An hybrid approach for robots learning folding tasks

Benjamin Balaguer BBALAGUER@UCMERCED.EDU
Stefano Carpin SCARPIN@UCMERCED.EDU
University of California, Merced, 5200 North Lake Rd, Merced, CA 95343

Abstract

We tackle the problem of using cooperative manipulators to perform towel folding tasks. Differently from other recent approaches, our method executes what we call a *momentum fold* - a swinging motion that exploits the dynamics of the manipulated object. We propose a new learning algorithm that combines imitation and reinforcement learning. Human demonstrations are used to reduce the search space of the reinforcement learning algorithm, which then converges quickly. The strengths of the algorithm come from its efficient processing, fast learning capabilities, absence of an object model, and applicability to other problems exhibiting temporally incoherent parameter spaces. Experiments were performed on a robotic platform, demonstrating the algorithm's capability.

1. Introduction

The popularity of service robotics has unveiled a multitude of novel challenges that researchers need to undertake before "a robot in every home" (Gates, 2004) can become a reality. One such challenge involves deformable object manipulation with cooperative manipulators. The inadequacy of deformable object models for robotic applications (Gibson & Balkcom, 1997), the absence of high-fidelity simulation tools for deformable objects, and the lack of literature on the subject are all factors delaying the development of robots capable of performing a variety of tasks involving flexible objects. The types of object we are interested in are highly deformable and we undertake the problem of folding rectangular towels using two manipulators working cooperatively.

Appearing in *Proceedings of the ICML Workshop on New Developments in Imitation Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

We exploit machine learning techniques to discard the requisite for a deformable object model, one of the major obstacles when working with flexible objects. Even though reinforcement learning has been shown to solve diverse tasks ranging from controlling a quadruped robotic dog (Theodorou et al., 2010) to playing the ball-in-a-cup game (Kober & Peters, 2009), flipping pancakes (Kormushev et al., 2010a), weightlifting (Rosenstein et al., 2006), and performing archery (Kormushev et al., 2010b), towel folding offers different research challenges: learning for two independent manipulators working cooperatively; exploiting a temporally incoherent parameter space (i.e. two or more successful folds can take a different amount of time to perform); dealing with an action-to-reward function composed of many-to-one mappings (i.e. there are many different ways to appropriately fold a towel). Due to the wide range of possible manipulator movements that yield correct folds, we combine human-to-robot imitation learning with reinforcement learning to not only converge faster to a solution, but also explore a wider range of the parameter space to find the action most replicable on the robotic platform. The contributions of this manuscript come from the human imitations combined with reinforcement learning, its efficient processing, fast learning capabilities, absence of a deformable object model, and applicability to other problems exhibiting temporally incoherent parameter spaces.

The rest of the paper is organized as follows. We start by describing, in Section 2, related works relevant to both our folding application and machine learning. A formal description of the problem we are addressing is given in Section 3, followed by our training data acquisition and proposed approach in Section 4 and 5, respectively. In Section 6, we present the experiments performed on our robotic platform. We conclude the paper with final remarks and future work in Section 7.

2. Related Work

The problem of using robotic manipulators to fold deformable objects has been studied before, although it has frequently relied on imperfect deformable object models or highly specialized robots. A review of deformable object models is beyond the scope of this paper, as is deformable one-dimensional object models, but interested readers can examine (Gibson & Balkcom, 1997) for more details. The robotics community has devised its own deformable object model where the object is decomposed into rigid links and foldable creases, resulting in a well-understood kinematic description. This simplified representation has been successfully applied to metal bending processes (Gupta et al., 1998), carton folding (Lu & Akella, 2000), paper craft (Song & Amato, 2004), and towel folding (Balaguer & Carpin, 2010). Each aforementioned application has drawbacks, however, in that they do not generalize well (Gupta et al., 1998; Balaguer & Carpin, 2010), or do not take into account the actuating robot when choosing a folding sequence using path planners like PRMs (Song & Amato, 2004) or RRTs (Lu & Akella, 2000). The kinematic representation is only suitable for deformable objects that retain their shape to a certain extent (e.g. metal) and cannot be used effectively for highly deformable objects. More practical robot systems have been designed for origami (Balkcon, 2004) and T-Shirt (Bell & Balkcom, 2010) folding. These robots are engineered for their specific tasks, however, and would be unsuitable for service robotics where one robot is tasked with highly heterogeneous assignments. The state of the art in folding comes from Abbeel et al. who have demonstrated the folding of towels (Maitin-Shepard et al., 2010) and clothes (Van Den Berg et al., 2010). Their work, however, depends on a parameterized shape model (Miller et al., 2011) created by a human and, as such, does not necessarily generalize to pieces of clothing that were not already parameterized. Additionally, the folding sequences are either pre-programmed or need to be entered by a user.

Typical off-the-book gradient-based policy learning approaches to learning (Sutton & Barto, 1998) have enjoyed only limited use in the robotics community, mainly due to the lack of adaptability to high-dimensional control and the manual parameter-tuning of the learning rate. Theodorou et al. realized these problems and implemented a reinforcement learning algorithm called Policy Improvements with Path Integrals (PI^2) (Theodorou et al., 2010). The authors' algorithm is capable of learning parameterized policies by using stochastic optimal control with path integrals. PI^2 does not require parameter tuning, although it requires an initial seed behavior that might be difficult to obtain, and works well with high-dimensional data, as exemplified by the learning of how to jump as far as possible on a quadruped robotic dog. Policy learning by Weighting Exploration with Returns (PoWER) (Kober & Peters, 2009) also solves the same problems seen in gradient-based policy learning and is thus far one of the leading algorithms when it comes to reinforcement learning for manipulation. Indeed, within a very short time, it has been applied to a great number of heterogeneous applications including the ball-in-a-cup task (Kober & Peters, 2009), flipping pancakes (Kormushev et al., 2010a), and performing archery (Kormushev et al., 2010b). PoWER is based on Expectation-Maximization, exploits a weighted sampling technique for exploration of the parameters space, and only requires an example motion to bootstrap the algorithm.

In previous works, the line between imitation and reinforcement learning is often blurred. Our definition of imitation learning, and the nomenclature used in this paper, follows that of Schaal et al. as “a complex set of mechanisms that map an observed movement of a teacher onto one's own movement apparatus” (Schaal et al., 2003). The distinct features between the two are that reinforcement learning exploits a trial-and-error methodology whereas imitation learning does not. Consequently, imitation learning requires multiple sample demonstrations in order to learn something. Related works on imitation learning differ greatly from our approach since we focus on imitation for the purpose of reinforcement learning. Interested readers should see (Schaal et al., 2003) for a good review of imitation learning techniques in robotics.

3. Problem Definition

The problem we aim to solve is to fold a towel symmetrically, where one half of the towel is folded on top of the other. Evidently, there are many ways that such a task can be performed and we choose to follow what we refer to as a *momentum fold*, where the force applied to grasping points on the towel is used to give momentum to the towel and lay half of it flat on the table (see Figure 1). We note that the momentum fold is used to make sure that half of the towel lays flat on the table and, as such, this is the motion we are trying to learn. Once half of the towel lays flat on the table, we can straightforwardly apply motion planning to finish the fold, in a similar fashion to (Balaguer & Carpin, 2010). While the majority of previous works utilizing reinforcement learning bootstrap their algorithm

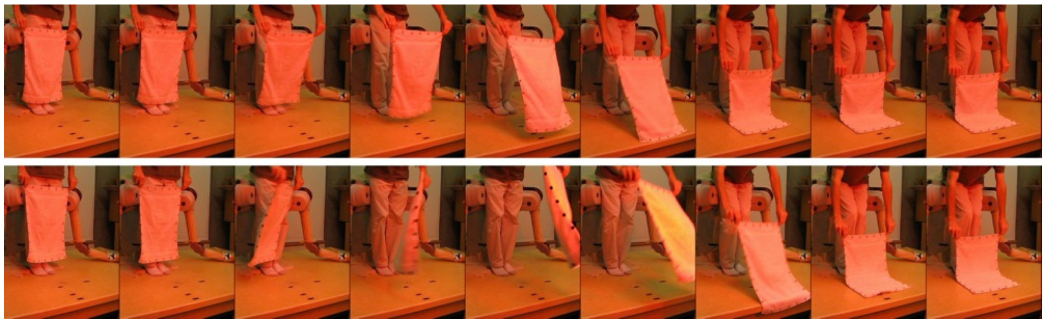


Figure 1. Two different, yet successful, momentum folds demonstrated by a human to the robot.

using kinesthetic teaching (i.e. having a human perform actions directly on a gravity-compensated robotic manipulator and recording the parameters from the robot), the fact that we are dealing with two manipulators renders this method impractical, if not impossible. Consequently, a human demonstrates an appropriate folding motion to the robot, two examples of which are shown in Figure 1. We assume that the towel can be picked by the robot and put into a starting position similar to the one in the first frame of Figure 1. This preliminary step has been previously solved by Towner et al. (Cusumano-Towner et al., 2011), so it is safe to assume this is a good starting configuration. It is worthwhile to note that we are faced with the problem of *temporally incoherent motion sequences.* This means that multiple - yet equally valid - folding motions will take different amounts of time to complete.

We designed and implemented a hybrid method that incorporates imitation and reinforcement learning. There are two reasons for incorporating imitation learning. First, from an algorithmic standpoint, exploiting knowledge acquired from human imitations can drastically reduce the parameter search space that the reinforcement learning algorithm has to explore, as will be shown in subsequent sections. Indeed, searching the parameter space based on previously-acquired rewards is both time-consuming and unnecessary and we harness the power of imitation learning to make the search more efficient. Second, from a more practical perspective, we cannot use kinesthetic teaching for folding applications. Since we have to use a human demonstrator and not all motions performed by a human will be replicable on a robot due to mechanical constraints, it is beneficial to acquire multiple demonstrations and learn from them.

4. Training Data for Imitation Learning

The demonstrator's goal is to acquire action-observation pairs that produce good folding motions by performing momentum folds, as exemplified in Figure 1. In order to facilitate the collection of actions and observations, we use a motion capture system along with a towel comprised of reflective markers that can be tracked. Formally, the i-th observation sequence O_i is comprised of the observation's time and of the Cartesian coordinates for the 28 markers at each time step of the motion. Similarly, the i-th action sequence, θ_i, contains the trajectories of the two manipulator (or human) control points.

We use data captured at 30 Hz, along with an average folding length of approximately 5 seconds, to dictate the number of time frames in our sequences ($30 \times 5 = 150$). Even though all examples have the same number of time frames, the time step between every example's time frame is different, thus accounting for temporal incoherence. The number of time frames (150), along with the data recorded for each observations and actions, defines the size of our vectors, namely $O_i \in \mathbb{R}^{12750}$ and $\theta_i \in \mathbb{R}^{1050}$. We collect 80 different folding sequences for our training data, which create our training data set where $O^t \in \mathbb{R}^{80\times 12750}$ and $\theta^t \in \mathbb{R}^{80\times 1050}$.

5. Proposed Approach

5.1. Reward Function

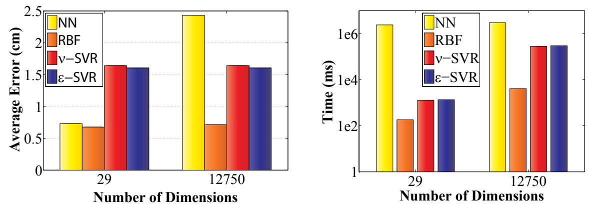
The reward function $R(O^t, O^c)$ computes the reward for a new observation O^c based on all the observations O^t acquired during our human training session. Its pseudo-code is shown in Algorithm 1. We use $O_i \in O^t$ (line 2) to indicate that we pick the i-th sample from O^t. We then extract the data points of the last time frame for the training sample and current observation (line 3 and 4). The Iterative Closest Point (ICP) algorithm is then applied to both data points, returning the average error (line 5) in millimeters. We repeat these steps for each training data sample and retain the smallest average error. Our reward is then the exponential function of the negative smallest average error in decimeter. The reward function effectively finds the best match between the current observation and any training observations, returning a pseudo-probability indicating how good the match is.

Algorithm 1 Computation of $R(O^t, O^c)$

```
1: $minAvgError \leftarrow 1000$
2: for all $O_i \in O^t$ do
3:     $Training \leftarrow LastFrame(O_i)$
4:     $Current \leftarrow LastFrame(O^c)$
5:     $AvgError \leftarrow \text{ICP}(Trainning, Current)$
6:     if $AvgError \leq minAvgError$ then
7:         $minAvgError \leftarrow AvgError$
   $R(O^t, O^c) = \exp(-minAvgError/100)$
```

5.2. Imitation Learning

We use imitation learning as a two-layer hierarchical approach to reduce the search space of the reinforcement learning algorithm. In the initial *exploratory layer* we use training data to let the robot execute a set of diverse folding sequences. Next, in the *expansion layer* we expand the search to motions similar to the best one that was found during the exploratory layer. In other words, we explore the action space based on human demonstrations, the best results of which will be used as seeds to the reinforcement learning algorithm. The exploratory layer's aim is to explore and find different motions in the trained action space, θ^t. Since we cannot execute all the trained motions, we apply k-means clustering, using $k = 10$, implicitly finding the most diverse set of 10 training motions. As a result, we have a set, $\theta^{Explore} = [\theta_1^{Explore}\theta_2^{Explore}\dots\theta_k^{Explore}]$ with $\theta_i^{Explore} \in \theta^t$. We let the robot execute each encoded trajectory, $\theta_i^{Explore}$, record its corresponding observation, $O_i^{Explore}$, and calculate the motion's reward using $R_i^{Explore} = R(O^t, O_i^{Explore})$. In the expansion layer, the action space is further explored starting with the best folding motion that the robot produced, $\theta_{Best}^{Explore}$, based on the collected rewards in the exploration layer. We train a learning algorithm using our training observations, O^t, and actions, θ^t, to learn the function $f : O_i \rightarrow \theta_i$. In other words, given an observation sequence O_i we want to find its corresponding action θ_i. We apply PCA to our observation data, O^t, which leads to a new observation data set, $\hat{O}^t$, projected in a lower-dimensional subspace where $\hat{O}^t \in \mathbb{R}^{80\times29}$. Learning is achieved by using $\hat{O}^t$ with Radial Basis Functions (RBF), since we empirically determined that it yields better accuracy and trained faster than Neural Networks (NN), ν Support Vector Regression (ν-SVR), and ε Support Vector Regression (ε-SVR), as shown in Figure 2. The average learning error is very low, 0.6767cm, which is much less than the mechanical inaccuracy of the manipulator we use.



(a) (b)

Figure 2. Figure 2(a) and 2(b) show the accuracy and training time for the NN, RBF, ν-SVR, and ε-SVR learning algorithms for both the dimensionally-reduced data (29) and the full data (12750). The reader shall note that we use log-scale for the Y-axis of Figure 2(b) because the values varied from more than 40 minutes to less than a second.

The RBF requires an observation as input and outputs the action matching that observation. Consequently, we need a process that generates a new observation, which is then fed to the RBF. To generate new observations, we fit a Gaussian distribution to the training data and sample from it. The process generates a new action, θ_s^{Expand}, using Algorithm 2. The time check on line 7 of the algorithm is performed to compensate for the temporal inconsistencies inherently encoded in our training data, where two valid folds can take a different amount of time to execute. We also use the motion's execution times to increase the likelihood that generated actions will be similar to $\theta_{Best}^{Explore}$. We run Algorithm 2 l times, resulting in $\theta^{Expand} = [\theta_1^{Expand}\theta_2^{Expand}\dots\theta_l^{Expand}]$. In similar fashion to the exploratory layer, we let the robot execute each encoded trajectory, θ_i^{Expand}, record its corresponding observation, O_i^{Expand}, and calculate the motion's reward using $R_i^{Expand} = R(O^t, O_i^{Expand})$.

Algorithm 2 Expand($\hat{O}^t$, $\theta_{Best}^{Explore}$, RBF, ϵ)

```
1: $n = \text{NumColumns}(\hat{O}^t)$    // $n = 29$ in our case
2: $\hat{O}^t \sim [\hat{O}_1^t\ \hat{O}_2^t \dots \hat{O}_n^t]$
3: $\mu = [E[\hat{O}_1^t]\ E[\hat{O}_2^t] \dots E[\hat{O}_n^t]]$
4: $\Sigma = [\text{Cov}(\hat{O}_i^t, \hat{O}_j^t)]_{i=1,2,\dots,n;j=1,2,\dots,n}$
5: repeat
6:     Sample $\hat{O}^s$ from $f(x)$
       s.t. $f(x) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} e^{\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)}$
7: until $|\text{Time}(\hat{O}^s)\text{-Time}(\theta_{Best}^{Explore})| \leq \varepsilon$
8: $\theta_s^{Expand} = \text{RBF}(\hat{O}^s)$
```

5.3. Reinforcement Learning

We finalize the algorithm using a modified version of the state-of-the-art reinforcement algorithm PoWER (Kober & Peters, 2009). The process is iterative and the action performed at time n is updated to produce a new action θ_{n+1} for the next rollout. The process is repeated until convergence, which we choose to be when the last three rollouts' rewards are within 0.1% of each other. PoWER's original update function does not work for our application, so we modify it to be

$$\theta^{RL}_{n+1} = \theta^{RL}_{n} + \left(\theta_{Top} - \theta^{RL}_{n}\right)\left[R(O^t, O_{Top}) - R(O^t, O_n)\right]$$

where Top is the index of the action with the best reward among $[\theta^{Explore}_{Best}\theta^{Expand}_{1}\dots\theta^{Expand}_{l}\theta^{RL}_{1}\dots\theta^{RL}_{n-1}]$. The update function is modified to account for two major issues that occur when using PoWER's unmodified update function. Firstly, we do not use importance sampling because different folds lying in different regions of the action space can yield similarly high rewards, resulting in poor and slow learning performance. Additionally, small potential time inconsistencies between multiple high reward actions can quickly lead the exploration into an action space region that is either not executable by the robot or does not resemble a folding motion. Secondly, we include the reward of the last rollout, $R(O^t, O_n)$, to influence the speed of the exploration. This method allows for a fine-grain search when the current action's reward is close to the best action's reward. Conversely, the further our current action's reward is from the best action's reward, the more space we cover during the update step.

6. Experimental Results

We present a real-world evaluation of the proposed algorithm on our robotic platform. The task of the robot is to symmetrically fold a thin hand-towel that is both light and highly susceptible to air flow resistance. The exploratory and expansion layers of the algorithm operate in constant time, since they always yield the same number of actions to be performed by the robot. Specifically, we always run the exploratory layer 10 times and the expansion layer 5 times. Once all 15 motions have been played back on the robot, the reinforcement learning iterates until convergence, which we define to be when the last three rewards are all within 0.001 of each other. Figure 3 shows the resulting rewards for a learning session. In the exploratory and expansion stages of the algorithm, the rewards oscillate, in no particular order since the actions are independent of each other, as the robot tries to find a good seed for the reinforcement learning algorithm. The reinforcement algorithm converges in 4 steps since high-quality motions were found during the exploratory and expansion stages of the algorithm. Figure 4 shows some rollouts performed by our robotic platform. We invite readers to go to our website[1] for videos, including the human demonstrations and trials, along with the data sets used in the paper.

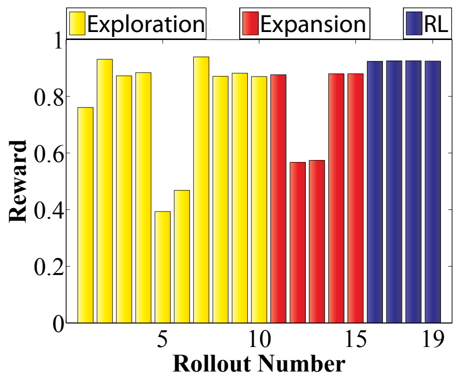


Figure 3. Rewards given to the robot for each rollout.

7. Conclusions and Future Work

We have shown that the combination of imitation and reinforcement learning provides a notable benefit to learning complex tasks. Indeed, once the exploratory and expansion steps are completed with the help of imitation learning, the reinforcement learning algorithm converges extremely quickly thanks to a very good starting seed. The approach is especially suited for tasks with different but equally-appropriate ways of solving them, where human-like motions are desirable, or where kinesthetic learning is impractical or impossible (e.g. when using two or more manipulators).

Acknowledgments

This research is partially supported by the National Science Foundation under grant BCS-0821766.

References

Balaguer, B. and Carpin, S. Motion planning for cooperative manipulators folding flexible planar objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3842–3847, 2010.

Balkcon, D. *Robotic Origami Folding*. PhD thesis, Carnegie Mellon University, 2004.

Bell, M. and Balkcom, D. Grasping non-stretchable cloth polygons. *International Journal of Robotics Research*, 29(6):775–784, 2010.

[1]http://robotics.ucmerced.edu/Robotics/ICML2011/

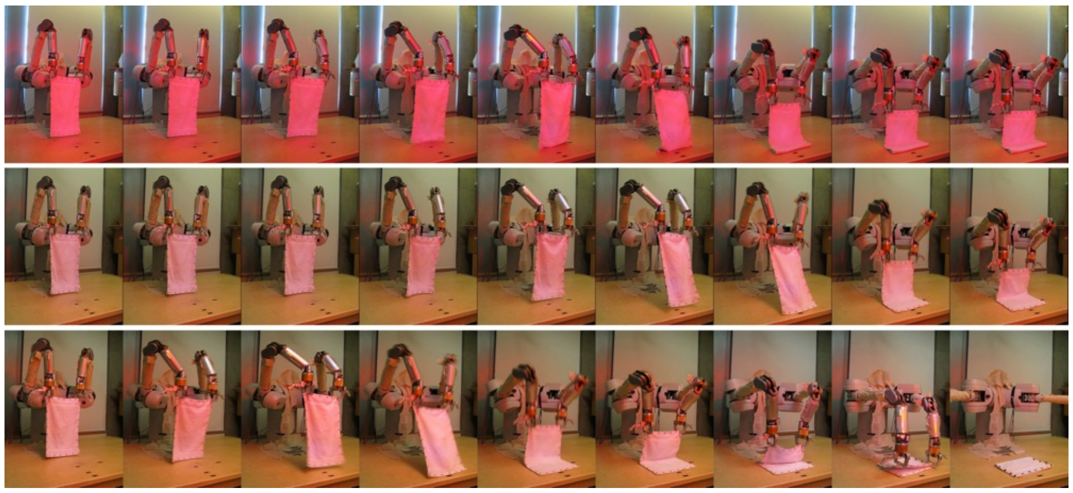


Figure 4. First 2 sequences received rewards of 0.57 and 0.94. The final motion is shown on the last sequence.

Cusumano-Towner, M., Singh, A., Miller, S., O'Brien, J., and Abbeel, P. Bringing clothing into desired configurations with limited perception. In *IEEE International Conference on Robotics and Automation*, pp. 3893–3900, 2011.

Gates, B. A robot in every home. *Scientific American Magazine*, December:58–65, 2006.

Gibson, S. and Mirtich, B. A survey of deformable modeling in computer graphics. Technical report, Mitsubishi Electric Research Laboratories, 1997.

Gupta, S., Bourne, D., Kim, K., and Krishnan, S. Automated process planning for robotic sheet metal bending operations. *Journal of Manufacturing Systems*, 17(5):338–360, 1998.

Kober, J. and Peters, J. Learning motor primitives for robotics. In *IEEE International Conference on Robotics and Automation*, pp. 2112–2118, 2009.

Kormushev, P., Calinon, S., and Caldwell, D. Robot motor skill coordination with em-based reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3232–3237, 2010.

Kormushev, P., Calinon, S., Saegusa, R., and Metta, G. Learning the skill of archery by a humanoid robot icub. In *IEEE/RAS International Conference on Humanoids Robots*, pp. 417–423, 2010.

Lu, L. and Akella, S. Folding cartons with fixtures: a motion planning approach. *IEEE Transactions on Robotics and Automation*, 16(4):346–356, 2000.

Maitin-Shepard, J., Cusumano-Towner, M., Lei, J., and Abbeel, P. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *IEEE International Conference on Robotics and Automation*, pp. 2308–2315, 2010.

Miller, S., Fritz, M., Darrell, T., and Abbeel, P. Parametrized shape models for clothing. In *IEEE International Conference on Robotics and Automation*, pp. 4861–4868, 2011.

Rosenstein, M., Barto, A., and Van Emmerik, R. Learning at the level of synergies for a robot weightlifter. *Robotics and Automation Systems*, 54(8):706–717, 2006.

Schaal, S., Ijspeert, A., and Billard, A. Computational approaches to motor learning by imitation. *Philosophical Transaction of the Royal Society of London: Series B, Biological Sciences*, 358(1431):537–547, 2003.

Song, G. and Amato, N. A motion planning approach to folding: From paper craft to protein folding. *IEEE Transactions on Robotics and Automation*, 20(1):60–71, 2004.

Sutton, R. and Barto, A. *Reinforcement Learning: an Introduction*. MIT Press, 1998.

Theodorou, E., Buchli, J., and Schaal, S. Reinforcement learning of motor skills in high dimensions: a path integral approach. In *IEEE International Conference on Robotics and Automation*, pp. 2397–2403, 2010.

Van Den Berg, J., Miller, S., Goldberg, K., and Abbeel, P. Gravity-based robotic cloth folding. In *International Workshop on "The Algorithmic Foundations of Robotics" at WAFR*, 2010.