Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2012)
J. Bender, A. Kuijper, D. W. Fellner, and É. Guérin (Editors)

Policies for Goal Directed Multi-Finger Manipulation

S. Andrews and 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 break the problem into three parts, providing an appropriate control strategy for each phase and resulting in cyclic finger motions that accomplish the task. All motion is physically based, and guided by a policy computed for a particular task. The exact trajectory is never specified as the goal of our different tasks are concerned with the final orientation and position of the object. The offline simulations used to learn the policy are effective solutions for the task, but an important aspect of our work is that the policy is general enough to be used online in real time. We present two manipulation tasks and discuss their performance along with limitations.

Categories and Subject Descriptors (according to ACM CCS): I.6.8 [Simulation and Modeling]: Animation—Human grasping

1. Introduction

Computer 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 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.

Problems related to grasping and manipulation have received significant attention in both computer animation and robotics, 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 and let the object trajectory be influenced by hand geometry and finger motions. We believe this is useful for creating plausible human-like manipulation, and relevant in many scenarios where only the final object configuration is important. Consider, for instance, preparing a coin for insertion into a vending machine, rotating a small package to read its labels, or orienting small parts as part of a larger assembly task.

In contrast to high-level motion planning techniques, which solve complex problems through a sequence of actions, we instead take a *mid-level* control approach that is well suited to grasp repositioning tasks. Specifically, 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 call these three phases *approach*, *actuate*, and *release*, and we also demonstrate a ball-in-hand re-orientation task as a more complicated example.

We use optimization to compute the parameters necessary for our mid-level controller phases. The objective is to produce a successful manipulation, which either moves toward or meets the desired goal. 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 and interpolation of controller parameters. The learning approach helps us tune the release phases so that fingers are better positioned for improved progress toward the goal in future cycles. But more importantly, once the policy has been computed, it is useful for simulation of goal directed manipulation in real time.

DOI: 10.2312/PE/vriphys/vriphys12/137-145

While we do not use motion capture in our low-level controllers, we do use a selection of natural hand poses. 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 include:

- 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 policies that run in real-time.

2. Related Work

A variety of control strategies can be used in object manipulation tasks, and contact changes are always 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 [FBJ06]. 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 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 [PZ05] present a physically based grasping simulation that mixes 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 not too different from 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. [LWB*10]. 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 [HG02] demonstrate robust finger gaits from closed-loop controllers. Their work is quite similar to ours, but they do not use reduced spaces for the desired control poses, and their technique does not result in a policy that can be used in 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. [HVNTH07] 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 phases transitions according to discrete events within the simulation, resulting in longer phase durations (typically $>$ 400 *ms*).

Other work has performed dexterous manipulation from a grasping pose by optimizing the forces necessary to move a manipulated object on a prespecified trajectory [Liu09]. 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 don't require an initial grasp. In contrast, and more recently, Ye and Liu [YL12] 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, in order for finger motions to appear natural. This is not unexpected, is also part of our motivation of 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. [MPT12] 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. [SFS98] 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 principle components.

In other work, there has been progress in resynthesizing human grasping motion. Kry et al. [KP06] capture forces and motion with the objective of estimating finger stiffnesses to use in a simulation to resynthesizing 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 resulting final object position and orientation matches a desired goal.

An important part of our work is that we use continuous optimization and machine learning to compute successful controllers, and ultimately a policy that can be used in real time simulation. In the context of locomotion, Coros et al. [CBvdP09] use reinforcement learning to create a policy that provides a controller to perform a series of walking tasks (for instance, 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. [WFH09] perform optimization for walking controllers that anticipate perturbation.

Finally, Okamura et al. [OSC00] provide an overview of dexterous manipulation in robotics, and discuss the idea of mid-level control which we use in our work.

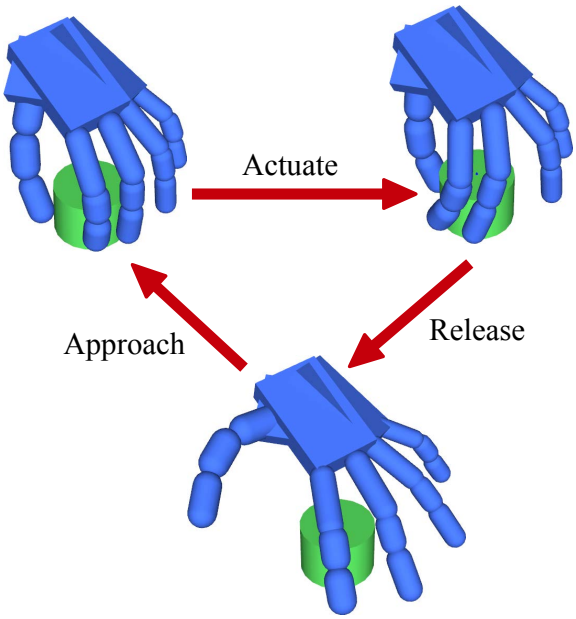

Figure 1: *Our three phase mid-level control strategy.*

3. Controller Structure

We propose a framework that incorporates an automata-based controller architecture. The rationale behind this approach is a result of observations taken during preliminary work and a goal that our work should produce motions that exhibit the pseudo-cyclic nature of human grasping, or *finger gaiting*. We observed that, in animating for human manipulation tasks, the fingers move in coordinated motions among a finite set of similar poses. For a broad range of tasks, these coordinated motions could be broken down into 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 a controller. Each controller uses a set of three reference hand poses $\tilde{q}_0, \tilde{q}_1, \tilde{q}_2$ to guide the hand in order to accomplish a manipulation task. During the ith phase, we apply joint torques, τ, that is computed as

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

where K and D are the joint stiffness and damping matrices, respectively. Although we use diagonal matrices, it is possible to use dense matrices to model coupling effects in the control of different finger joints. For our human hand model, each pose represents joint angles corresponding to the 20 degrees of freedom for the hand as shown in Figure 3.

Throughout the rest of this paper we will use 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

We note that the initial obstacle faced in many grasping problems is determining where to form contacts with the object to be manipulated. During the approach phase, hand pre-shaping occurs to prepare for contact and the finger end-effectors typically end up in a configuration such that there is a stable grasp with good dexterous capabilities for manipulating the object. The subsequent phase involves actuating the object using finger-object forces, typically until further actuation is no longer possible (i.e., due to joint limits) and some or all fingers break contact. The pose of the hand moves toward a recovery pose where pre-shaping and approach can begin again and the cycle repeats. In our control structure, phase transitions occur asynchronously and are tied to contact events occurring within the simulation. In this section, we describe the conditions used to trigger transitions between phases.

The approach phase ends whenever 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, yet effective, and we provide details 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 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) or an allotted time for the phase has elapsed. At this point, the controller transitions back to the approach phase and the cycle repeats. No contact information is used to trigger a transition out of the release phase.

Note that for all phases, we force a transition to the next phase if the joint velocities of the hand become small or the duration of the phase, T, exceeds a maximum value. The one exception is when the goal has been reached, in which case we hold in either the actuate or release phase waiting for the goal to change. The choice here is to either 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.

Each state is represented by the joint angles of the hand, q, the orientation of the object using a quaternion representation, θ, and the 3D position x of the object in the hand frame. The aggregate state vector (q, θ, x) is succinctly referred to as s.

Other components of the simulation state, such as the the linear and angular velocity of the object and hand joint velocities, are used to initialise the dynamic 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.

Algorithm 1 Value Iteration

```
while not converged do
  for s ∈ S do
    a* = OPTIMIZE(s)
    s' ← FORWARD_DYNAMICS_SIMULATION(s, a)
    Ṽ(s) = R(s, a*) + γV(s')
    V(s) ← αṼ(s) + (1 − α)V(s)
    Π(s) = a*
    if ISNOVEL(s') then
      S ← S ∪ s'
    end if
  end for
end while
```

An action, a, is represented by the tri-phase controller described in the previous section, and each action may be decomposed as a sequence of three desired hand poses– (q_0, q_1, q_2). 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. This progression occurs by means of a forward dynamical simulation, and the value of being in a state is provided by the value function, $V(s)$.

We represent the value and control policy functions using a k-nearest neighbor (k-NN) function approximator, with $k = 6$. Distance between neighboring states is computed as a combination of state components. For example, for states s_a and s_b, the distance is

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

Here the distance for the object's orientation is the angle between the two configurations, computed using the magnitude of the quaternion logarithm. Note that this distance is an unweighted metric, and that combining angular and linear distance measures works in our case due to the size of the objects in our simulations.

The interpolation weight for the ith neighbouring state is computed as

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

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 optimal 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 [SB98]. We begin by bootstrapping the value

function using a set of random states, S, chosen uniformly across variations of pose and task.

For each state in S, we solve an optimization problem to determine the optimal action at that state. The value function for the state is updated using the value of the proceeding state, s', and the reward function, $R(s)$. If s' is sufficiently novel, we add it to S. Pseudo-code for the value iteration algorithm is provided in Algorithm 1.

Evaluating the action requires performing a forward dynamics simulation. Thus, the optimization problem is non-linear, with the parameter search occurring across a rugged landscape. We use covariance matrix adaptation (CMA-ES) [Han06] to determine the optimal controller parameters at each state s, performing outer loops of the algorithm until the policy converges.

The CMA-ES optimization is performed over one complete cycle of the state machine, and phases are not optimized in isolation. The coupling between phases is key to our approach, since evaluation of the success of a controller is determined not only by minimizing a set of objective terms for each phase independently, but by their performance as a sequence.

The desired poses for all phases of the controller are determined by minimizing the following composite objective function:

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

where 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.

Note that the terms of the composite objective function are scaled to bring the range of values for each objective term to within the same order of magnitude. These scaling factors are determined empirically. Also, we use the notation $\sum_{t \in T_i}$ to indicate a term that is accumulated over period T_i, with its value being sampled at each time step.

4.2. Approach

This 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

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

and ensures that the grasp quality at the end of the phase, Q_{T_0}, is at least the desired grasp quality $\tilde{Q}$, provided by the user. Note that the transitions occur as per the descriptions in the previous section. 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.

Details for how the grasp quality is computed are provided in Section 4.2.1.

4.2.1. Grasp Quality

We measure the grasp quality similarly to [KB87]. The grasp Jacobian $G \in \mathbb{R}^{6\times(mN)}$ is assembled for a $6D$ wrench space, accounting for N finger-object contacts and m basis vectors that provide a discretized Coulomb friction cone. Each wrench is written in the coordinate frame of the object. We compute the singular value decomposition $G = U\Sigma V^T$, and let our grasp quality measure Q be the smallest singular value of the grasp Jacobian. Because G has 6 rows, Q is equal to the sixth entry on the diagonal of Σ. Intuitively, the smallest singular value corresponds with the wrenching direction that is weakest. By avoiding poses where Q is equal or close to zero, singular grasp configurations may be avoided. In addition, the columns of U provide the axes of the wrench ellipsoid, and the axis with the smallest singular value provides a direction in which the least amount of force and torque is needed to break the grasp.

Since contacts are single-sided, as an additional check we ensure that the cone spanned by the contact force normals is at least $\frac{\pi}{2}$. If this is not the case then we assign a quality of $Q = 0$. We use $\frac{\pi}{2}$ instead of π because friction forces will allow an object to be grasped even in the absence of opposing normal contact forces.

Note that the most common way of determining grasp quality is by computing the *force closure* of the grasp Jacobian [MSL94]. However, this requires that we compute the convex hull in a $6D$ wrench space at each simulation step, resulting in protracted computing time. In practice, there was little difference in the motions generated using each metric.

4.3. Actuation

It is during the actuation phase that the agent makes most of its progress on the task. Wrenches acting on the object change the object's position and orientation such that progress is made toward the goal state, in a greedy sense. Based on this assumption, it's necessary that the predominate objective for this stage of the controller optimization is to minimize a task-based objective function. Specifically,

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

Here, we consider the norms unitless, since no scaling or normalization is used to combine the linear and angular components of L_T. For our experiments, the units of x are selected such that the typical range of the values

for the Euclidean norm approximately matches the range of quaternion logarithm norm (i.e., within an order of magnitude).

Note that $L_T \geq 0$, and the minimal value occurs when the goal orientation and position are reached. The value function should reflect the optimality of our task state. We choose $R(s) = -L_T$, giving a reward function that is non-positive for all states. This means that $V(s)$ is also non-positive, ensuring that the minimization of L_T corresponds to choosing actions that maximize the return of the reward and value function.

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

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

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. We compute the summed magnitude of contact forces affecting each finger 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, ε, a penalty proportional to the contact force is added. Conversely, if F_j for a participating falls below ε, a penalty is also added. The penalty is accumulated at each time step of the phase as

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

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 fingers break contact and move them in preparation for another approach. Any net wrench on the object during this phase is penalized in order to ensure that progress made during the actuation phase is not undone. This is done by looking at the magnitude of the net wrench applied to the object,

$$L_w = \left\| \sum_{i}^{N} {}^{b}_{i}Ad^{-T}\, {}^{i}f_i \right\|$$

where f_i is the ith contact force and ${}^{b}_{i}Ad$ is the adjoint matrix that transforms this contact force to a wrench expressed in the object frame (see [MSL94] for details on the adjoint transformation and expressing wrenches in different coordinate frames).

Simply minimizing the the net wrench affecting the object will not ensure good manipulation, since the hand must recover to a pose where it can make good progress during the following approach phase. This information is contained in the value function $V(s)$. Conveniently, like the reward function, the value function is also bounded above by 0, meaning that we can use its value directly as a penalty term in the controller optimization problem:

$$L_V = V(s) .$$

At the beginning of each inner loop of Algorithm 1, we reset the covariance matrix entries pertaining to the release phase. This is necessary because the value function changes at each iteration of the algorithm. Thus, the solution found for the release pose in previous iteration may no longer minimize $V(s)$.

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

$$L_2 = L_w + L_V .$$

4.5. Global Terms

In addition to the individual objectives for each phase, we add a global penalty term in order 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

$$L_g = \sum_{t \in T'} \|K(\tilde{q} - q(t))\| + \sum_{t \in T'} \sum_{j}^{M} b_j(t)$$

where $b_j(t)$ is equal to 1 if there is contact on the back of finger j at time t, or zero otherwise. This is determined by comparing the contact force with a vector defining the backhand direction on each finger segment. The term T' is the time to complete a full controller cycle, and this is accumulated over all phases.

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

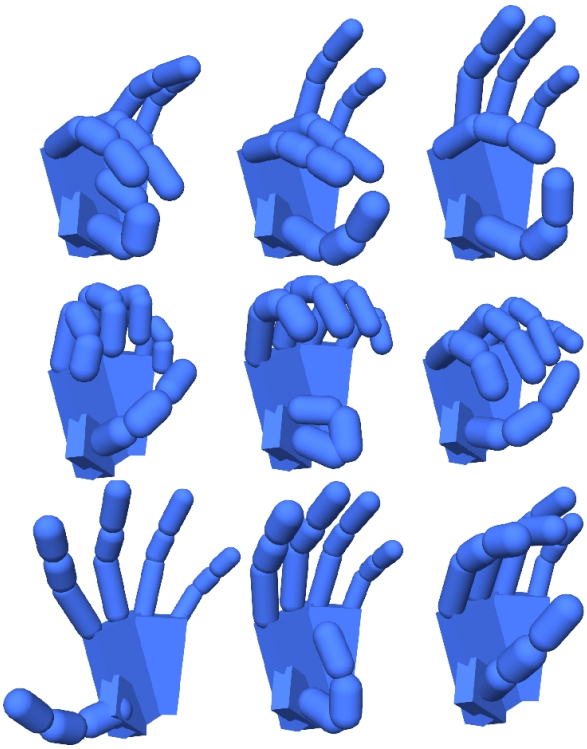

Figure 2: *Poses used to build a reduced basis for the control parameter search.*

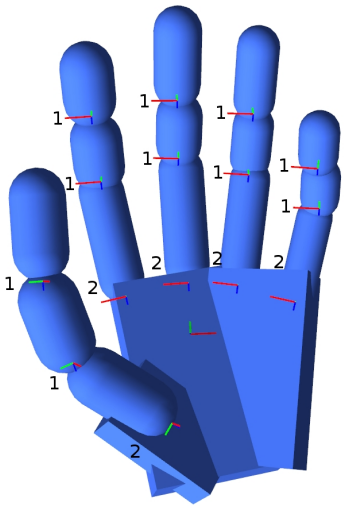

Figure 3: *Our hand model showing the associated number of degrees of freedom at each of the joints.*

choices which greatly reduce the computing time of the CMA-ES optimization required for controller selection.

One obvious choice is to use a parallelized version 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. [SFS98] that 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 10 to 20 grasp poses, from which the user may select some or all, a reduced pose basis is constructed. Figure 2 shows some of the poses we use for building our reduced pose space. Principal component analysis (PCA) is used to generate a set of orthogonal basis in which to perform the controller optimization. For the tasks shown in this paper, the parameter space is reduced from 20 degrees of freedom for each phase to just 8, giving a total of 24 parameters for each tri-phase controller. This significantly improves the performance of our optimization algorithm.

Finally, we use an implicit version of Equation 1 to generate joint torques, allowing us to take larger time steps while remaining stable. For the examples shown in this paper, $\Delta t = 17ms$.

5. Results and Discussion

In this section be provide some results for two examples: knob 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 geometry hand, with finger segments being actuated by joints with 1 and 2 degrees of freedom (see Figure 3), for a total of 20 joint angles.

The Vortex [Vor12] toolkit is used to simulate the forward dynamics, which include contact and gravity forces. All results were obtained using a 6-core Intel i7 3.2 GHz processor and running 12 simulation threads.

The maximum time per phase was set at 0.6 s. The CMA-ES optimization was performed using parameters $\sigma = 0.1$ for the knob turning example and $\sigma = 0.2$ for ball-in-hand; $\lambda = 60$ and a maximum iterations count of 2000 for both. Approximately 350 states were used to represent the control policy, giving learning times that ranged from 1 to 5 hours. However, we found it sufficient to stop the value iteration algorithm after just few iterations.

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.

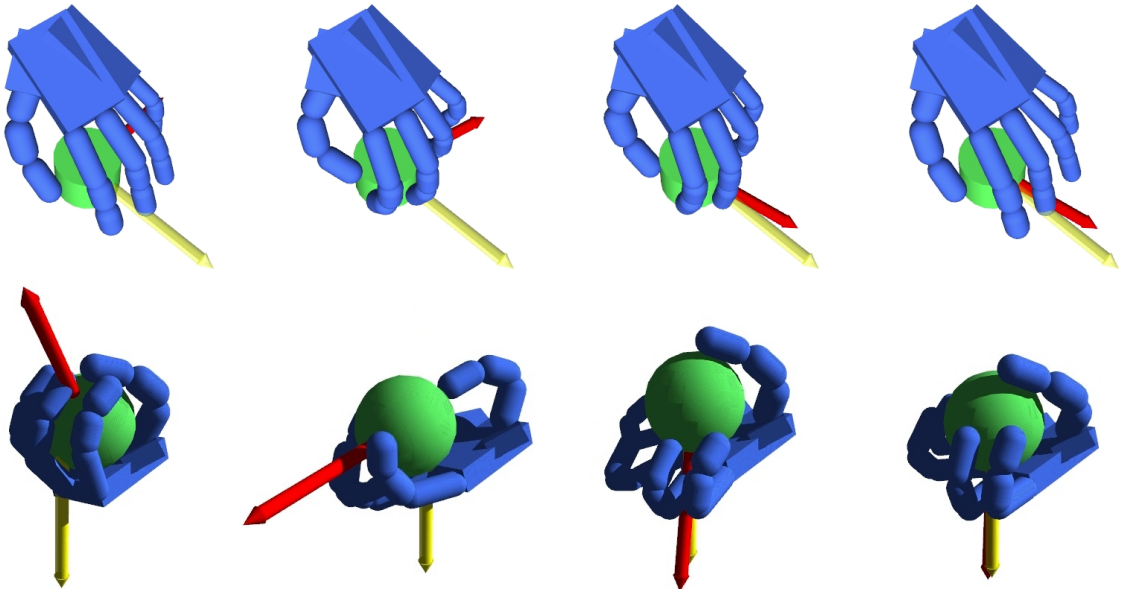

Figure 4: *Showing hand motion sequences for the knob turning example (top) and the ball-in-hand example (bottom). The desired (yellow) and current (red) configuration are shown.*

Figure 4 shows a temporal sequence of hand poses generated for our two examples. Complete animations are available in the accompanying video.

5.1. Knob Turning

For this 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

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

where $\tilde{\phi}_{hinge}$ and ϕ_{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 first row of Figure 4. On average, it takes about 5 to 10 seconds to perform the controller optimization for each state $s \in S$, with $\sigma = 0.1$ and $\lambda = 60$ used as CMA-ES parameters.

5.2. Ball-in-hand

The ball-in-hand involves re-orienting and re-positioning an unconstrained ball. The task state may be decomposed as $t = (x, \theta)$, where x and θ are the 3D position and orientation of the object in the hand frame, respectively.

There is special consideration for states where, at the end of the controller optimization, the ball is no longer in contact with the hand. For these states, we remove them from S.

On average, it takes about 30 to 60 seconds to perform the controller optimization for each state $s \in S$, with $\sigma = 0.2$ and $\lambda = 60$ used as CMA-ES parameters. The phase time is limited to 0.6 s.

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 doesn't assume a pre-defined trajectory and the focus is to achieve some 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 human finger gaiting.

As future work, we intend to investigate the use of linear feedback control to improve the robustness of the learned policies. Similar to the work in character locomotion [YLvdP07] and balance, we endeavour to make similar progress in the domain of grasping. By performing feedback control on components of the grasping state, such as grasp quality and net wrench, 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 aim to incorporate contact force and joint data, captured from human subjects, into the framework. This is key to understanding and building more complex control strategies that correspond to phase transitions occurring in actual manipulation. Also, by increasing the fidelity of the underlying dynamics simulation (e.g., by adding soft finger contacts), it may be possible to not only make the control problem simpler [JL11], but allow our controller architecture to make better transitions given the contact rich nature of the problem.

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

[CBvdP09] COROS S., BEAUDOIN P., VAN DE PANNE M.: Robust task-based control policies for physics-based characters. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 28*, 5 (2009), 170:1–170:9. 3

[FBJ06] FLANAGAN J. R., BOWMAN M. C., JOHANSSON R. S.: Control strategies in object manipulation tasks. *Current Opinion in Neurobiology 16* (2006), 1–10. 2

[Han06] HANSEN N.: The cma evolution strategy: A comparing review. *Towards a New Evolutionary Computation: Advanceson Estimation of Distribution Algorithms* (2006), 75–102. 5

[HG02] 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* (2002), vol. 2, pp. 1578–1584. 2

[HVNTH07] HAUSER K., VICTOR NG-THOW-HING H. G.-B.: Multi-modal motion planning for a humanoid manipulation task. In *Proceedings of International Symposium on Robotics Research (ISRR)* (2007). 2

[JL11] JAIN S., LIU C. K.: Controlling physics-based characters using soft contacts. In *Proceedings of the 2011 SIGGRAPH Asia Conference* (New York, NY, USA, 2011), SA '11, ACM, pp. 163:1–163:10. 9

[KB87] KLEIN C. A., BAHO B. E.: Dexterity measures for the design and control of kinematically redundant manipulators. *The International Journal of Robotics Research 6*, 2 (June 1987), 72–83. 5

[KP06] KRY P. G., PAI D. K.: Interaction capture and synthesis. *ACM Transactions on Graphics (Proc. of SIGGRAPH) 25*, 3 (2006), 872–880. 3

[Liu09] LIU C. K.: Dextrous manipulation from a grasping pose. *ACM Transactions on Graphics (Proc. of SIGGRAPH) 28*, 3 (2009), 59:1–59:6. 2

[LWB*10] LEE Y., WAMPLER K., BERNSTEIN G., POPOVIĆ J., POPOVIĆ Z.: Motion fields for interactive character animation. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia) 29*, 5 (December 2010), 138:1–138:8. 2

[MPT12] MORDATCH I., POPOVIC Z., TODOROV E.: Contact-invariant optimization for hand manipulation. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2012), pp. 137–144. 2

[MSL94] MURRAY R. M., SASTRY S. S., LI Z.: *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994. 5, 6

[OSC00] OKAMURA A. M., SMABY N., CUTKOSKY M. R.: An overview of dexterous manipulation. In *Proceedings of IEEE International Conference on Robotics and Automation* (2000), IEEE, pp. 255–262. 3

[PZ05] POLLARD N. S., ZORDAN V. B.: Physically based grasping control from example. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), pp. 311–318. 2

[SB98] SUTTON R. S., BARTO A. G.: *Reinforcement Learning I: Introduction*. The MIT Press, 1998. 4

[SFS98] SANTELLO M., FLANDERS M., SOECHTING J. F.: Postural hand synergies for tool use. *The Journal of Neuroscience 18*, 23 (December 1998), 2123–2142. 2, 7

[Vor12] VORTEX: *version 5.1.1*. CMLabs Simulations Inc., Montreal, QC, 2012. 7

[WFH09] WANG J. M., FLEET D. J., HERTZMANN A.: Optimizing walking controllers. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia) 28*, 5 (Dec. 2009), 168:1–168:8. 3

[YL12] YE Y., LIU C. K.: Synthesis of detailed hand manipulations using contact sampling. *ACM Transactions on Graphics (Proc. of SIGGRAPH) 31*, 4 (2012). 2

[YLvdP07] YIN K., LOKEN K., VAN DE PANNE M.: Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics (Proc. of SIGGRAPH) 26*, 3 (2007), Article 105. 8