation with hierarchical optimal control

Alex Simpkins† and Emanuel Todorov‡

***Abstract*— This paper develops a hierarchical model predictive optimal control solution to the complex and interesting problem of object manipulation. Controlling an object through external manipulators is challenging, involving nonlinearities, redundancy, high dimensionality, contact breaking, underactuation, and more. Manipulation can be framed as essentially the same problem as locomotion (with slightly different parameters). Significant progress has recently been made on the locomotion problem. We develop a methodology to address the challenges of manipulation, extending the most current solutions to locomotion and solving the problem fast enough to run in a realtime implementation. We accomplish this by breaking up the single difficult problem into smaller more tractable problems. Results are presented supporting this method.**

***Index Terms*— Optimal control, object manipulation, legged locomotion, hierarchical control, optimization, adaptive control, nonlinear systems**

I. Introduction

Manipulating objects is a complicated problem (Fig. 1). As you read this paper you are either controlling your mouse or flipping through paper, performing control in the context of uncertainty, contacts (changing dynamics), nonlinearities, redundancy, and high dimensionality. Both manipulation and locomotion are difficult, related problems which have not yet been completely solved in the control sense. We present extensions to current methods and demonstrate application to simulated robots performing manipulation tasks. Additionally, we address a methodology for making difficult problems more parsimonious by breaking the problem into pieces.

Related work, and the connection between locomotion and manipulation: Among the most relevant work that has been done is the work in [8] who has assumed a grip is given and contacts are static, and then determines the forces to move an object in a prescribed manner. This is akin to solving part of our problem. Though this is groundbreaking work, it is limited for manipulation problems where the object may need to be in continuous rotation. In that case, the grip must be constantly changed, and how the grip is changed is a dynamic problem coupled to the low level problem of when, where, and how to move each individual manipulator. Many studies have been performed about how humans grip objects[13], but this assumes a somewhat quasi-static notion of holding an object. In robotics, work has been done on determining forces to constrain an object with a gripper[11] and gripper reference positions[5], again in a static equilibrium sense, and only with a particular number of manipulators (usually a 1-degree-of-freedom gripper). Here we do not limit ourselves to quasi-static situations, but rather determine a solution approach which deals with static and dynamic interactions, and even changing numbers of manipulators.

This work was supported by the US National Science Foundation

†Department of Computer Science and Engineering, University of Washington, Seattle 185 Stevens Way, Seattle WA 98195-2350 email: csimpkin@cs.washington.edu

‡Department of Computer Science and Engineering, University of Washington, Seattle, 185 Stevens Way, Seattle, WA 98195-2350, email: todorov@cs.washington.edu

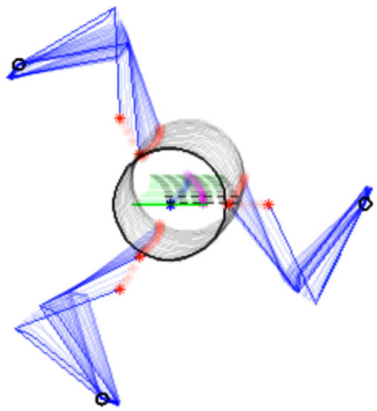

Fig. 1. Overview of the two dimensional problem. How does one apply forces to an object to (a) not allow it to drop, and (b) cause it to follow a prescribed trajectory?

Significant work has been done in control toward improving the dynamics of gripper mechanisms themselves, such as [3], but this work does not solve the entire problem as one dynamic system, including the object and gripper dynamics. It assumes open loop trajectories to follow in order to achieve a given grip, dealing only with what is the lowest level of the present control scheme. Additionally, there is evidence that biological systems intelligently use the passive dynamics built-into their bodies to aid in control rather than cancel their dynamics[16][9]. The present work attempts to draw on the intelligent approaches used by biology.

Prior work has addressed the problem of locomotion, but we can consider that manipulation of an object is essentially functionally identical to locomotion with only relativistic differences. In object manipulation, the manipulator is typically larger than the manipulated object (or on a similar scale), and so the object is moved and controlled relative to the manipulator origin. But in locomotion, the object being manipulated is the ground, which is much larger than the manipulator, so the origin of the manipulator moves relative to the manipulated object. In both these cases there is a relative movement between the object and manipulator origin, and the goal is to control that movement.

If this assumption holds for the purposes of our model, similar approaches to locomotion can be applied to manipulation. Unfortunately, locomotion is still largely an unsolved problem in the sense that most solutions require a great deal of insight and manual 'tweaking' by the experimenter to create a successful locomotion algorithm. In addition, most locomotion approaches can perform only a very limited number of behaviors (walk only over smooth ground, run only, etc.). However, the approach that is currently quite promising is a hierarchical control approach[7][10], combined with model predictive control. Indeed, the most successful exam-

ples can be found in computer graphics where people are interested in creating interactive characters which function based upon physics rather than motion capture or kinematic data[18][12][2][6]. This approach has the potential to create very adaptable interactive characters which have unscripted and rich behaviors. Such work has not yet been implemented in real physical robotic systems (see [1] for an overview), and this is currently in process in our lab.

On redundancy and underactuation: A parallel between walking and manipulation is that in both cases the end effector is underactuated - i.e. one does not have direct control over the object in question (either the ground relative to the agent or the small manipulated object relative to the fingers). This is another reason that similar techniques should be applied to both problem classes.

Contributions: This work contributes a new dynamic hierarchical optimal control approach to the manipulation problem, which can also be applied to the locomotion problem. It also requires a minimum of manual tweaking, can be tuned through a few intuitive parameters, and can be expanded in terms of behavioral complexities easily. This is also a realtime implementation of this solution, requiring no offline computation to adapt or pre-learn trajectories. These methods are also designed to be implemented on real physical robots, beginning with those developed in our lab specifically for manipulation and locomotion. Another contribution is the concept of treating manipulation and locomotion as essentially the same problem with slightly different parameters. Finally, the method of handling joint limits by a useable workspace with a soft boundary that acts as a spring when the manipulator is drawn past the workspace facilitates numerical stability, and parallels biological systems in a way not yet implemented to our knowledge.

Paper outline: The rest of the paper is organized as follows: Section II describes the system model and hierarchical control approach to the problem, Section III describes the experiments used to test the algorithm, Section IV describes the results of the experiments and discusses the implications, and finally Section V presents a number of conclusions to the paper and discusses the next steps to be taken.

II. Model

Objects will be modeled in two dimensions (2D) in this paper, though all techniques developed here extend to three dimensions directly (and this is the topic of an upcoming paper). An object is represented by a center of mass, mass, inertia, and an external boundary. This allows for objects of any 2D shape to be represented. Locations for contact points on the object are defined by angles relative to the origin of the object (a coordinate system located at its center of mass). The boundary of the object is represented in the control structure as a functional of the angle relative to the origin (this is mathematically stored as a cubic spline representation in order to facilitate the concept of the controller 'learning' about various objects it manipulates).

The system includes a given number (which may change at any time) of 2D manipulators (See Fig. 2) which can move in the plane of the object(s) and apply forces. These are 2D representations of manipulators which are in development in our laboratory and are currently in their prototype stages.

The manipulators are represented mathematically as a four bar linkage where each bar has mass, inertia, and a center of mass. The end effector extends from the second link in the loop, and torques are applied at the first joint, with effects upon the first link and the fourth link. The nature of the link structure creates a bounded workspace, which is represented as a function of the lengths of each link and the range of motion of joint one and four.

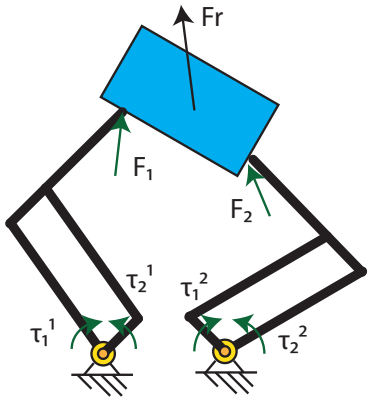

Fig. 2. Each prismatic manipulator has two degrees of freedom, and measures position of each joint and output force.

As with the physical robot, the rotor inertia in the DC motors (as coupled through the drive system) dominates the inertia in the system, so in general the system behaves like a point mass at the endpoint. By treating the complex trajectory part of the control problem at this level (i.e. controlling a point mass), optimal trajectories can be found quite rapidly, and output forces can be generated at a lower level by local feedback. This facilitates a realtime implementation. Complex dynamics can also be addressed by the use of parallel processing such as with GPU's, and implementation (at least partially) in C versus Matlab where loops are performed.

A. n-finger 2-D manipulation problem

It can be shown that the sum of forces on the object with n manipulators applying forces are given by the sum of forces and moments as,

$$\sum F_{x,o} = \sum_i f_{x_i} - m_o a_{x,o} \tag{1}$$

$$\sum F_{y,o} = \sum_i f_{y_i} - m_o g - m_o a_{y,o}$$

$$\sum M_o = \sum_i \big(- f_x d_y(\theta) + f_y d_x(\theta) \big)_i - J\ddot{\theta}_o$$

with f_i, $d(\theta)$, J, and θ representing, respectively, the force applied by manipulator i, the surface location relative to the object center as a function of angle, the object rotational inertia, and the object angle relative to horizontal, with $()_o$ representing 'with respect to the manipulated object of interest.'

An important constraint in contact is the following (unless slip occurs or the objects are moving apart, which is dealt with in the contact algorithm), with a representing acceleration of either the object a_o or point mass i, a_i, and $\ddot{\theta}_o$ the angular acceleration of the object:

$$a_{x,i} = a_{x,o} + \ddot{\theta}_o d_x(\theta_i) \tag{2}$$

$$a_{y,i} = a_{y,o} + \ddot{\theta}_o d_y(\theta_i)$$

This states that the point masses must move in such a way when in contact that they do not slip. The velocities

must also match during contact, and thus a discontinuous change (unless a soft contact algorithm is implemented) in velocity is experienced by the point masses and object. This can be modeled as an impulsive force which exactly brings about this change in velocity. The point masses are assumed to be insignificantly small compared to the object mass, though this assumption (could be dropped with some minor adjustments to the equations).

When not in contact, the accelerations of the point masses are given by, with external forces due to the robot actuators b_i:

$$a_{x,i} = \frac{1}{m_i} b_{x,i} \tag{3}$$
$$a_{y,i} = \frac{1}{m_i} b_{y,i} - g$$

The object dynamics are given from Equation (1) after rearranging by (note in the absence of contact the object is only affected by gravity):

$$\sum_i f_{x,i} - m_o a_{x,o} = 0 \tag{4}$$
$$\sum_i f_{y,i} - m_o a_{y,o} = m_o g$$
$$-\sum_i f_{x,i} d_y(\theta_{i,p}) + \sum_i f_{y,i} d_x(\theta_{i,p}) - J\ddot{\theta}_o = 0$$

The end effector dynamics when in contact with the object are given similarly by, after substituting in Equation (2):

$$f_{x,i} + m_i a_{x,o} + m_i \ddot{\theta}_o d_x(\theta_i) = b_{x,i} \tag{5}$$
$$f_{y,i} + m_i a_{y,o} + m_i \ddot{\theta}_o d_y(\theta_i) = b_{y,i} + m_i g$$

Which can be rearranged more succinctly by combining the force, mass/inertial, and external force variables and constants as,

$$f - \mathbf{m}a = b \tag{6}$$

or in matrix form, where bold face denotes appropriately sized sub-matrices, and I represents the identity matrix,

$$A = \begin{bmatrix} 1 & 0 & -m_o & 0 & 0 \\ 0 & 1 & 0 & -m_o & 0 \\ -d_{y,i} & d_{x,i} & 0 & 0 & -J \\ I & \mathbf{0} & m_i & 0 & m_i \ddot{\theta}_o d_x(\theta_i) \\ \mathbf{0} & I & 0 & m_i & m_i \ddot{\theta}_o d_y(\theta_i) \end{bmatrix} \tag{7}$$

$$w = \begin{bmatrix} f_{x,i} & f_{y,i} & a_{x,o} & a_{y,o} & \ddot{\theta}_o \end{bmatrix}^T \tag{8}$$

$$b = \begin{bmatrix} 0 & g & 0 & b_{x,i} & b_{y,i} + m_i g \end{bmatrix}^T \tag{9}$$

$$Aw = b \tag{10}$$
$$w = A^+ b \tag{11}$$

where (x^+) representing the pseudoinverse of x. The result, w, contains the forces the point masses apply on the object (and corresponding opposing forces by Newton's law of equal and opposite reactions), as well as the resulting acceleration of the main object. The size of the A matrix changes 'on the fly' as point masses come in contact and out of contact. In this way one can solve for the set of contact forces and acceleration. If there are multiple solutions, this determines a local minimum.

We now have our problem: when in contact - compute the individual forces and locations which optimally track the reference position or force, when not in contact, compute the required individual force trajectories to move the fingers along a trajectory. The two states of contact or non-contact trajectories form an object manipulation 'gait.'

In order to maintain contact with the surface, an orthogonal (to the surface) friction force must be maintained, unless the forces on the object are in static equilibrium. Let us consider the friction case. Essentially, one must maintain a friction force which balances the forces orthogonal to the contact point. Whatever the desired forces computed by the above equations from the reference trajectories of the object, the friction force required poses a constraint on the individual manipulator controls. In this paper, we assume the simplest form of friction, which is that of complete 'stickiness' of the object. In other words, any force into the object normal to the object surface causes the manipulator not to slip in the orthogonal direction to the surface at the point of contact. This does not avoid the issue of dropping the object, and so the intention is to simplify the system dynamics without sacrificing addressing the important aspects of this problem.

The problem also becomes more interesting when one considers redundant manipulation[1]. In this case, a minimal intervention principle[16] can be used to generate appropriate control signals. This works by one of two approaches, depending on how one is interested in solving the problem. If we don't want the (already in contact) manipulators to adjust their 'grip' positions on the object to be optimal, then the problem is 'given a particular distribution of contact points, what is the set of minimal forces to apply which will produce the desired acceleration and maintain constraint satisfaction.' The second way to approach the problem is to consider the previous problem but add the following, 'additionally, what is the optimal configuration of manipulators to produce the minimal set of forces required to move the object with a desired acceleration.' With redundancy, this may be a problem that has multiple solutions, so the best solution will be considered to be the one which is the lowest dimensional. The trajectory to move an object becomes part of the optimization problem.

This can all be formulated within the optimal control framework[15], and solved either with a globally optimal method (approximate the value function or control surface over the region of interest) or with a locally optimal control method, such as iLQG[2]. Additionally, iLQG can be initialized by an approximation to the globally optimal solution.

The actual robots to which these algorithms will be applied (which are called ModBots, developed in our lab by the first author) have more than enough dynamic capability to be controlled as point masses or ideal force applicators (with a carefully developed local controller, a well-identified system, and inertia dominated by the rotors of the motors), so the end effectors are modeled as point masses. This simplifies the optimization problem which we will discuss below, and helps with making the algorithm applicable to realtime systems.

[1] In other words, more degrees of freedom in the system (inputs) than in the object being manipulated (outputs), or end effector. An example in biological systems is the human arm, which has seven degrees of freedom, while the output end effector, the hand, can be oriented in a six-axis space (position and orientation). Therefore more than one solution exists for many tasks.

[2] iLQG refers to the "Iterative Linear Quadratic Gaussian" method.

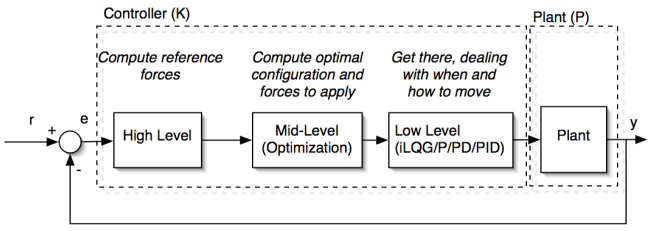

Fig. 3. Hierarchical control scheme block diagram.

B. Overall hierarchical control methodology

Though the entire problem can be framed within a single level stochastic optimal control, this is a high dimensional nonlinear problem, and to solve this in realtime, even approximately, is difficult. Additionally, there is evidence that the human brain solves this problem in a hierarchical fashion, even though the hierarchies are highly integrated[4], therefore why not use nature as a model? Instead of one high dimensional, difficult problem, we can reduce this problem to several smaller simpler ones which address each level of controlling an object in stages, and sum up to a solution to the more difficult problem. Solving the problem hierarchically, with multiple independent terms at the various stages allows for different strategies to be inserted at various points in the hierarchy without having to reformulate the entire problem. For example, one could decide that distributing manipulators evenly is key, and thus this stage would have different terms in the cost function, and in fact could be broken down into two sub-problems to solve for positions and forces.

There are essentially three components to this algorithm (Fig. 3). The top level computes, given a reference position and orientation (this can also be operated with force control), an ideal force and torque to be applied at the center of mass of the object. This can be done with a model predictive controller (using iLQG to allow for nonlinearities in the cost function and/or system), or a simple feedback controller (such as a proportional derivative controller). The middle level computes, depending on the actual position of each end effector, the target locations for each point mass and forces that should be applied to the point masses either to get them into position or to apply the force to the object. The low level maps those endpoint coordinate commands into feasible joint torques, and performs feedback if needed to make that happen. The problem can be succinctly stated as follows:

1) Determine overall force to apply on object (via optimal MPC, PD feedback, splines, etc)
2) Decide, given the number of manipulators (max.) where to place them according to an optimal criterion
3) Decide, given placements (and this is coupled to 2), what force output vector each finger will perform once in contact/in place
4) Move them there and apply the forces, avoiding any collisions

It may come to pass that individual end effectors must be shifted in terms of contact position. This does not need to be described as an independent step, as there are two strategies that can be employed to resolve this which work seamlessly as part of step three. The first and quickest to implement in a tuning sense is to use a force field which affects finger placement based on various system states, essentially an output map of the second strategy. The second strategy is to use an appropriately set-up optimal control problem with terms in the cost function for staying within bounds. The optimal approach will produce the correct movement and contact breaking behaviors, as this will simply be the optimal thing to do. This paper explores the force field methodology, and future work will exploit the knowledge gained to formulate the low level problem as an optimal control problem, solved with iLQG.

C. High level

For an in depth explanation of the iLQG technique, see [17]. Essentially what this method does is iteratively approximate a linear quadratic gaussian solution to the actual nonlinear cost function and nonlinear system about the current state. This method has a number of advantages discussed in the reference, but perhaps the most significant is its ability to construct a general solution approach to the varied problem structure which arises in manipulation. We applied this in the model predictive setting, running a time horizon of ten time-steps into the future.

Another simple high level controller which is not model predictive is a proportional-derivative (PD) controller, given as (in discrete form),

$$e_k = r_k - y_k \tag{12}$$
$$u_k^{PD} = K_p(e_k) + \frac{K_d}{\Delta T}(e_k - e_{k-1})$$

where ΔT is the time increment, k is the current time-step, K_d and K_p the differential and proportional gains, respectively, e is the error, y is the output state feedback, and r is the reference. One can implement any of the variations on classical control approaches (i.e. including an integral term), depending on the goal. Generally, removing the integral term creates more compliant manipulation, and biological systems do tend to have some error tolerance during normal behavior[16].

D. Mid-level

Now given the required force/torque to apply to the object, we need to determine where the best place is to apply the individual forces, given a certain number of manipulators such that they combine into the overall forces we want. We also need to determine what those individual forces should be.

In order to address this problem, we must consider the whole behavior and system. Recall that the goal is to track the high level reference, which may move through a particular manipulator's workspace and beyond, or may not enter a particular manipulator's workspace at all. So it must be determined what a manipulator should do if the object to manipulate goes outside of its workspace.

We build a method by considering behavior in humans. Hold an object in the air and rotate it (if it is breakable hold it over a pillow or other safety net please), using only your fingers. As each finger nears the boundary of its workspace, if you pay attention to your experience, you feel an increasing need, a desire to change the location of that finger to a more 'comfortable' one. It seems to take conscious effort to override this natural instinct. Essentially what we can distill from this in our case is that, as one nears a boundary, a way of conceptualizing this is that the human feels a type of virtual force, drawing their finger away from its current location. Then a trajectory evolves to a more central to the

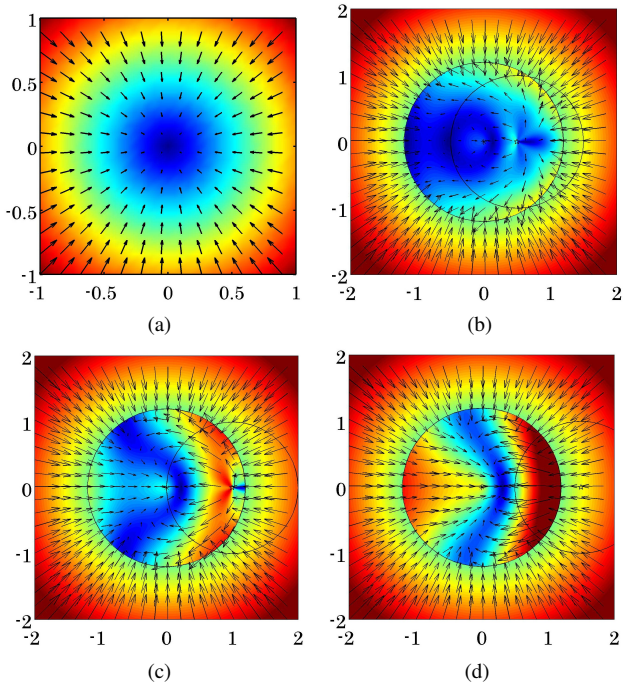

Fig. 4. A force field that facilitates contact breaking can be constructed which is dependent on several factors - center of workspace, center of object, and finger position. The field changes (as shown in (b), (c), and (d)) as the object enters the region of usable workspace. Here the workspace and object are circular for clarity, but any shape can be represented. The field is scalable and movable. The arrows depict the force vector at a particular point in space. The concept is that as the finger is manipulating an object, and moves too close to a workspace boundary, the field (which blends a position feedback controller with a number of other features) generates a force that causes a contact break and subsequent reposition of the finger more in the center of the workspace. Essentially this is an explicit representation of a cost function, though employed at the control policy level. The open circle in (b), (c), and (d) represents the object being manipulated, and the centered circle is the workspace definition. Outside the workspace a simple feedback pushes the finger back into the workspace, for stability purposes.

workspace point of contact. We call this a virtual force because it is used in a weighted comparison with the contact force at each iteration, and the one with the larger normal force component 'wins,' and is then used as the force applied to the end effector.

Workspace virtual force field *:* A method for distributing high level commands is the concept of a virtual force field (As depicted in Fig. 4(a)-4(d)) which is dependent on the shape of the workspace for each finger, the location of the end effector, and the location (or estimated location) of the object. This strategy is helpful for constructing terms which will ultimately be employed for a model predictive low level component. It is actually a parallel concept except that the model predictive approach will compute control policies which may achieve lower mean squared error for tracking desired object trajectories than a feedback-only scheme. However it is important to eventually compare a high level model predictive control (MPC) approach to a low level one, and finally to a scheme with all MPC methods.

This force is built by a number of terms in a function which we will call a 'Force-Cost' function (FC function), since the terms are applicable to either the optimal control setting or our force field setting. When referring to the computed output of the FC function, for simplicity, we will hereafter refer to the output as a 'cost' in either setting[3]. This function must include a variety of terms in order to address the required behaviors, the latter of which are:

1) Getting to target given some state of the system
2) Applying some force to object
3) Distribution of contact points
4) Desire to change contact points (break contact) when appropriate
5) Handle limited workspace
6) What to do if an object moves outside of the workspace, or the end effector moves outside the workspace

Many of these behaviors can be combined into single equations. When in contact, the contact forces are computed with a quadratic (and thus very fast) constrained (to only apply forces toward the object surface, or the force is zero) optimization that minimizes the error between the sum of forces (due to any number of manipulators in contact) and reference force. $d(\theta)$ is the shape of the object (parametric curve or surface - note in the case of a surface the parameter will be a 2D parameter), where θ is a parameter of the curve, and f_r represents the reference force and torque to apply at the next time-step,

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

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

Where the $\sum_{(x,y),i}$ is defined as 'over x, y, and all i.' A second term to minimize maximum forces can be added, similar to a control cost. At each time-step the optimization is initialized with the previous result, which tends to produce more smooth outputs. Thus the optimization problem is,

$$\Big(f_i^{opt} = \underset{f_i}{\operatorname{argmin}} \big[J(f_i)_k \big], s.t. \big[f \bullet n < 0, \big] \Big) \tag{13}$$

where n is the normal vector to the surface of the object, and $(\bullet)$ denotes the dot product, and f_i includes both x and y dimensions implicitly here.

The force field consists of the following terms, the first of which is, with gain K_e,

$$F_e = K_e \frac{(x - x_c)}{1 + ||x - x_w||^2} \tag{14}$$

which generates a force as the end effector location is further from the estimated or actual center of the target object to pull it toward the object surface, but this force decreases as the end effector moves further from the center of its workspace, until the next term dominates. The one prevents a vanishing denominator. This term works in concert with the second term to determine where contact will occur (Fig. 5(a)-5(d)). It is heavily affected by the location of the object relative to the workspace. The second term, (with gain K_s, surface normal n, and parameter a),

$$F_s = K_s n ||x - x_w|| e^{-a(x - x_c)^2} \tag{15}$$

[3]We state 'may' since the force field yields a predictive component, similar to an infinite horizon setting, since the resulting policy will act such that force over the entire behavior is minimized (recall we are considering force and cost parallel in our FC function), and so it is unclear wether one is superior, or wether they are functionally equivalent when provided with equivalent parameters

generates a force which tends to pull the end effector away from the surface of the object. This force is larger the further the end effector moves from the center of the workspace, but drops to zero as a gaussian function of distance from the center of the object. This is what is responsible for contact breaking. The forces are summed,

$$F_t = F_e + F_s \tag{16}$$

Which yields the field displayed in Fig. 4(b)-4(d). Note how the field adapts as the various system states change. Fig. 4(a) shows a simple force field concept - the further a point is from the center, the larger the force pulling toward the center of the field. More complex fields are possible. It is not difficult to determine empirical force fields from behavioral data, similar to inverse optimal control or using system identification methods to fit a function to reproduce biological data. Thus this method is extendable to many coordinated movement patterns, and parameters can be optimized for different styles of movement, or they can be learned in an active learning setting (such as[14]).

This is both a predictive and adaptive approach. It is predictive in that the endpoint is not computed, but the next point is a portion of an entire trajectory which will emerge if all the other states were to remain constant during the trajectory. Essentially the MPC is recomputing all possible trajectories at each time-step as the functions are continuous functions of the states, so the trajectory adapts as all states change.

In order to deal with workspace boundaries, a saturation effect is produced by switching the field to, where K_k is a gain,

$$F_t = K_k(x - x_w) \tag{17}$$

which tends to push the manipulator back into the workspace if it exits it. In other words this is similar to biological systems where a boundary is reached by the combination of tendons with the physiological structure creates a nonlinear spring effect (move your finger up as high as it can go by itself, that is the boundary of the workspace where you can control it. Using your other hand, you can move your finger quite a bit higher, but the effect is that of a spring - the finger pulls against your hand back toward the center of its workspace).

E. Low Level

The low level is responsible for mapping the commands for forces to apply at the endpoints into joint torques. There are several advantages to working in endpoint coordinates then mapping them into joint angles and torques. These include structural variability in the actuators used (solve the problem then reverse the joints and no change in the overall solution occurs, but the system still works just as well), simpler kinematic equations (one removes several trigonometric relationships, which are nonlinear relations from the optimizations), and more intuitive FC function terms.

The algorithm implements a bounded Newton's method optimization, initialized at the current state, to compute the nearest torques to apply which are within capabilities of the virtual actuators at the joints which map most closely to the desired end effector forces. This turns out to, in practice work well and execute with only a few iterations on average until convergence, even when a large change occurs, such as transitioning between contact and no contact.

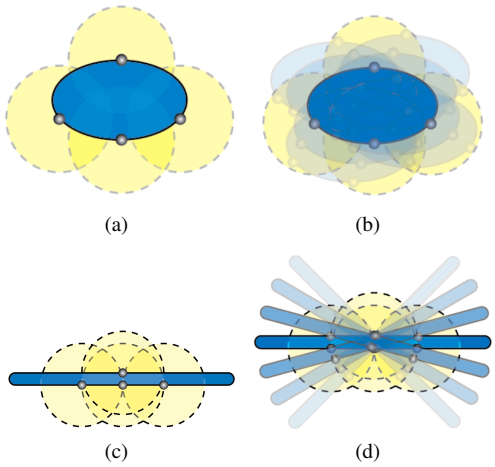

Fig. 5. Why workspace maximization is an important consideration when choosing where contact points occur. By choosing a point which puts each manipulator closest to the center of its workspace, the object workspace (how far in various directions the object can be moved without further changing contacts) is maximized, either for rotation or translation. If a series of commands into the future are predictable, this may skew where the best place for these contacts will be.

Finally, a simple PD controller tracks the required joint torques. We found that, in fact, this final component can be disabled and the algorithm functions well, but in practice it can improve robustness, and allow the higher levels to be operated at a lower bandwidth than without low level feedback.

When implemented on the physical robot, the low level will run on the local digital signal controller, and so the overall loop running at the high and middle levels will execute in less than the current time-step.

III. Experiments

Manipulate a known shape and mass/inertia with a specified number of end effectors

Consider an object of arbitrary shape with some mass m and inertia about the z-axis I. There are n manipulators which can have a (specified) limited range of motion and maximum force (or even a forcing function over the workspace, but for this experiment we consider a constant force capability over the workspace). These manipulators have bases located at x_b, and the workspace has a center relative to that base at x_w. All simulations are integrated using Symplectic Euler integration and a time-step of 5msec (Animations are available at http://casimpkinsjr.radiantdolphinpress.com).

Grab object, reference track, not specifically attempting contact breaking: This consists of grabbing the object with the end effectors, which are initialized at some random non-contact state away from the object. Then the object is, in the first sub-case, brought back to the zero position and must track a sinusoidal angular reference trajectory. In the second sub-case, the object is moved through a sinusoidal pattern in both position and orientation.

Grab and perform continuous rotation: Rotate the object at a continuous velocity. This will force the fingers to change grip, while maintaining appropriate forces to keep the object center at a fixed point.

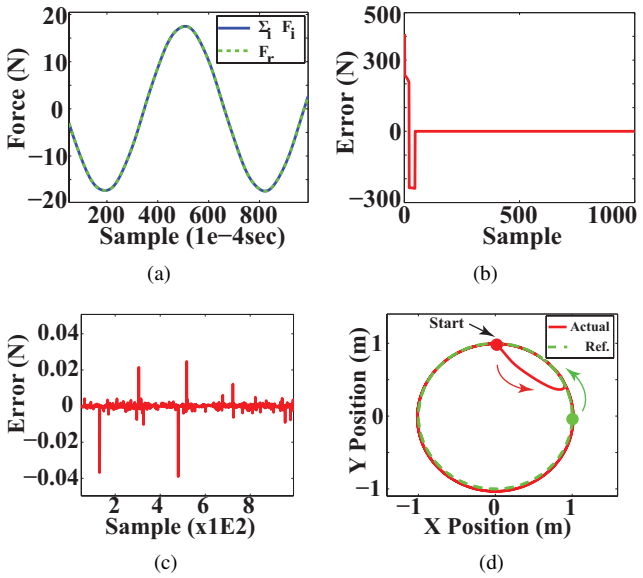

Fig. 6. This shows the high accuracy and precision with which the optimization breaks up the generalized force and torque into forces to apply to the object through the finger endpoints. (a) shows one dimension of the reference force, and the sum of the multiple endpoint forces with three end effectors. (b) and (c) show the error between the two signals, with (c) zoomed to see the fact that the force sum is accurate to approximately 10 mN. (d) shows the object being pushed through a circular trajectory by three fingers initially not in contact (the circles denote the starting points). The reference is tracked very well.

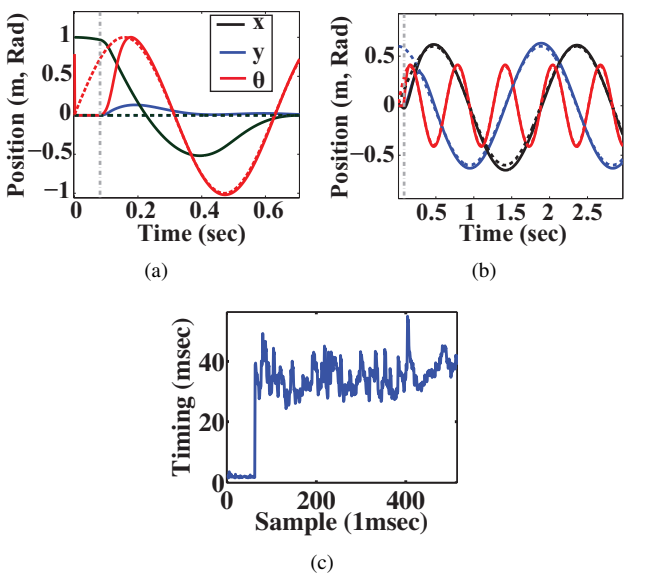

Fig. 7. (a) The object position is shown in solid lines (x, y, θ), respectively, with each corresponding reference trajectory as a dashed line. The contact point happens at various intervals with the first finger contacting at t=0.081 sec. approximately, and the last approximately 50 msec. later (three finger manipulation). After the first finger contact, the object begins a trajectory toward the goal angle and position. This is desirable in object manipulation where the number of fingers in contact is not as important as controlling the trajectory of the object. (b) Same color plot representations, this time demonstrating rapid tracking of all three axes. (c) Timing of the computational loop in Matlab (even with realtime graphics plots this is fairly fast).

IV. Results

A. Convergence and execution speed

As the optimization is convex, convergence was achieved very rapidly. The iLQG algorithm, since it is solving a simplified high level system which is lower dimensional, converges in an average of two iterations. The entire loop, though not fully optimized, took on average 1.1msec when no contact occurred, and, with four manipulators 34.5msec during contact (Fig. 7(c)). This is executed in Matlab R2007a running on a macintosh computer with a 2.3GHz dual core processor. This is fast enough to run in realtime, though it can be further optimized.

B. Tracking the high level reference forces

The ability to reproduce the reference forces required by the high level control is clear in Fig. 6(a)-6(d). The error between desired force and combined forces is very low at under 20mN when the forces are on a scale of 20N. The object can be made to track a fast-moving circle reference, while keeping the object horizontal. The error is quite Gaussian, which is to be expected. There is no evidence of oscillation or instability. Indeed, the algorithm, during impossible initializations (fingers too far apart, or object too low to grip), behaved well, with no unpleasant accelerations or oscillations. Instead a very biological-appearing behavior of reaching after the falling object appears.

C. Trajectory tracking in object manipulation

Fig. 7(a)-7(c) shows that the object position and orientation can be controlled using this methodology. This control works with various numbers of manipulators, as a result of the high level making a plan and the low level executing it. Thus the low level can change structure (number of manipulators, physical structure, etc) and the algorithm will produce similar results. It is interesting to note that contact timing for each end effector is slightly different, but the algorithm's method of recomputing optimal forces to apply given current configurations consistently works, since it is the same output, for example, with two fingers and one not in contact, as if there were only two fingers in existence. The contact locations would change, but as the movement progresses through time the grip would change to be more optimal for two fingers only.

D. Contact breaking and continuous motions

Fig. 8(a)-8(c) depicts the manipulators successfully rotating an object (quite rapidly) to track a constant angular velocity. The fingers move through a number of gait transitions as necessary, similar to Fig. 8(c). It is worth noting that the trajectory, contact point, and moment of contact breaking are all implicit behaviors defined in the middle level. As one can see in Fig. 8(b), it is possible that the parameters of the system can be optimized further to more closely track the reference, but object manipulation is not about perfectly tracking a reference, as in fact it has been shown that some error is completely ignored in the human sensorimotor system. Manipulation and movement in general is performed in order to achieve an overall goal, regardless of small perturbations of the system or small errors.

V. Conclusion

Object manipulation using multiple manipulators is a complex problem. Optimal control and model predictive methods appear to yield very useful strategies for addressing the multitude of behaviors which are required to create a

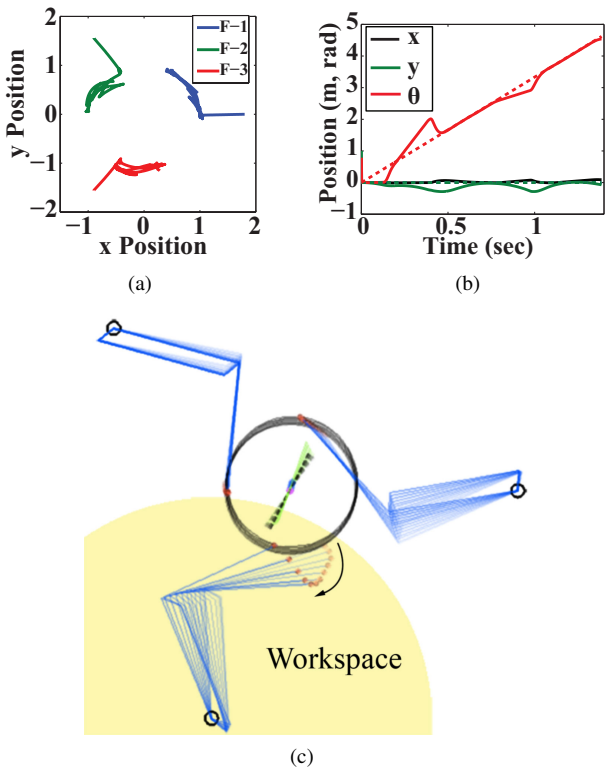

Fig. 8. (a) Depicts xy-data of three manipulators rotating an object, and changing grip a number of times. The object-finger grip position relationship is interesting in that the 'gait' of the fingers varies depending on how large of a correction is required as the finger needs to reposition itself. (b) Depicts a portion of this data, where the objective is to rotate the object continuously at a particular velocity while maintaining xy-center position. (c) One manipulator during a contact break and return trajectory. The manipulator, as it nears the edge of its workspace (depicted as the shaded area), breaks contact, moves away from the object, and then follows a trajectory to a central-to-the-workspace contact point on the object. Note that collisions are only computed between end effectors and the object, and end effectors and each other.

manipulation algorithm that deals with more than a single simple behavior. However, attacking it as a single part problem often is not feasible in realtime using even the most modern techniques and hardware. Since creating solutions which are adaptable and dynamic on-the-fly are important to emulate the complex behavior patterns of biological systems, a hierarchical control approach was proposed in this paper. Is this the one and only way to solve the problem? Is the brain doing exactly this? Though there is evidence that the brain does address control in an online and hierarchical fashion, more than half of the neurons in the brain are devoted to movement, so any single approach will not capture the richness of behavior patterns available to the biological organism. In addition, we realize that the model is not the reality, and so we do not claim this is precisely the process the brain follows because that would be incorrect.

The method proposed here makes the process of generating new behaviors simply a matter of creating new terms in the FC function. This is among the first algorithms to demonstrate, without motion capture data, a method for object manipulation which can not only control an arbitrary object to track a reference trajectory, but also when necessary break contacts and determine how to grip an object successfully. This method allows minimal 'tweaking,' and can potentially have all parameters learned online. It also runs in realtime with no pre-computing required to solve the problem. The method of handling joint limits by a workspace with a spring force boundary which pushes the system back into the workspace is drawn from biological systems, and works well, and avoids rigid nonlinearities which can contribute to numerical instabilities.

Future work will compare these results to biological systems in depth, apply the method to physical robots, apply it to locomotion, and extend the equations into 3D and soft contacts. While we have not reported quantitative results regarding robustness to parameter perturbations, and this is another item to present in the next paper, MPC with feedback tends to be quite forgiving regarding these issues. The difference becomes an error signal which is fed back into the system, and then is compensated for automatically. In fact, sometimes a simple model is purposefully used, with the intent of pushing the system to behave closer to the model. This can also be seen in biological systems, where the system behaves well even in the face of a poor model.

Solving the problems of locomotion and manipulation will not only contribute new mathematical techniques to science and engineering, it will also contribute to the development of new rehabilitation strategies, artificial limbs, assistive technologies, and far more.

References

[1] H. Asada and J.-J. E. Slotine. *Robot Analysis and Control*. John Wiley and Sons, New York, NY, 1986.

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

[3] 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.

[4] E. Kandel, J. Schwartz, and T. Jessell. *Principles of Neuroscience*. McGraw-Hill, Inc., New York, 3rd edition, 1991.

[5] E. Kefalea, E. Mael, and R.P. Wurtz. An integrated object representation for recognition and grasping. *Knowledge-based intelligent information engineering systems. Third conference*, 1999.

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

[7] W. Li, E. Todorov, and X. Pan. Hierarchical feedback and learning for multi-joint arm movement control. *In Proceedings of the IEEE Engineering in Medicine and Biology 27th Annual Conference*, 2005.

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

[9] D. Liu and E. Todorov. Evidence for the flexible sensorimotor strategies predicted by optimal feedback control. *Journal of Neuroscience*, 27(35):9354–9368, 2007.

[10] D. Liu and E. Todorov. Hierarchical optimal control of a 7-dof arm model. *In proceedings of the IEEE Symposium of Adaptive Dynamic Programming and Reinforcement Learning*, 2009.

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

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

[13] 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.

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

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

[16] E. Todorov and M. Jordan. A minimal intervention principle for coordinated movement. *Advances in Neural Information Processing Systems*, 15:27–34, 2003.

[17] E. Todorov and W. Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. *In proceedings of the American Control Conference*, 2005.

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