Goal Directed Multi-Finger Manipulation: Control Policies and Analysis

S. Andrews, P. G. Kry

School of Computer Science and Centre for Intelligent Machines, McGill University, Canada

Abstract

We present a method for one-handed, task-based manipulation of objects. Our approach uses a mid-level, multi-phase approach to organize the problem into three phases. This provides an appropriate control strategy for each phase and results in cyclic finger motions that, together, accomplish the task. The exact trajectory of the object is never specified since the goal is defined by the final orientation and position of the object. All motion is physically based and guided by a control policy that is learned through a series of offline simulations. We also discuss practical considerations for our learning method. Variations in the synthesized motions are possible by tuning a scalarized multi-objective optimization. We demonstrate our method with two manipulation tasks, discussing the performance and limitations. Additionally, we provide an analysis of the robustness of the low-level controllers used by our framework.

Keywords: grasping, manipulation, hand animation

1. Introduction

Animation of human manipulation is a difficult and time consuming task. It is arguably one of the most challenging genres of human motion to synthesize due to the fact that it involves the coordination of many degrees of freedom and multiple contacts. Furthermore, successful simulation of physically based manipulation depends on many variables such as the shape, size, texture, and physical properties of the object being manipulated.

Problems related to grasping and manipulation have received significant attention from both the computer animation and robotics communities, with extensive work addressing the issues of motion planning, contact placement, and grasp quality. In this paper, we focus on physically based simulation of one-handed manipulation. A key feature of our approach is that we do not require a scripted path for the object. Instead, we specify only the goal, allowing the object trajectory to be influenced by hand geometry and finger motions. We believe this is useful for creating plausible human-like manipulation, and relevant to many scenarios where only the final object configuration is important (e.g., preparing a coin for insertion into a vending machine, rotating a small package to read its label, or orienting small parts as part of a larger assembly task).

In contrast to high-level motion planning techniques that solve complex problems through a sequence of actions, we instead take a mid-level control approach that is well suited to grasp repositioning tasks. We introduce an automata-based controller architecture that produces cyclic finger gaiting actions, wherein contact related events trigger different low-level controller phases. Adjusting a volume dial or removing a lid from a jar are simple examples that work well with this approach; the goal can be achieved by chaining together a number of similar turning actions with repeated releasing and re-grasping interleaved to reposition the contacts. We term these three phases *approach*, *actuate*, and *release*, and a ball-in-hand re-orientation task as a more complicated example.

Continuous optimization is used to compute the parameters necessary for our mid-level controller phases. The objective is to successfully perform a manipulation task, but also to produce a physically plausible and natural motion sequence. Our controllers tend to work well for a collection of nearby states, and because several cycles are often necessary to reach farther goals, we build a policy using reinforcement learning (RL) and interpolation of controller parameters. Unlike traditional RL, a discretized action space is not used and parameter selection occurs in a continuous manner for each state. This learning approach helps us tune the release phases so that fingers are better positioned for improved progress toward the goal in future cycles. More importantly, once the policy has been computed, it is useful for simulation of goal directed manipulation in real time.

Although motion capture is not used, we do use a selection of natural hand poses to compute a reduced parameter space for our low-level controllers. This limits the number of degrees of freedom that we need to include in searching for solutions, and encourages the use of natural hand poses. Despite the reduced degrees of freedom, we still have a full simulation that produces poses outside of the reduced pose space, with finger joints bending to accommodate contacts.

We believe our method makes important progress toward the development of improved virtual humans that can perform successful goal oriented physically based interactions with virtual objects. Our main contributions build on our previous work [1] and include:

Email addresses: sheldon.andrews@mail.mcgill.ca (S. Andrews), kry@cs.mcgill.ca (P. G. Kry)

- A novel framework for synthesizing motions for human manipulation problems where the generated motions exhibit finger gaiting;
- A reduced search space based on natural poses to increase the performance of our method while ensuring the use of plausible hand shapes;
- Learned control policies that run in real-time;
- An analysis of the robustness of the control policies, providing insight into the selection of learning parameters, as well as providing indications on how to improve the low-level controllers used by our framework.

2. Related Work

A variety of control strategies can be used in object manipulation tasks, of which contact changes are a critical aspect. Work in neurobiology observes that changes in motor control are triggered by discrete events, with the contact information provided by different mechanoreceptor signals [2]. When this information is suppressed, it becomes difficult to perform fine manipulation. For instance, imagine trying to open a combination lock with fingers numbed by cold.

In physically based computer animation, contact changes are also important and are included in the design of finite state machines and automata-based controllers. Such controllers are a natural choice for modeling virtual motor control involving environmental interactions, such as grasping, manipulation, and locomotion. Pollard and Zordan [3] present a physically based grasping simulation that combines motion capture at the wrist, key poses selected from the capture, and a finite state machine. Their method only performs a grasp and release of an object, but the state machine and transitions are fairly similar to the controller we use. However, their approach uses a simple heuristic for triggering the release of the object, as opposed to a grasp quality metric.

In an early approach to this problem, we attempted to direct the hand through a learned policy of optimal joint angle velocities, as in the *motion fields* work of Lee et al. [4]. However, it proved difficult to generate motions that remained stable in the contact-rich environments typical of human manipulation tasks. Instead, we found it much more tractable to use a mid-level control approach, whereby contact related events trigger specific controller *phases*.

In robotics work, Huber and Grupen [5] demonstrate robust finger gaits from closed-loop controllers. Their work is similar to ours, but they do not use a latent parameter space to compute control poses, and their technique does not result in a policy that can be used for different scenarios with changing goals. Other robotics work has used a multi-modal control approach to perform motion planning for full body manipulation tasks. Hauser et al. [6] break down the planning problem for robot pushing tasks into a sequence of walking, reaching, and pushing motions. These modes are high-level compared to the phases used by our controller framework. Their approach uses shorter phases (10-100 *ms*), making exploration costly for scenarios where high branching factors exist. Our work schedules phase transitions according to discrete events within the simulation, resulting in longer phase durations (typically 200-1000 *ms*).

Other work has performed dexterous manipulation from a grasping pose by optimizing the forces necessary to move a manipulated object on a pre-specified trajectory [7]. These optimized forces are then used to drive finger motions with appropriate torques at the joints. Our work differs in that the complete trajectory of the object is not known *a priori*, and we do not require an initial grasp. In contrast, and more recently, Ye and Liu [8] use contact sampling to animate fingers given motion captured data for the object. Interestingly, they note that it is important that the motion of the object come from a captured manipulation, as opposed to a key-framed trajectory, for finger motions to appear natural. This is not unexpected, and is part of our motivation for using a goal based approach, as opposed to simplifying the problem by first scripting or planning a path for the object.

In our work, we focus on goal directed dexterous manipulation. Mordatch et al. [9] present a solution to this problem that produces impressive results. Their approach solves a sequence of space-time constraints with a special treatment for contact. They avoid optimizing the motion of each finger joint by considering only end effector positions, and finger poses are reconstructed with inverse kinematics. In our approach we simulate all the finger joints, and our optimization takes the form of a shooting method as opposed to encoding physics as constraints. As a result, our solutions have better physical plausibility and hard contacts, though we are only able to solve well structured manipulation in comparison.

To speed up our optimization, we recognize that aspects of the grasping problem can be described in a low dimensional manner. Santello et al. [10] show that the variation in final imagined grasp poses for a large number of objects is quite small, with well over 80% of the variation explained by only two principal components. Recently, Ben Amor et al. [11] used a low-dimensional sub-space built from a database of recorded human grasping postures to perform grasp optimization on a robot. The focus of their work is complementary to ours in that their approach synthesizes motions for "reach-and-grasp" tasks. Also, we avoid correspondence problems between human and robot kinematics by allowing the user to build a pose corpus directly using the simulation model.

In other work, there has been progress in resynthesizing human grasping motion. Kry and Pai [12] capture forces and motion with the objective of estimating finger stiffnesses to use in a simulation to resynthesize the captured motion. The controller in this case is entirely feed-forward. While the resynthesized interactions have a natural motion due to estimated compliance, there is no feedback to ensure the resulting final object position and orientation match a desired goal.

An important part of our work is that we use continuous optimization and machine learning to compute successful controllers, which produces a policy that can be used in a real time simulation. In the context of locomotion, Coros et al. [13] use reinforcement learning to create a policy that provides a controller to perform a series of walking tasks (e.g., walking on a line). Their controllers benefit from a learned control policy in that they are made more robust by interpolating optimal control parameters from nearby states. Similarly, Wang et al. [14] perform optimization for walking controllers that anticipate perturbation.

Finally, Okamura et al. [15] provide an overview of dexterous manipulation in robotics, and discuss the idea of mid-level control, which we use in our work.

3. Controller Structure

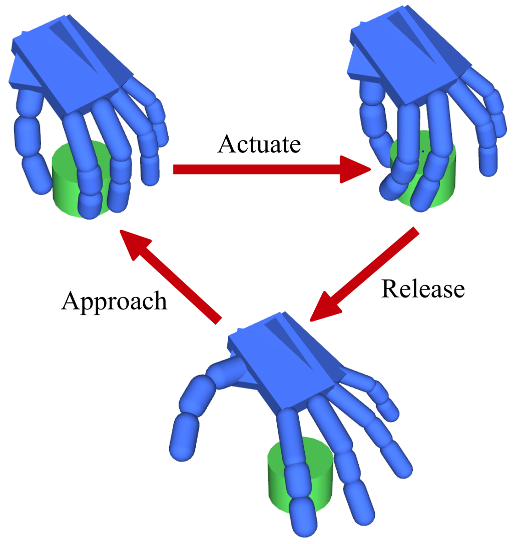


Figure 1: Our three phase mid-level control strategy.

Our approach is motivated by the observation that human finger motion exhibits pseudo-cyclic characteristics, or finger gaiting, for a broad range of hand manipulation tasks. The fingers move in a coordinated fashion with an effort determined by one of three distinct phases: (i) a pre-shaping and finger planting phase wherein the hand forms a stable grasp around the object, (ii) an actuation phase in which wrenches due to contact forces are used to translate and rotate the object toward some desired configuration, and (iii) a release phase wherein the fingers adjust to a pose that is suitable for a subsequent approach phase. We refer to these phases simply as approach, actuation, and release (see Figure 1).

These phases represent strategies that are encompassed by an automata-based controller architecture. Each controller uses a set of three reference poses, $(\tilde{q}_0, \tilde{q}_1, \tilde{q}_2)$, to guide the hand in order to accomplish a manipulation task. The method for selecting these poses is discussed later, in Section 4.1.

During the ith phase, we apply joint torques, τ, computed as

$$\tau = K(\tilde{q}_i - q) - D\dot{q} , \qquad (1)$$

where K and D are the joint stiffness and damping matrices, respectively. The hand model used in our experiments is shown in Figure 2. Each pose consists of 20 joint angles corresponding to the degrees of freedom of the hand.

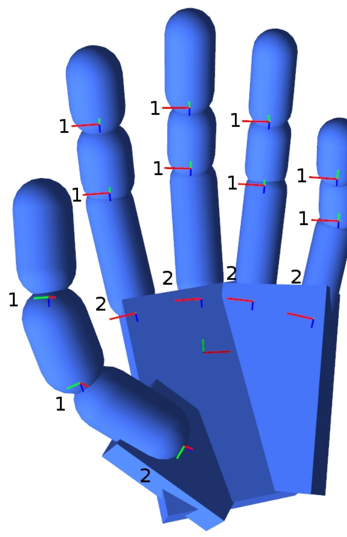


Figure 2: Our hand model showing the associated number of degrees of freedom at each of the joints.

Throughout the rest of this paper we use the integer subscripts $0, 1, 2$ on scalar and vector parameters to denote a correspondence with each of the approach, actuation, and release phases, respectively.

3.1. Phase Transitions

Phase transitions occur asynchronously and are tied to contact and joint limit events occurring within the simulation. In this section, we describe the conditions used to trigger transitions between phases.

The initial obstacle faced in many grasping problems is determining where to form finger-object contacts so that manipulation may be performed. This is the main objective of the approach phase, wherein pre-shaping occurs and finger end-effectors ultimately make contact. It is most beneficial to end in a configuration that results in a stable grasp and good dexterous potential for manipulating the object. The subsequent phase involves actuating the object using contact forces, typically until further actuation is no longer possible (i.e., due to joint limits) and some or all fingers break contact. The hand then moves toward a recovery pose where pre-shaping and approach can begin again, repeating the cycle.

The approach phase ends once the fingers have planted and the desired grasp quality, $\tilde{Q}$, has been achieved. Quality here means that some stable, dexterous manipulation is possible, and there are several possibilities for measuring this quantitatively. We use a metric that is computationally inexpensive, but effective. Details about estimating grasp quality are provided in Section 4.2.1. Once the grasp quality condition is met, the approach phase transitions to the actuation phase.

At this point, the fingers are ready to manipulate the object. The direction of manipulation is a result of contact with the object and the accumulation of joint torques as computed by the proportional-derivative (PD) control given in Equation 1.

During actuation, joint torques are applied until the grasp quality drops below an acceptable threshold, indicating that dexterous manipulability is no longer possible and the controller transitions to the release phase.

The transition between release and approach occurs when the total joint velocity of the fingers becomes small, indicating that the desired pose has been reached, or that motion is hindered due to contact forces; there is also a transition when the allotted time for the phase has elapsed. The controller state is set to the approach phase and the cycle repeats. Contact information is not used to trigger a transition out of the release phase.

Note that there is no assurance that a stable grasp is maintained during the release phase. However, it is assumed that "good" trajectories ultimately lead to the goal state; this information is encoded in the *value function*, $V(s)$. We include the value function as part of the selection process for controller parameters. This a subtle, but important, aspect of our approach and further details are provided in Section 4.4.

For all phases, we force a transition to the next phase if the joint velocities of the hand become small or the duration of a phase exceeds a maximum value, T_{max}. The one exception is when the goal has been reached, in which case the hand holds in either the actuate or release phase, waiting for the goal to change. The choice here is to let the hand remain in the actuation phase, ready to apply forces to achieve a new goal, or to remain in the release phase, allowing the hand to be moved between objects as part of a higher level control.

4. Control Policy Creation

In this section, we provide details on how to build a control policy for object manipulation tasks, beginning with a description of the simulation environment.

Since our work focuses on single-handed manipulation tasks, state information regarding a full character skeleton is ignored; only the wrist and fingers joints are considered. Therefore, each state vector, s, contains the joint angles of the hand, q, the orientation of the object, θ, and its 3D position, x. The object orientation is stored as a quaternion, and both the position and orientation use a coordinate frame affixed to the wrist of the hand model. A homogeneous transformation matrix is used to transform the object's position and orientation from a global coordinate frame to a hand centric frame. The state vector is updated at each time step.

Other components of the simulation state, such as the the linear and angular velocity of the object and hand joint velocities, are used to initialize the dynamics simulation when evaluating control parameters. However, we found these state components had little effect on the results when querying the control policy for optimal control parameters. This can be partly explained by the quasi-static nature of the hand based on the stiffness and damping control parameters we use. Therefore, we exclude all velocity level quantities from the state when building our control policy function.

An action, a, is represented by the tri-phase controller described in the previous section. Each action is composed of a sequence of three desired hand poses, $(\tilde{q}_0, \tilde{q}_1, \tilde{q}_2)$, which are determined during the offline continuous optimization stage, as described in Section 4.1. The control policy, $\Pi(s)$, provides a mapping from the environment state to an optimal action, a^*, whose control parameters are used to bring the environment to a higher valued state by making progress on the task. Progress occurs by means of a forward dynamics simulation, and the value of being in state s is stored in the value function, $V(s)$.

The value and control policy functions are represented by a k-nearest neighbor (k-NN) function approximator. Distance between neighboring states is computed as a combination of state components. The distance between states s_a and s_b is computed as

$$d(s_a, s_b) = \beta_q \, \|q_a - q_b\| + \beta_x \, \|x_a - x_b\| + \beta_\theta \, \|\log(\theta_a^{-1}\theta_b)\|.$$

Of these components, the distance between hand postures is most critical for selecting the most appropriate action. Our implementation represents the object's position in centimeters. The range of values for this component is similar in magnitude to the angular component, which is measured in radians and computed by the logarithm of quaternions. Hence, the distance between hand postures tends to dominate the function $d(s_a, s_b)$. The scalar values β_q, β_x, β_θ are used to weight contributions of the pose, object position, and orientation components, respectively. Through empirical evaluation we determined that $\beta_q = \beta_x = \beta_\theta = 1.0$ gave good results, and this is coincidentally equivalent to an unweighted metric.

The interpolation weight for the ith neighboring state is computed using an inverse distance-squared kernel,

$$w_i = \frac{1}{\sigma} \frac{1}{\left(d(s, s_i)\right)^2}.$$

Convexity is ensured by computing a normalizing factor $\frac{1}{\sigma}$ such that $\sum_{i=1}^{k} w_i = 1$. The optimal action for an arbitrary state is estimated by interpolating the actions for the k closest states in the policy,

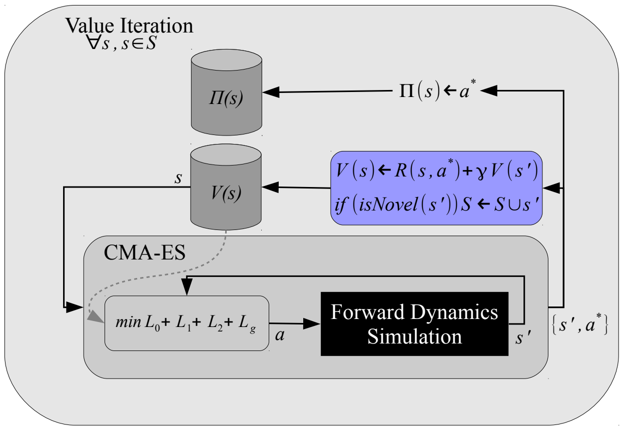
$$a^* = \sum_{i=1}^{k} w_i \, a_i \, .$$

4.1. Value Iteration

A control policy is learned using the value iteration method [16]. The collection of states, S, that make up the instance-based functions $\Pi(s)$ and $V(s)$ is bootstrapped with random states that are chosen uniformly across variations of pose and task.

Controller parameters are determined by performing a multi-objective optimization. For each state $s \in S$, candidate actions are evaluated using a forward dynamics simulation. This means that the parameter search occurs across a non-linear, rugged landscape.

The value function is updated using the reward and value of the proceeding state, s', at the end of the simulation. The method $\textsc{IsNovel}(s')$ uses a threshold, ε, to determine if the state is sufficiently novel and the method returns true if $dist(s, s') > \varepsilon$ for all $s \in S$. Novel states are added to S.


Value Iteration
$\forall s, s \in S$
$\Pi(s)$
$\Pi(s) \leftarrow a^*$
$V(s)$
s
$V(s) \leftarrow R(s,a^*) + \gamma V(s')$
$if\ (isNovel(s'))\ S \leftarrow S \cup s'$
CMA-ES
$min\ L_0 + L_1 + L_2 + L_g$
a
Forward Dynamics Simulation
s'
$\{s', a^*\}$

```
while not converged do
    for $s \in S$ do
        $a^* = \text{OPTIMIZE}(s)$
        $s' \leftarrow \text{FORWARD\_DYNAMICS\_SIMULATION}(s,a)$
        $\tilde{V}(s) = R(s,a^*) + \gamma V(s')$
        $V(s) \leftarrow \alpha \tilde{V}(s) + (1-\alpha)V(s)$
        $\Pi(s) = a^*$
        if $\text{ISNOVEL}(s')$ then
            $S \leftarrow S \cup s'$
        end if
    end for
end while
```

Figure 3: A block diagram visualizing our method (left) and the value iteration algorithm used to build the control policy (right).

Pseudo-code for the value iteration algorithm is provided in Figure 3.

Covariance matrix adaptation (CMA-ES) [17] is used to determine the optimal controller parameters at each state s. The phases are not optimized in isolation, and instead CMA-ES optimization is performed over one complete cycle of the state machine. The coupling between phases is key to our approach, since evaluation of the success of a controller is determined not only by the minimization of a set of objective terms for each phase independently, but by their performance as a sequence.

Each objective term is a function of the simulation and task state across all controller phases, leading to the formulation of a composite objective function

$$\min_{a^*} L_0 + L_1 + L_2 + L_g \; .$$

Here, L_0, L_1, L_2 pertain to the approach, actuation, and release phases, respectively; L_g is an aggregation of global terms pertaining to all phases. The contents of the phase specific and global objective functions are discussed in the following sections.

The terms of the composite objective function are denoted by the symbol L, and subscripts are used to distinguish individual quantities pertaining to phase-specific features. These features are quantities accumulated throughout a phase, or computed at phase transitions. The notation $\sum_{t \in T_i}$ is used to indicate a term that is accumulated over period T_i, with its value being sampled at each time step.

Scaling is used to bring the range of values across objective terms to within an order of magnitude. The scaling factors, which are determined empirically by data collected from manually defined manipulation sequences, are computed once for each task. By scaling each objective term in this way, it simplifies the process of the tuning the weights, l, for the multi-objective optimization problem. The scaling factors are omitted from the equations in order to improve readability.

4.2. Approach

The approach is a pre-shaping phase, wherein the agent makes contact with the object in preparation for actuation. The objective function for this phase is simply

$$L_0 = l_0 \max\left(0, \tilde{Q} - Q_{T_0}\right) \; ,$$

which ensures a penalty if the grasp quality at the end of the phase, Q_{T_0}, is below the desired grasp quality, $\tilde{Q}$. As such, the duration of the approach phase T_0 can equal the time out T if the grasp quality was not achieved, in which case L_0 will be some positive value to penalize this action. Alternatively, if grasp quality is achieved, then T_0 is the time at which the approach phase transitions into activation, and the objective L_0 will simply be zero. Overall, this encourages the optimization method to find solutions where the fingers are planted and ready to actuate the object.

4.2.1. Grasp Quality

In this section, we describe our method for computing the grasp quality. A common metric for determining grasp quality is by computing the *force closure* of the *grasp wrench space* (GWS) [18]. If the convex hull of the GWS contains the origin, it has closure, and any external wrench acting on the object can be resisted by a convex combination of the grasp wrenches. However, computing the convex hull at each simulation step is a computational bottleneck, and instead we estimate the grasp quality with an ellipsoid, similar to the method outlined by Klein and Baho [19].

The matrix $G \in \mathbb{R}^{6\times(mN)}$ is assembled from a set of representative vectors, accounting for N finger-object contacts and m basis vectors at each contact that approximate the Coulomb friction cone (as shown in Figure 4). The positive linear span of the friction cone represents the set of potential forces a contact may apply to the object.

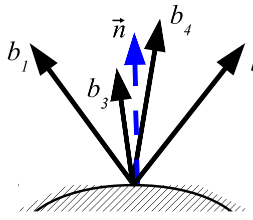


Figure 4: Discretized friction cone showing the contact normal, $\vec{n}$, and the frictional basis vectors, $b_{1\ldots4}$.

For our experiments $m = 4$, and the basis for the ith contact is denoted by $\left(b_{i_1}, b_{i_2}, b_{i_3}, b_{i_4}\right)$.

The matrix, $\Gamma_i \in \mathbb{R}^{6\times3}$, is used to map contacts forces in the global coordinate frame to wrenches in the object's local frame. This matrix has the block form $\Gamma = \left[R^T - \hat{p}R^T\right]^T$, where R is a rotation matrix transforming vectors in the global coordinate frame to the object's coordinate frame, and p is the contact location used to form the skew symmetric cross product matrix $\hat{p}$. This gives the set of wrench vectors

$$W_i = \left[\Gamma_i b_{i_1} \quad \Gamma_i b_{i_2} \quad \Gamma_i b_{i_3} \quad \Gamma_i b_{i_4}\right], \tag{2}$$

and G becomes the block row matrix $G = [W_1 \ \dots \ W_N]$.

We compute the singular value decomposition $G = U\Sigma V^T$, and estimate grasp quality, Q, as the smallest singular value of G. By avoiding poses where the singular value is equal or close to zero, the optimization is more likely to select non-singular grasp configurations. The columns of U provide the axes of the wrench ellipsoid, and the axis corresponding to the smallest singular value provides a direction in which the least amount of force and torque is needed to break the grasp. Informally, the smallest singular value corresponds to a GWS direction that is "weakest".

The example illustrated in Figure 5 shows the convex hull computed for a grasp in a 3D wrench space. The grasp lacks closure since the hull does not contain the origin yet the wrench ellipsoid is not degenerate and does contain the origin. Therefore, as an additional check, we ensure that the cone spanned by the contact wrenches is at least π, otherwise $Q = 0$. Although this approximated metric often leads to grasps which have the force closure property, there is no assurance of stable grasps throughout a manipulation. Rather, this heuristic sufficiently guides the optimization towards manipulable and plausible grasps.

We justify the use of the wrench ellipsoid heuristic by considering computational performance. For complex scenarios involving ~ 10 contacts, the time required to compute the wrench hull with an Intel 3.2 GHz processor is 15 *ms* compared to 0.1 *ms* for the wrench ellipsoid based quality metric. In our experiments, there was little difference in the motions generated using these two quality metrics, so we choose the one that is less expensive to compute.

4.3. Actuation

It is during the actuation phase that most of the progress is made on the task. Wrenches acting on the object change its position and orientation such that it moves toward the goal state. Based on this assumption, it is necessary that the predominant objective for this phase is to minimize the task-based objective function

$$L_T = l_x\|\tilde{x} - x\| + l_\theta\|\log(\tilde{\theta}^{-1}\theta)\|.$$

Note that $L_T \geq 0$, and the minimal value occurs when the goal state is reached.

The value function should reflect the optimality of the task state. By using the reward function

$$R(s) = -\left(\|\tilde{x} - x\| + \|\log(\tilde{\theta}^{-1}\theta)\|\right),$$

this results in a $V(s)$ that is non-positive for any state s. Choosing an action that minimizes the objective term L_T will

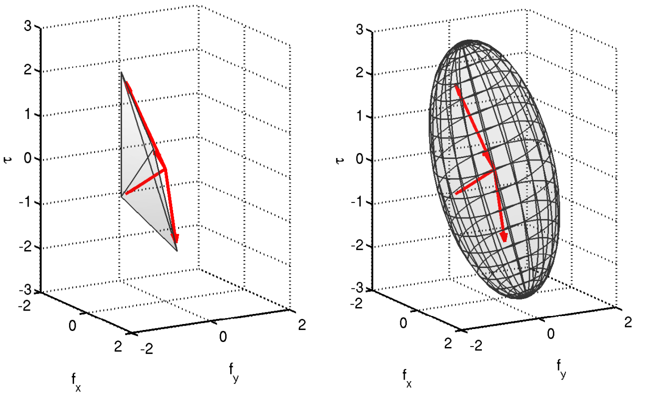


Figure 5: The wrench convex hull (left) of a 2D grasp, with two linear force (f_x,f_y) components and an angular torque (τ) component; the 3D wrenches are drawn in red. The grasp does not have form closure (the hull does not contain the origin), yet the wrench ellipsoid (right) is not degenerate. An additional check is performed to ensure that the grasp wrench space spans a cone of at least π.

maximize the return of the reward function, meaning that in a greedy sense progress is made on the task. This duality is also exploited during the release phase for choosing a recovery posture by incorporating $V(s)$ as an objective term, maximizing future rewards.

During manipulation, it is also a requirement that the fingers maintain a certain degree of stability with the object. Using the same metric from the approach phase, a minimum level of grasp quality is maintained throughout the actuation phase by the objective

$$L_Q = l_Q \frac{1}{T_1} \sum_{t \in T_1} \max\left(0, \tilde{Q} - Q_t\right).$$

Here, T_1 is the duration of the actuation phase and Q_t is the quality at time t of the actuation phase.

Additionally, we include a penalty term allowing the user to specify which of the M fingers participate in the actuation. An array of boolean values, p, contains an entry for each finger, indicating if it should participate– *true* if participating, *false* otherwise. The summed magnitude of contact forces affecting each finger is computed as

$$F_j = \sum_{k}^{N_j} \|f_{j,k}\|,$$

where $f_{j,k}$ is the kth of N_j contact forces between the object and finger j.

If the value of F_j for a non-participating finger exceeds a threshold, η, a penalty proportional to the contact force is added. Conversely, if F_j for a participating finger falls below η, a penalty is also added. The penalty is accumulated at each time step of the phase as

$$L_P = l_P \sum_{t \in T_1} \sum_{j}^{M} \begin{cases} \eta - F_j & \text{if } F_j < \eta \text{ and } p_j \\ F_j - \eta & \text{if } F_j > \eta \text{ and not } p_j \\ 0 & \textit{otherwise} \end{cases}.$$

For the results shown in this paper, $\eta = 2.0$.

Assembling each of the task, quality, and non-participating finger penalty terms, the total objective function for the actuation phase is

$$L_1 = L_T + L_Q + L_P.$$

4.4. *Release*

The primary objective of the release phase is to let the fingers break contact and move them in preparation for another approach. Since the types of tasks with which we are concerned focus on re-orientating and re-positioning an object, any spatial velocity of the object during this phase is penalized in order to ensure that progress made during the actuation phase is not undone. We introduce objective terms that penalize any spatial velocity throughout this phase:

$$L_\omega = l_\omega \sum_{t \in T_2} \|\omega_t\|_2$$
$$L_v = l_v \sum_{t \in T_2} \|v_t\|_2 \ .$$

Here, ω_t and v_t are the angular and linear velocities of the object at time t.

However, simply minimizing the spatial velocity of the object will not ensure successful manipulation, since the hand must recover to a pose where it can make progress during the subsequent approach phase. Upon transitioning to this phase, if the object has reached the target configuration, the hand simply maintains a static posture to hold the object. However, consider the case where the configuration of the object is not the goal state. Progress must be made in subsequent phases. Given the control policy, $\Pi(s)$, states corresponding to future progress are considered high value states. High value states are discerned using the value function, $V(s)$, and we include it as part of the optimization for the release phase to determine a good recovery posture.

Conveniently, like the reward function, the value function has an upper bound of 0 and lower bound of $-\infty$. Therefore, we can use its value directly as a penalty term in the controller optimization problem as

$$L_V = -l_V \ V\left(s_{T_2}\right),$$

where s_{T_2} is the state of the simulation at the end of the release phase. Note that a negative scalar is used to weight this objective term. By minimizing L_V, postures that result in good finger planting at the subsequent approach phase are encouraged.

At the beginning of each inner loop, the covariance matrix entries pertaining to the release phase are reset. This is necessary because the value function changes at each iteration of the algorithm. Thus, the solution found for the release pose in a previous iteration may no longer minimize $V\left(s_{T_2}\right)$.

Combining the wrench penalty and value function terms, the objective function for the release phase is

$$L_2 = L_\omega + L_v + L_V.$$

4.5. *Global Terms*

In addition to the individual objectives for each phase, a global penalty term was added to minimize the energy used to perform the action, and to discourage contacts that use surfaces on the back of the fingers. The global component of the objective function is

$$L_g = l_\tau \sum_{t \in T'} \|K\left(\tilde{q} - q(t)\right)\| + l_h \sum_{t \in T'} \sum_{j}^{M} h_j(t),$$

where $h_j(t)$ is equal to 1 if there is contact on the back of finger j at time t, or 0 otherwise. This is determined by comparing the contact force with a vector defining the backhand direction of each finger segment. We use T' to denote the time interval necessary to complete a full controller cycle.

4.6. *Tractability and Implementation*

In addition to using a grasp quality metric which is simpler and faster to compute, we have made a few other technical choices which greatly reduce the computing time of the CMA-ES optimization required for controller selection. For instance, we use a parallelized implementation of the OPTIMIZE(s) method. On a modern multi-core CPU, this reduced the time required to compute an optimal action by nearly an order of magnitude.

Rather than performing optimization in the full coordinate space, we were inspired by the work of Santello et al. [10], which suggests human hand postures, when interacting with tools and everyday objects, may be represented using just a few principal component vectors.

Using a set of approx. 20 grasp poses, from which the user may select some or all, a reduced pose basis is constructed. Figure 6 shows examples of the poses we use for building our reduced pose space. Principal component analysis (PCA) is used to generate a set of orthogonal basis vectors in which to perform the controller optimization. Basis vectors are selected according to their component scores and the set of vectors remains the same for all phases. For the tasks shown in this paper, the parameter space for controlling the joints of the hand is reduced from 20 per phase to just $3-5$ per phase, depending on the task, giving a total of $9-15$ parameters for each tri-phase controller. This significantly improves the performance of our optimization algorithm.

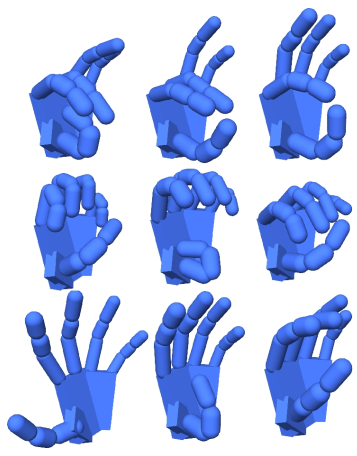


Figure 6: A sample of the poses used to build a reduced basis for the control parameter search.

4.7. Implicit Joint Torques

Constraint force mixing (CFM) is a feature of several off-the-shelf rigid body physics engines and is often a required, practical consideration for building stable simulations of complex systems. We use CFM to implicitize the joint torques needed in Equation 1. This permits taking large time steps ($\Delta t = 1/60$ s) while remaining stable. For the range of values of the stiffness and damping coefficients used by our experiments, it would require advancing the simulation at sub-millisecond time steps if PD control torques were computed explicitly [3]. This gives a significant speedup in the time required perform the CMA-ES optimization.

This approach is similar in spirit to the work of Tan et al. [20], and although the formulation is different, the results are equivalent. They correctly identify the analogy between PD controllers and penalty based constrained dynamics.

Consider that a hinge constraint has 5 constrained degrees of freedom and 1 controllable degree of freedom– about the hinge axis. However, the controllable DOF may be considered a "soft" constraint, and some violation of its position is allowed. We refer readers to the related literature for more information on CFM (see Erleben et al. [21]).

5. Results and Discussion

In this section, we provide some results for two examples: dial turning and ball-in-hand manipulation. The tasks involve re-positioning and re-orienting an object to match a desired configuration. A collection of capsule and box collision geometries is used to model the hand, with finger segments being actuated by joints with 1 and 2 degrees of freedom (see Figure 2), for a total of 20 joint angles.

The Vortex toolkit [22] is used to simulate the forward dynamics, including contact and gravity forces. All results were obtained using a 6-core Intel i7 3.2 GHz processor and running 12 simulation threads for the CMA-ES optimization. Conveniently, Vortex supports the method of PD control discussed in Section 4.7 for hinge and universal joints; stiffness and damping coefficients may be assigned directly through the API. For all results presented in this section, the RL parameters $\alpha = 0.8$ and $\gamma = 0.9$ were used.

Although learning times are long, the result is a policy that runs in real time. Individual iterations of our algorithm work like a space-time constraints optimization problem, giving a partial solution for performing the overall task.

Figure 7 shows a temporal sequence of hand poses generated for our two examples. Complete animations are available in the accompanying video.

5.1. Dial Turning

For the dial turning (DT) example, a cylindrical object is constrained using a hinge joint, similar to a mounted dial or volume knob. Since the object is constrained to rotate about a single axis and no linear motion is allowed, the reward function simplifies to

$$R(s) = -\left|\tilde{\theta}_{hinge} - \theta_{hinge}\right|$$

	l_0	l_Q	l_x	l_θ	l_ω	l_v	l_h	l_V	l_P	l_τ	T_{max}
DT	0.4	0.4	1.0	1.0	0.5	0.5	10	0.6	0.2	0.1	0.6 s
BIH	0.4	0.4	1.0	1.0	0.5	0.5	10	0.6	0.0	0.1	1.0 s

Table 1: Parameter values used to learn the example tasks.

where $\tilde{\theta}_{hinge}$ and θ_{hinge} are the desired and current angle of rotation about the hinge axis, respectively. One of the trajectories generated by the learned policy is shown in the bottom row of Figure 7.

The mean wall clock time to perform the controller optimization for a single state s is approximately 18 seconds. For the example shown in Figure 7 (bottom row), the policy was bootstrapped with 16 states, finally growing to 32 states after 23 minutes of running our algorithm.

The CMA-ES parameters used by this example are $\sigma = 0.2$, $\lambda = 30$. The multi-objective weights and other parameter values are provided in Table 1.

5.2. Ball-in-hand

The ball-in-hand (BIH) example involves re-orienting and re-positioning an unconstrained ball. The task state consists of the 3D position and orientation of the object in the hand frame. There is special consideration for states s' where, at the end of the controller optimization, the ball is no longer in contact with the hand. These are not added to S.

On average, it takes approximately 45 seconds to perform the controller optimization for each state $s \in S$. For the example shown in Figure 7 (top row), the policy was bootstrapped with 40 states, finally growing to 320 states after 4 hours 21 minutes of running our algorithm.

The CMA-ES parameters used by this example are $\sigma = 0.4$, $\lambda = 60$. The multi-objective weights and other parameter values are provided in Table 1.

5.3. Variation

It is possible to synthesize variations of motion for a task by changing the weights of the multi-objective optimization problem and the maximum phase duration, T_{max}. Figure 8 shows two styles of motion generated for the dial turning task. For the controllers with shorter phase duration (top row), multiple cycles are required to reach the goal. The controller with a longer phase duration reaches the goal in one cycle, but only the thumb, index and middle fingers are flagged as participating.

5.4. Robustness of Controllers

To evaluate the robustness of the control policy and individual controllers, we performed a series of experiments on the success of performing a task as a function of perturbations to s. A control policy was built using a small number of states (≈ 12) with the value of θ and $\tilde{\theta}$ remaining fixed. The states were carefully selected so that the learned controllers were capable of reaching the goal state at the end of the release phase, or $R(s') = 0$. For this experiment, $\varepsilon = \infty$ to avoid updates to the control policy.

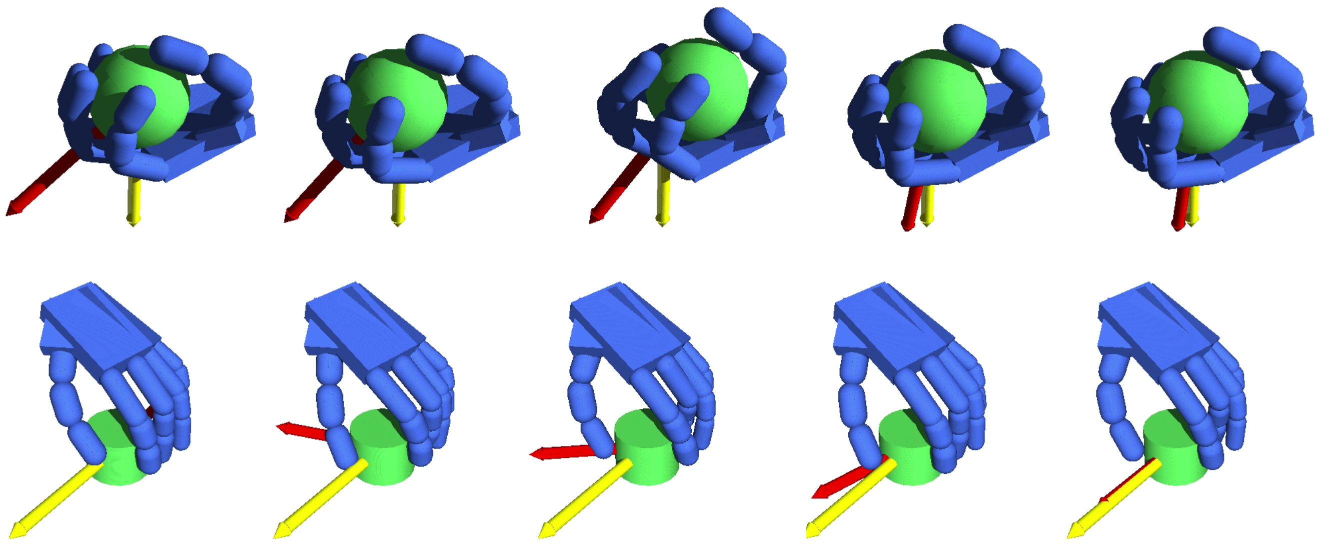


Figure 7: Showing hand motion sequences for the ball-in-hand example (top) and dial turning example (bottom). The desired (yellow) and current (red) configuration are shown.

The state of the simulation is initialized to $s_\delta = s + (\delta(\phi), 0, 0)$, where $s \in S$, and $\delta(\phi)$ is a function used to generate a randomized perturbation to the joints of the hand. Specifically, $\delta(\phi) = [g(\phi), \cdots, g(\phi)]^T$. The function $g(\phi)$ generates a random number with a uniform distribution in the range $[-\phi, \phi]$.

Figure 9 shows the $\bar{R}(s')$ as a was increased from 0.05 to 0.35 radians. Each data point in the plot represents the mean value of R over 100 trials. As expected, when ϕ is small, the controllers perform well. However, as ϕ increases, performance decreases. Increasing k for the nearest neighbor interpolation mitigated this problem.

These experiments provide insight into the selection of learning parameters for the instance-based function approximators. For our experiments, $k = 6$ and $\varepsilon = 0.22$, giving a good trade-off of performance versus learning times.

Further analysis gives additional insight as to why the performance drops as ϕ grows. Figure 10 shows the duration of the approach and actuation phase as ϕ is increased. In the same graph, the average grasp quality, $\bar{Q}$, is plotted. Recall that, during the actuation phase, when the grasp quality drops below a minimum value, it triggers a transition to the subsequent phase. This indicates that the controller phases are transitioning early due to an insufficient grasp on the object. The controllers used by our framework are feed forward in nature, and ignores aggregate features of the simulation, such as grasp quality, throughout the duration of the controller cycle. We speculate that incorporating some feedback about these features will not only improve the performance of our controllers, but also lead to sparser control policies.

5.5. Limitations

The results we have shown involve the manipulation of convex and symmetric objects. Generating control policies that perform well across variations in object size, shape, and mass is a challenge for our framework. Although our method does not exploit these properties for successful manipulation, and no examples involving non-convex or asymmetric objects are provided, there is no evidence indicating that control policies cannot be learned for other classes of objects. This is mainly attributed to the robustness of the CMA-ES method and its ability to find solutions that successfully achieve task objectives. However, properties such as complex object geometry may void the smoothness assumption we make by interpolating controller parameters for a k-NN representation of $\Pi(s)$. This is also true for scenarios where object momentum must be exploited in order for successful manipulation to occur (e.g., contact juggling). One solution may be to increase the density of the function approximator, albeit with the consequence of protracted learning times.

Large-scale motion planning is not considered by our work, since we focus only on single-handed manipulation. In all of our examples, the wrist is immobile, which may be unrealistic for some tasks where motion of the arm, elbow, shoulder, and other body joints may be useful. Note that for the examples shown in Figure 8, additional control parameters (2 joint angles per phase) are used to allow motion of the wrist. However, providing additional degrees of freedom to control other body joints is not appropriate given the restrictions of our phase-based controller architecture. Furthermore, aperiodic finger gaiting styles may not be synthesized without modifications to the controller framework.

6. Conclusion

We have introduced a framework for generating human grasping motion. By building a policy of phase based controllers and performing optimization of control parameters using a forward dynamics simulation, we synthesize motions for a variety of manipulation tasks. Not only are the motions plausible, but since our approach does not assume a pre-defined

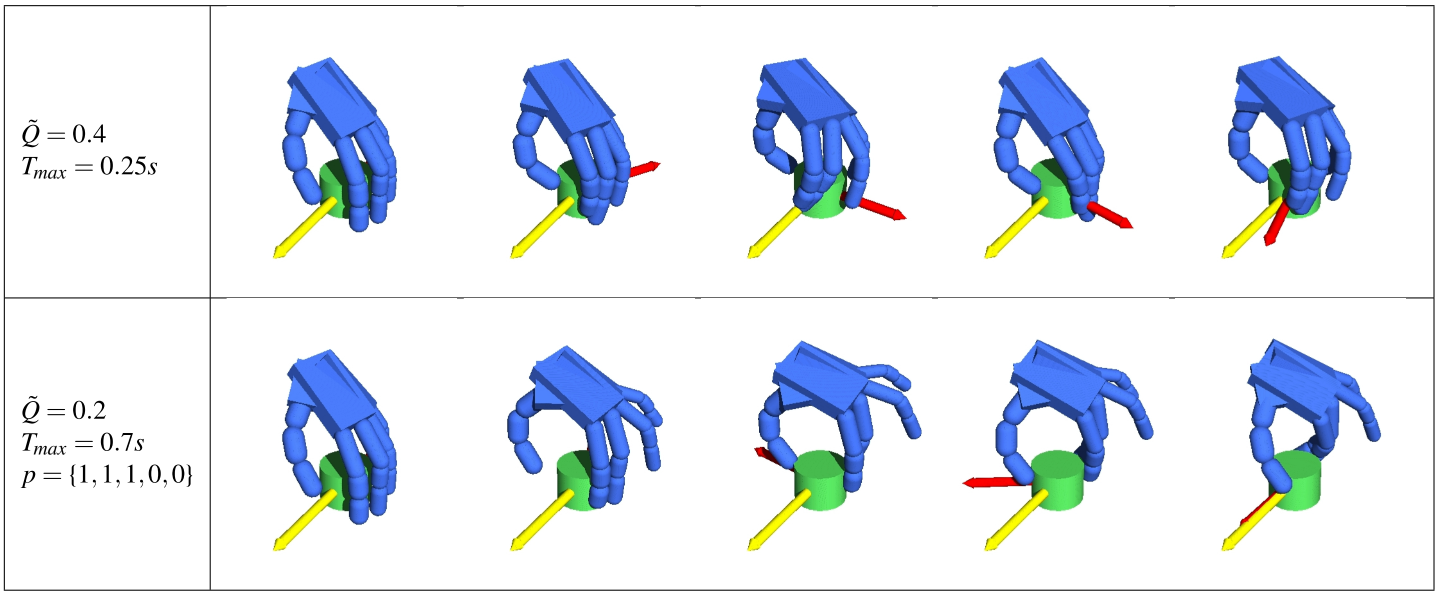


Figure 8: Variations in the synthesized motion may be achieved by changing parameters of the multi-objective optimization.

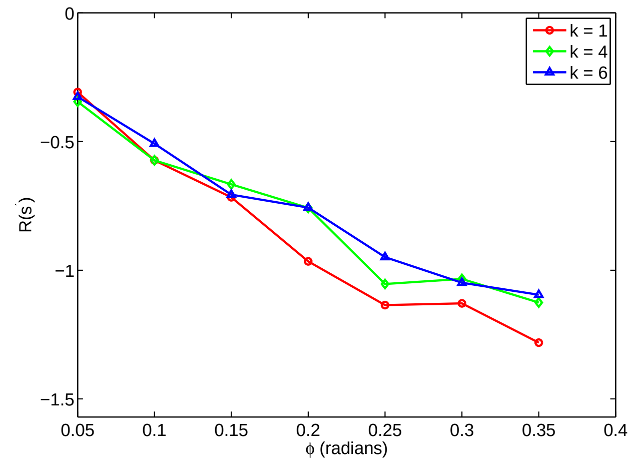


Figure 9: The effect of perturbation, ϕ, on the effectiveness of the controller. Random joint perturbations in the range $[-\phi, \phi]$ are applied to the hand joints. Effectiveness is measured as the reward at the end of the controller cycle, $R(s')$.

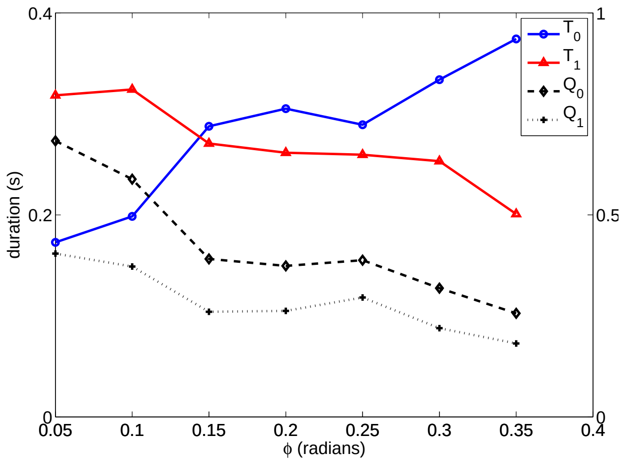


Figure 10: The effect of perturbation, ϕ, on the duration of controller cycles and the grasp quality. The duration of the approach phase, T_0, is lengthened and the duration of the actuation phase, T_1, is shortened. Similarly, the average grasp quality at the end of the approach phase, Q_0, and throughout the actuation phase, Q_1, drops, indicating poor progress on the task.

trajectory and the focus is to achieve a given goal state, the agent is capable of adapting to task changes in real-time. The use of a multi-phase controller architecture also generates motion sequences that exhibit periodic finger gaiting.

As future work, we intend to investigate the use of linear feedback control to improve the robustness of the learned policies. This type of approach has been successfully applied to character locomotion tasks [23] and we intend to investigate its use in the grasping domain. By performing feedback control on grasping features such as grasp quality, net wrench, and contact distribution, it may be possible to not only produce a larger variety of finger motions, but to improve the overall robustness and stability of the control policy.

Additionally, we intend to incorporate contact force and joint data, captured from human subjects, into the framework. A straightforward extension to the work presented in this paper is to use a motion capture corpus to compute the latent parameter space for our low-level controllers. We believe that investigating the force-joint synergies of human grasping is key to building more complex control strategies that correspond to phase transitions that occur in actual manipulation tasks.

Lastly, another potentially interesting avenue of research is in building control policies for coordinated manipulation with two or more hands. One possibility is to treat control policies learned for single-handed tasks as abstract actions within a hierarchical reinforcement learning framework, as in the work by Huber and Grupen [24]. Such strategies remain relatively unexplored in computer animation applications.

Acknowledgments:
We thank the anonymous reviewers for their suggestions for improving the paper. This work was supported by funding from NSERC and GRAND NCE.

References

[1] Andrews S, Kry PG. Policies for goal directed multi-finger manipulation. In: The 9th Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS). 2012, p. 137–45.

[2] Flanagan JR, Bowman MC, Johansson RS. Control strategies in object manipulation tasks. Current Opinion in Neurobiology 2006;16:1–10.

[3] Pollard NS, Zordan VB. Physically based grasping control from example. In: SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation. 2005, p. 311–8.

[4] Lee Y, Wampler K, Bernstein G, Popović J, Popović Z. Motion fields for interactive character animation. ACM Transactions on Graphics (Proc of SIGGRAPH Asia) 2010;29(5):1–8.

[5] Huber M, Grupen R. Robust finger gaits from closed-loop controllers. In: IROS '02: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems; vol. 2. 2002, p. 1578–84.

[6] Hauser K, Ng-Thow-Hing V, Gonzalez-Banos H. Multi-modal motion planning for a humanoid robot manipulation task. Proceedings of International Symposium on Robotics Research (ISRR); 2007.

[7] Liu CK. Dextrous manipulation from a grasping pose. ACM Transactions on Graphics (Proc of SIGGRAPH) 2009;28(3):1–6.

[8] Ye Y, Liu CK. Synthesis of detailed hand manipulations using contact sampling. ACM Transactions on Graphics (Proc of SIGGRAPH) 2012;31(4).

[9] Mordatch I, Popovic Z, Todorov E. Contact-invariant optimization for hand manipulation. In: ACM SIGGRAPH / Eurographics Symposium on Computer Animation. 2012, p. 137–44.

[10] Santello M, Flanders M, Soechting JF. Postural hand synergies for tool use. The Journal of Neuroscience 1998;18(23):2123–42.

[11] Ben Amor H, Kroemer O, Hillenbrand U, Neumann G, Peters J. Generalization of human grasping for multi-fingered robot hands. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. 2012, p. 2043–50.

[12] Kry PG, Pai DK. Interaction capture and synthesis. ACM Transactions on Graphics (Proc of SIGGRAPH) 2006;25(3):872–80.

[13] Coros S, Beaudoin P, van de Panne M. Robust task-based control policies for physics-based characters. ACM Transactions on Graphics (Proc SIGGRAPH Asia) 2009;28(5):1–9.

[14] Wang JM, Fleet DJ, Hertzmann A. Optimizing walking controllers. ACM Transactions on Graphics (Proc of SIGGRAPH Asia) 2009;28(5):1–8.

[15] Okamura AM, Smaby N, Cutkosky MR. An overview of dexterous manipulation. In: Proceedings of IEEE International Conference on Robotics and Automation. IEEE; 2000, p. 255–62.

[16] Sutton RS, Barto AG. Reinforcement Learning I: Introduction. The MIT Press; 1998.

[17] Hansen N. The CMA evolution strategy: A comparing review. Towards a New Evolutionary Computation: Advanceson Estimation of Distribution Algorithms 2006;:75–102.

[18] Murray RM, Sastry SS, Li Z. A Mathematical Introduction to Robotic Manipulation. CRC Press; 1994.

[19] Klein CA, Baho BE. Dexterity measures for the design and control of kinematically redundant manipulators. The International Journal of Robotics Research 1987;6(2):72–83.

[20] Tan J, Liu K, Turk G. Stable proportional-derivative controllers. Computer Graphics and Applications, IEEE 2011;31(4):34–44.

[21] Erleben K, Sporring J, Henriksen K, Dohlmann H. Physics-based Animation. Charles River Media; 2005.

[22] Vortex . version 5.2. Montreal, QC: CMLabs Simulations Inc.; 2012. URL `http://www.vxsim.com/`.

[23] Yin K, Loken K, van de Panne M. Simbicon: Simple biped locomotion control. ACM Transactions on Graphics (Proc of SIGGRAPH) 2007;26(3).

[24] Huber M, Grupen RA. Learning robot control – using control policies as abstract actions. NIPS Workshop on Abstraction and Hierarchy in Reinforcement Learning; 1998.