Learning to Dress: Synthesizing Human Dressing Motion via Deep Reinforcement Learning

ALEXANDER CLEGG, The Georgia Institute of Technology
WENHAO YU, The Georgia Institute of Technology
JIE TAN, Google Brain
C. KAREN LIU, The Georgia Institute of Technology
GREG TURK, The Georgia Institute of Technology

Creating animation of a character putting on clothing is challenging due to the complex interactions between the character and the simulated garment. We take a model-free deep reinforcement learning (deepRL) approach to automatically discovering robust dressing control policies represented by neural networks. While deepRL has demonstrated several successes in learning complex motor skills, the data-demanding nature of the learning algorithms is at odds with the computationally costly cloth simulation required by the dressing task. This paper is the first to demonstrate that, with an appropriately designed input state space and a reward function, it is possible to incorporate cloth simulation in the deepRL framework to learn a robust dressing control policy. We introduce a salient representation of haptic information to guide the dressing process and utilize it in the reward function to provide learning signals during training. In order to learn a prolonged sequence of motion involving a diverse set of manipulation skills, such as grasping the edge of the shirt or pulling on a sleeve, we find it necessary to separate the dressing task into several subtasks and learn a control policy for each subtask. We introduce a *policy sequencing* algorithm that matches the distribution of output states from one task to the input distribution for the next task in the sequence. We have used this approach to produce character controllers for several dressing tasks: putting on a t-shirt, putting on a jacket, and robot-assisted dressing of a sleeve.

CCS Concepts: • **Computing methodologies** → **Reinforcement learning**; **Neural networks**; **Animation**;

Additional Key Words and Phrases: dressing, reinforcement learning, policy sequencing

ACM Reference Format:
Alexander Clegg, Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. 2018. Learning to Dress: Synthesizing Human Dressing Motion via Deep Reinforcement Learning. *ACM Trans. Graph.* 37, 6, Article 1 (November 2018), 10 pages. https://doi.org/10.1145/3272127.3275048

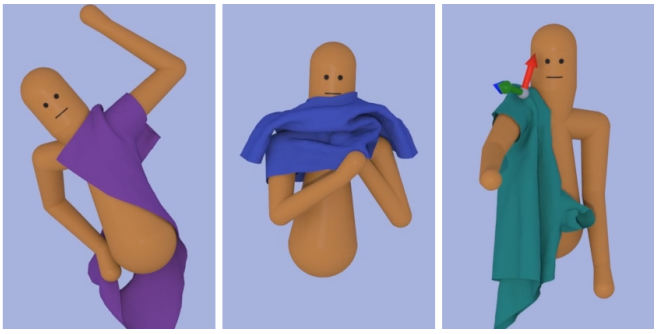

Fig. 1. A virtual human dresses a jacket, t-shirt and hospital gown.

1 INTRODUCTION

Putting on clothes is one of the most common tasks that each of us performs daily, yet animation of dressing is rare. We put our head and arms into a shirt or pull on pants without a thought to the complex nature of our interactions with the clothing. We may use one hand to hold a shirt open, reach our second hand into the sleeve, push our arm through the sleeve, and then reverse the roles of the hands to pull on the second sleeve. All the while, we are taking care to avoid getting our hand caught in the garment or tearing the clothing, often guided by our sense of touch. Animating this complex interplay between a character and clothing is challenging.

To tackle the challenges of animating dressing motions, we draw upon techniques in physics simulation and machine learning. We use a physics engine to simulate character motion and the motion of the cloth. To produce character motion, we use reinforcement learning to train a neural network that acts as the dressing control

Authors' addresses: Alexander Clegg, The Georgia Institute of Technology, Atlanta, Georgia, 30303, alexanderwclegg@gmail.com; Wenhao Yu, The Georgia Institute of Technology, Atlanta, Georgia, 30303, ; Jie Tan, Google Brain, ; C. Karen Liu, The Georgia Institute of Technology, Atlanta, Georgia, 30303, ; Greg Turk, The Georgia Institute of Technology, Atlanta, Georgia, 30303, .

0730-0301/2018/11-ART1 $15.00
https://doi.org/10.1145/3272127.3275048

policy for the character. As with any character control policy, the actions of the policy are decided based on the state of the character and the environment. Unlike many locomotion and manipulation motor skills, however, the motion of dressing does not follow a typical trajectory and the state of the environment (i.e. cloth) is highly dynamic and difficult to characterize. Rather than manually design a controller for a particular dressing behavior, we take a model-free deep reinforcement learning approach to automatically discovering robust dressing control policies represented by neural networks. Deep reinforcement learning largely removes the need for dimension reduction and feature selection of the state and action spaces and has led to several notable successes in recent years [Heess et al. 2017; Liu and Hodgins 2017; Mnih et al. 2015; Peng et al. 2018; Silver et al. 2016, 2017; Yu et al. 2018]. However, directly learning in a high-dimensional (possibly redundant) input state space is intractable for the dressing problem due to exceedingly expensive computation of cloth simulation in contact-rich scenes. As such, the rollout generation is the limiting factor that significantly impacts the design of the reward function, states, and actions, rendering the end-to-end learning approach impractical. This paper is the first to demonstrate that, with an appropriately designed input state space and a reward function, it is possible to incorporate cloth simulation in the reinforcement learning framework to learn a robust dressing control policy.

While dressing shares many attributes with reaching and grasping, it departs from the common manipulation tasks in that it heavily relies on the sense of touch to infer the progress and the goal of the task. Specifically, the character needs to learn to utilize haptic perception for two opposing tasks: *applying force* to traverse inside of the garment and *avoiding force* to prevent damage to the garment. We introduce a salient representation of haptic information to guide the dressing process—part of the state of the environment fed into the control policy is a haptic map that measures the contact forces between the character and the simulated cloth. In addition, the reward function used during training provides a strong learning signal responding to the haptic state and the action taken. For example, the cloth deformation is measured and penalized by the reward function to discourage the behavior of ripping the garment in order to achieve the dressing goal.

Another unique challenge about dressing is that it requires the character to perform a prolonged sequence of motion involving a diverse set of subtasks, such as grasping the front layer of a shirt, tucking a hand into the shirt opening, and pushing a hand through a sleeve. Learning a single control policy to achieve all these distinct motor skills and execute them sequentially is impractical due to the computational costs of cloth simulation. Consequently, we have to break down a full dressing sequence to subtasks and learn a control policy for each subtask. To ensure that the dressing task does not fail due to policy switches, we introduce a *policy sequencing* algorithm that matches the distribution of output states from one subtask to the input distribution for the next subtask in the sequence. Producing a successful policy for a single subtask requires hours of simulation and optimization. The benefit of this high computational cost is that the end result is not a single animation, but a character control policy that is capable of handling variations in the initial cloth position and character pose, as well as perturbations during the execution of control policies. The dressing control policy can thus be re-used to produce a variety of animations of the given dressing task.

2 RELATED WORK

Synthesizing complex motor skills for humanoid characters has received considerable interest in computer animation, robotics and more recently, machine learning. Researchers have developed methods to control animated characters to perform various tasks such as walking and running [de Lasa et al. 2010; Geijtenbeek et al. 2013; Hodgins et al. 1995; Yin et al. 2007; Yu et al. 2018], climbing [Jain et al. 2009; Naderi et al. 2017] and parkour [Heess et al. 2017; Liu and Hodgins 2017; Liu et al. 2016; Peng et al. 2018]. In most of this work, the character interacts with an environment consisting of rigid-bodies, for which highly efficient and stable simulation tools are available [Erez et al. 2015; Lee et al. 2018].

Being able to synthesize complex interactions with deformable objects such as dressing can greatly enrich the repertoire of simulated humanoid motor skills. However, simulating deformable objects is computationally expensive, susceptible to instability and greatly increases the degrees of freedom in the system, making it challenging to design controllers that interact with deformable objects. Some existing work resorts to approximations such as static and quasi-static simulation [Balaguer and Carpin 2010; Berenson 2013; Miller et al. 2012; Van Den Berg et al. 2010]. Though computationally more efficient, these methods cannot handle highly dynamic motion. Bai *et al.* developed a model predictive control based approach that predicts future states of the garment using linearized cloth dynamics and demonstrated synthesizing animations of cloth folding, wringing and picking up [Bai et al. 2016]. In their method, contacts between the character and cloth are assumed to be static, making it unsuitable to be applied to tasks involving many contact changes such as dressing. A few researchers have demonstrated self-dressing characters. Ho and Komura incorporated topological coordinates into keyframe interpolation and demonstrated a virtual character putting her arms through the sleeve holes of a large t-shirt [Ho and Komura 2009]. Using electric flux in path planning, Wang *et al.* demonstrated self-dressing of sock and a pair of shorts [Wang et al. 2013]. Miguel *et al.* developed a dressing system where an artist can author dressing motions such as putting on a scarf [Miguel et al. 2014]. These methods have demonstrated results for sub-tasks of self dressing such as stretching the arm outside the sleeve hole. However, it is unclear how to apply them to generate more complex motions that involve multiple stages of dressing, such as putting on a shirt from scratch. Clegg *et al.* described the dressing process as a sequence of primitive actions and developed a set of feedback controllers to chain the primitive actions together [Clegg et al. 2015]. They demonstrated dressing of a jacket, a pair of shorts, a robe, and a vest. Our method differs from theirs in several major aspects. First, we leverage the haptic information from the garments during dressing, which allows us to handle highly constrained tasks that involve utilizing deformation of the garment such as dressing a t-shirt. Second, we decompose the dressing tasks into coarser subtasks, which provides the user with a more intuitive way to create dressing animations. For example, we train a single controller to

put the right arm into the right sleeve, while the approach in [Clegg et al. 2015] would require three primitive actions. Third, the result of our method is a controller, rather than an animation, which can be re-run to produced a variety of motions and used as a starting point for re-training when simulation parameters change.

We formulate self-dressing as a series of Markov Decision Processes (MDPs) and solve them using Deep Reinforcement Learning (DRL). DRL has been demonstrated on complex humanoid motor skills with high dimensional state and action spaces [Lillicrap et al. 2015; Peng et al. 2018; Schulman et al. 2015b]. The main application of DRL has been rigid-body based control tasks due to the vast amount of data that is required to explore different parts of the state and action spaces. It has been conjectured that directly applying reinforcement learning to tasks in deformable environments such as dressing is computationally impractical, and existing work in this direction has been limited to simplified scenarios. For example, Tamei *et al.* applied reinforcement learning to dress a fixed mannequin with the arms of the mannequin initialized inside the sleeves of a t-shirt [Tamei et al. 2011]. They applied finite difference policy gradient to improve the control policy initialized using human demonstration. Clegg *et al.* applied reinforcement learning to train a modular haptic feedback controller in a rigid body scenario and showed that this controller can be transferred directly to a deformable environment [Clegg et al. 2017]. They demonstrated self-dressing results using an aggregation of the learned controller with guiding trajectories for navigating inside the garment. In contrast, our method directly applies reinforcement learning in a dressing scenario, where the agent has to learn not only to avoid tearing the garment, but also to dress itself.

3 OVERVIEW

We propose a reinforcement learning framework to train a virtual character to put on clothes in a physically simulated environment. Our approach splits the dressing task into a sequence of subtasks, and transitions between these tasks are guided by a state machine. For example, dressing a jacket consists of following four subtasks: pulling the sleeve onto the first arm, moving the second arm behind the back, scooping the second sleeve onto the arm, and finally returning the body to a rest position. For each subtask, we formulate a separate reinforcement learning problem to learn a control policy (Section 4). To ensure that these individual control policies can lead to a successful dressing sequence when executed sequentially, we introduce a policy sequencing algorithm (Section 5) that matches the initial state distribution of each subtask with the final state distribution of the previous subtask. The resulting control policies can be applied sequentially at interactive rates, transitioning via a state machine with subtask completion criteria, to produce a variety of successful dressing motions.

4 REINFORCEMENT LEARNING BACKGROUND

Each subtask of dressing is formulated as a partially observable Markov Decision Process (POMDP). The solution to each POMDP is a stochastic control policy $\pi : \mathcal{O} \times \mathcal{A} \mapsto [0, 1]$, which models the distribution of an action $\mathbf{a} \in \mathcal{A}$ conditioned on an observed state $\mathbf{o} \in \mathcal{O}$. A Markov Decision Process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, r, \rho, P_{sas'}, \gamma)$, where $\mathcal{S}$ is the state space; $\mathcal{A}$ is the action space; r is the reward function that maps a state $\mathbf{s}$ or/and an action $\mathbf{a}$ to a score; ρ is the distribution of the initial state $\mathbf{s}_0$; $P_{sas'}$ is the transition probability; and γ is the discount factor. Our goal is to optimize the policy π, represented as a neural network, such that the expected accumulated reward is maximized.

Although in simulation we can access the full physical state of the human and the garment (i.e. the joint positions and velocities of the human and the vertex positions and velocities of the garment), defined as $\mathbf{s} \in \mathcal{S}$, we formulate a MDP that is only *partially observable* because humans do not have direct perception of the full state of the world and themselves. In the case of dressing, humans have limited perception of the state of the garment outside of haptic and visual observations. Giving our human model full observability might result in motions that real humans are incapable of performing. As such, we design a compact observation space, $\mathcal{O}$, which is a subspace of the state space $\mathcal{S}$ (Section 6). While the input to the policy, $\mathbf{o}$, should be restricted by the capability of human perception, the input to the reward function, $\mathbf{s}$, can take advantage of the full state of the simulated world. We propose a novel reward function (Section 7) that quantifies dressing progress.

All subtasks share the same action space, while the reward terms, their weights, and the observation features vary by subtask (see Table 1).

5 SEQUENCING CONTROL POLICIES

Since each subtask is formulated as a separate MDP, directly executing their policies sequentially tends to fail at transition points. We introduce a policy sequencing algorithm that ensures successful transitions between policies to achieve a task that requires a prolonged sequence. The algorithm consists of two passes and is summarize in Algorithm 1.

The first pass solves for an intermediate policy, $\tilde{\pi}^i$, for each subtask MDP $\mathcal{M}^i$. In addition to the original reward function r^i, the policy also tries to minimize the deviation from a reference pose derived from the mean ($\boldsymbol{\mu}^{i+1}$) of the initial state distribution of the subsequent MDP (Line 2). All the intermediate policies are trained separately in parallel on a narrow initial state distribution in order to encourage rapid convergence to a successful policy. In the second pass, we revisit every MDP sequentially. We first widen the initial state distribution of the first task ρ^1 by multiplying the covariant matrix Σ^1 with a scaling factor α (Line 5). We then solve the policy π^1 from the intermediate policy $\tilde{\pi}^1$ as an initial guess (Line 6). For the subsequent MDP's ($\mathcal{M}^2, \cdots \mathcal{M}^N$), we sequentially widen the initial state distributions by matching the final state distribution generated by the policy of the previous MDP π^{i-1} (Line 8-10) and solve for the policy π^i (Line 11). Figure 2 illustrates how the initial state distributions are widened through the two-pass process.

6 OBSERVATION SPACE

The full state space of dressing tasks is typically high-dimensional which leads to an extremely large policy network with hundreds of thousands variables. As such, directly optimizing a policy with the full state as input requires a large amount of training examples (rollouts) from a costly cloth simulator, rendering the deepRL

Algorithm 1 Policy sequencing algorithm

for each MDP $\mathcal{M}^i, i = 1 \cdots N-1$ **do**
  $r^i \leftarrow r^i + w \cdot r_r(\mathbf{s}; \boldsymbol{\mu}^{i+1})$
  Solve $\tilde{\pi}^i$ for $\mathcal{M}^i$
Solve $\tilde{\pi}^N$ for $\mathcal{M}^N$
$\rho^1 \leftarrow \mathcal{N}(\boldsymbol{\mu}^1, \alpha\Sigma^1)$
Solve π^1 for $\mathcal{M}^1$ from intermediate policy $\tilde{\pi}^1$
for each MDP $\mathcal{M}^i, i = 2 \cdots N$ **do**
  Generate trajectories from π^{i-1}
  Collect final states of trajectories in $\mathcal{D}$
  $\rho^i \leftarrow$ Gaussian that best fits $\mathcal{D}$
  Solve π^i for $\mathcal{M}^i$ from intermediate policy $\tilde{\pi}^i$

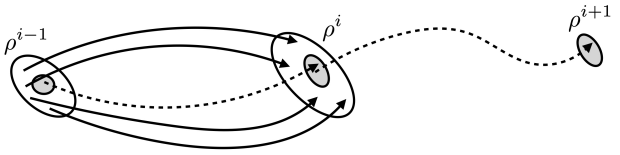

Fig. 2. The policy sequencing algorithm consists of two passes. The first pass solves for intermediate policies in parallel on narrow initial state distributions, shown in gray. Example trajectories for those intermediate policies are illustrated as dashed curves. In the second pass, for each MDP $\mathcal{M}^i$, the initial state distribution ρ^i is widened by matching the final states of trajectories (shown as solid curves) generated by the policy of the previous MDP π^{i-1}. The policy π^i is then solved again from the intermediate policy $\tilde{\pi}^i$ as an initial guess.

approach intractable. Additionally, we are interested in modeling what humans are capable of, and therefore we would like to limit policy input to the kind of information a human could reasonably be expected to have. For these reasons, we formulate a compact observation space that is tailored for dressing tasks. The observation space includes the human's joint angles, garment feature locations, haptics, surface information and a task vector.

$$O = [O_p, O_f, O_h, O_s, O_t] \tag{1}$$

With carefully picked components, our observation is a 163 dimensional vector.

6.1 Proprioception

Following other recent work on reinforcement learning of motor control policies, we provide the policy with a proprioceptive input feature,

$$O_p = [\cos(\mathbf{q}(\mathbf{s})), \sin(\mathbf{q}(\mathbf{s})), \dot{\mathbf{q}}(\mathbf{s})] \tag{2}$$

where $\mathbf{q}(\mathbf{s})$ is the vector of joint angles describing the human pose at state $\mathbf{s}$. The human model in this work contains 22 degrees of freedom, all of which are actuated.

6.2 Garment feature location

The current location of a garment feature (e.g. , a sleeve opening) and its relative position to the end effector are important information humans use for dressing, because the typical garment has more than one feature which could be dressed. We provide the policy with the world position of the centroid, $\mathbf{c}$, of the garment feature polygon, $\mathcal{P}$, and the displacement vector between the end effector and the centroid:

$$O_f = [\mathbf{c}(\mathbf{s}), \mathbf{p}_0(\mathbf{s}) - \mathbf{c}(\mathbf{s})] \tag{3}$$

6.3 Haptics

Humans rely on haptic sensing during dressing to avoid damage to clothes and to minimize discomfort. Similar to Clegg *et al.* [Clegg et al. 2017], we distribute a series of haptic sensors along the medial axis of the body nodes of the human character. We then construct a haptic feature vector by aggregating all contact forces between the human's body and the cloth. We do so by summing each contact into the closest haptic sensor, resulting in a haptic feature:

$$O_h = [\mathbf{f}_0(\mathbf{s}), \cdots, \mathbf{f}_n(\mathbf{s})] \tag{4}$$

where $\mathbf{f}_i(\mathbf{s})$ is the 3-dimensional accumulated force vector of the i-th sensor and $n = 21$ for our human model, placed as shown in Figure 3. This haptic information encodes the location of contacts, as well as the magnitude and direction of the contact forces, which are essential to respecting the garment strain constraints.

6.4 Signed surface

When putting on a shirt, humans typically stretch their arms through the sleeve while making contact with the inside of the garment. While humans seem to be able to distinguish the inner surface of a garment from the outer surface, this poses a great challenge for the simulated human with no vision and low-resolution tactile sensors. As such, we provide the policy with a surface sign for each haptic sensor i, that differentiates the contact between the inner and outer surfaces of the garment. We consider that in many common situations this information could be acquired by a combination of vision and haptic perception.

$$s_i = \begin{cases} 0 & \text{if not in contact,} \\ 1 & \text{if inner surface contact,} \\ -1 & \text{if outer surface contact.} \end{cases} \tag{5}$$

For each contact point binned into sensor i, we assign it -1 if its contact force is opposing the outward facing vertex normal at the contact location. Otherwise, we assign it 1. If the sum of the assigned values for sensor i is positive, we consider that the sensor is in contact with the surface from inside. It the sum is negative, the sensor is in contact with the garment from outside. Each of the 21 sensors for our human model provides one signed surface value to the observation space:

$$O_s = [s_0, \cdots, s_n], \tag{6}$$

6.5 Task vector

Since our network architecture has no memory (no recurrency), the high level planning and order of events required to robustly complete the dressing task present significant challenges, making recovery from mistakes nearly impossible. For example, in the case that the end effector misses the garment on the first attempt, an optimal policy must reduce its immediate reward by backtracking

in order to insert the limb into the feature, thus achieving higher total reward in a later state.

To overcome these challenges, we provide the policy with a unit length task vector that suggests a direction for the end effector to move based on dressing progress. This task vector mimics a human's intuitive understanding of how to make progress on the dressing task based on prior knowledge about garment geometry and unique textural hints about garment features and the contact surface such as layers and seams which can not be obtained efficiently with existing cloth simulation techniques. This vector directs the end effector toward the garment feature until it makes contact with the garment and through the garment feature once it contains the limb. When the end effector is in contact with the garment but the limb is not yet contained, the task vector guides the end effector toward the feature along the gradient of the geodesic field defined on the garment. This allows the policy to backtrack from mistakes and to navigate the folds and occlusions between limb and feature, akin to the method that humans use to navigate a garment with touch alone.

The task vector depends on geodesic information when the limb is in contact with the garment but has not yet entered the garment feature. We compute the gradient of the geodesic field $g(\mathbf{v})$ evaluated at $v^* = \text{argmin}_{\mathbf{v}\in\mathcal{V}_c}\, g(\mathbf{v})$, the vertex that is in contact with the limb and is the closest to the garment feature:

$$O_t = \begin{cases} \frac{\mathbf{c}-\mathbf{p}_0}{||\mathbf{c}-\mathbf{p}_0||} & \text{if not in contact,} \\ \mathbf{n}_{plane} & \text{if limb contained,} \\ \nabla g(\mathbf{v})|_{\mathbf{v}=\mathbf{v}^*} & \text{otherwise.} \end{cases} \tag{7}$$

where $\mathbf{n}_{plane}$ is the normal of the feature plane and limb containment is determined using the approach detailed in section 7.1.

7 REWARD FUNCTION

The reward is a numerical measure of how close the character is to completing its current task, as a function of the state of the character and the garment. In this work, state $\mathbf{s}$ consists of the joint angles and velocities of the human model, the vertex positions of the garment, contact information from the previous simulation step and the values of a precomputed geodesic field on the garment. A good reward function is important to the success of reinforcement learning. Designing a reward function for the dressing task is nontrivial due to the difficulties of quantitatively defining dressing progress and balancing conflicting objectives, such as exploiting contact to push limbs through the garment while preventing large amounts of force that could tear apart the cloth. Additionally, care must be taken to ensure that reward is not too sparse for off-policy exploration to reach high reward states. For these reasons, we propose a novel reward function:

$$r(\mathbf{s}) = w_1 \cdot r_p(\mathbf{s}) + w_2 \cdot r_d(\mathbf{s}) + w_3 \cdot r_g(\mathbf{s}) + w_4 \cdot r_t(\mathbf{s}) + w_5 \cdot r_r(\mathbf{s}) \tag{8}$$

where r_p is the progress reward, r_d is the deformation penalty, r_g is the geodesic reward, r_t rewards end effector motion in the direction of the task vector, and r_r attracts the character to a target position. The scalar weights w's are determined empirically for each subtask. Please see Table 1 for details.

Table 1. Reward Weights for subtask controllers

Task	Sub-task	w_1	w_2	w_3	w_4	w_5	α_1	α_2
T-Shirt	Grip	-	400	-	-	1	100	-
	Tuck 1	-	20	2	-	1	-	1
	Sleeve 1	15	5	4	50	1	-	-
	Re-grip	-	15	-	-	5	20	-
	Tuck 2	-	10	2	-	0.5	-	20
	Sleeve 2	10	5	4	50	1	-	-
	Finish	-	10	-	-	1	-	-
Jacket	Sleeve 1	10	5	2	50	1	-	-
	Transition	-	5	-	-	2	0.5	-
	Sleeve 2	10	5	2	50	1	-	-
Assistive	Sleeve	10	5	2	50	1	-	-

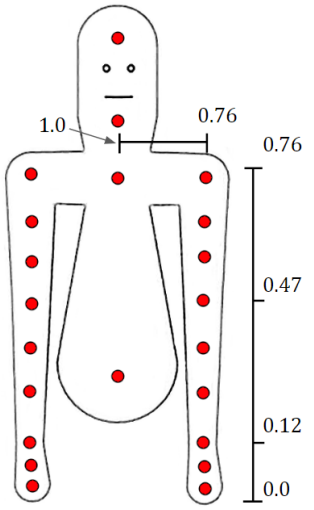

Fig. 3. Haptic sensor placement on the character model (red), and limb progress function values for garment feature containment of a limb at various depths.

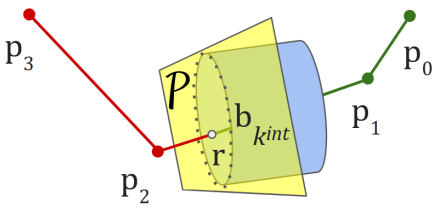

Fig. 4. Visualization of the containment computation for a limb ($p_{0\sim3}$) partially inserted into a garment (blue cylinder). Here $k_{int} = 2$

7.1 Progress Reward

Without a continuous metric for progress, the dressing task becomes a sparse reward problem, making it challenging and data-inefficient for standard policy gradient methods. The progress reward metric measures the extend to which a limb is dressed at a state, $\mathbf{s}$. Figure 3 shows progress measured along the limb being dressed in our experiments.

We first define a garment feature $\mathcal{F}$ as a set of indices to cloth vertices that best approximate a cloth structure of interest (e.g. the end of a sleeve or the waistband of a pair of pants) [Clegg et al. 2015]. In our application, these features are often closed loops of vertices. For clarity of the exposition, we also define the world position of each joint of the limb as $\mathbf{p}_i$, where $i = 0, \cdots, m$. Unlike conventional joint index scheme, we index joints from distal to proximal. That is, $\mathbf{p}_m$ is the most proximal joint on the limb and $\mathbf{p}_1$ is the most distal. $\mathbf{p}_0$ is a dummy joint that refers the tip of the end-effector. All joint positions can be derived from the current state $\mathbf{s}$.

To measure progress, we first need to determine whether the intended limb is contained by the feature loop. If so, we reward the depth of the insertion. If not, we penalize the distance from the end-effector to the feature loop.

The containment of the limb in garment feature, $\mathcal{F}$, is visualized in Figure 4 and computed as follows. First, we compute the best fit plane to the feature loop and project the feature vertices in $\mathcal{F}$ onto the plane, resulting in a 2D polygon $\mathcal{P}$. We then check whether a bone segment $\mathbf{b}_i$ between $\mathbf{p}_i$ and $\mathbf{p}_{i-1}$ intersects $\mathcal{P}$:

$$c_i(\mathbf{s}, \mathcal{F}) = \begin{cases} 1 & \text{if } \mathbf{b}_i \cap \mathcal{P} \neq \emptyset \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Starting from the most distal bone segment, we check c_i for each bone until the first encounter of $c_i = 1$. We then define k_{int} as the index to the bone that intersects with the garment loop. If no bone intersects with the polygon, we set $k_{int} = 0$. The depth of containment is then defined as

$$l(\mathbf{s}, \mathcal{F}) = \|\mathbf{p}_{k_{int}-1} - \mathbf{r}\| + \sum_{i=1}^{k_{int}-1} \|\mathbf{b}_i\|, \tag{10}$$

where $\mathbf{r} = \mathbf{b}_{k_{int}} \cap \mathcal{P}$ and $\|\mathbf{b}_i\|$ is the length of the bone. The reward function can be defined as:

$$r_p(\mathbf{s}) = \begin{cases} l(\mathbf{s}, \mathcal{F}) & \text{if } k_{int} \neq 0 \\ -\|\mathbf{c} - \mathbf{p}_0\| & \text{otherwise.} \end{cases} \tag{11}$$

where $\mathbf{c}$ is the centroid of the polygon $\mathcal{P}$.

7.2 Deformation Penalty

In simulated dressing, the garment can be torn apart when it is deformed beyond reasonable strain limits by the movement of the character. The reward function penalizes such undesired cloth deformation. Since we represent the cloth as a triangle mesh, we first define the deformation of a single cloth triangle, indexed by i, as the ratio of its current area, a, to its rest area, a_{rest},

$$d_i(\mathbf{s}) = \frac{a}{a_{rest}} \tag{12}$$

We then use the largest deformation across all triangles to calculate the deformation penalty:

$$r_d(\mathbf{s}) = -\frac{\tanh(w_{scale}(\max_i d_i(\mathbf{s}) - w_{mid}))}{2} - \frac{1}{2} \tag{13}$$

where w_{mid} defines the midpoint of the deformation penalty range. w_{scale} scales the slope and upper/lower limits of the deformation penalty function. This formulation for deformation penalty results in little to no penalty for small deformations in order to encourage the use of contact for dressing, while providing a smoothly increasing penalty for larger deformations and a guaranteed maximum penalty which is important for reward tuning. We choose $w_{mid} = 25$ and $w_{scale} = 0.14$ in all of our experiments, effectively ignoring deformation below 15 and capping the penalty at deformations exceeding 35. Only the largest deformation is considered due to the highly local nature of the collision resolution approach applied in a position based dynamics simulator. This approach constrains the location of mesh vertices and virtual spherical particles embedded in each mesh face such that they end each simulation cycle outside of the character's limbs. Therefore, typical limb penetration failures involve a limb penetrating one or more cloth faces causing extreme deformation while outside of the immediate neighborhood deformation remains low. An alternative choice would be to penalize the sum of deformations across the full garment. However, doing so would result in equivalent penalty for the cases of small deformation across the entire garment and high deformation of only one or few mesh faces. In reality, humans exploit the deformation of the clothing when dressing some garments, and thus small deformation should not be penalized heavily. For this reason, we choose to use tanh to cap the penalty. Note that an alternative approach to penalizing deformation would be to terminate the simulation at any state in which the deformation is above a threshold. However, we observe that this strategy tends to punish exploration in the early stages of learning and results in overly-cautious policies.

7.3 Geodesic Contact

When the limb is far from the garment feature, $\mathcal{F}$, $r_p(\mathbf{s})$ is the negative Cartesian distance to the loop centroid. This term alone is not enough to encourage successful policy behavior. The garment feature can be occluded by layers of cloth which must be moved aside in order to reach it and complete the dressing task. To guide the end effector through folds inside the cloth, we utilize the geodesic distance on the cloth to the garment feature. We first calculate a field $g(\mathbf{v})$ that maps a vertex $\mathbf{v}$ on the cloth mesh to the normalized geodesic distance between $\mathbf{v}$ and the garment feature (i.e. vertices in $\mathcal{F}$ have distance $g(\mathbf{v}) = 0$ while those farthest from the feature have distance $g(\mathbf{v}) = 1$). Among all the cloth vertices that are in contact with the end-effector $\mathcal{V}_c$, we select the one with the smallest geodesic distance:

$$g^*(\mathbf{s}) = \min_{\mathbf{v} \in \mathcal{V}_c(\mathbf{s})} g(\mathbf{v}). \tag{14}$$

and define $r_g(\mathbf{s}) = 1 - g^*(\mathbf{s})$.

If the end effector is not touching the cloth, the reward is zero. This encourages contact with the garment and maximizes the number of haptic observations. Additionally, we only reward end effector contact with the cloth before the limb has entered the garment feature. Therefore, we return the maximum value if any part of the limb is contained by the feature.

$$r_g(\mathbf{s}) = \begin{cases} 0 & \text{no contact,} \\ 1 & \text{limb contained,} \\ 1 - g^*(\mathbf{s}) & \text{otherwise.} \end{cases} \quad (15)$$

7.4 Task Vector Displacement

The task vector described in section 6.5 hints at a desirable direction for the end effector to move in order to make progress on a limb-insertion subtask. While enough exploration may allow a control policy to learn this connection from its relationship to other reward terms, rewarding end effector motion relative to the shoulder in the direction of the task vector can serve as a straightforward way to encourage progress when other reward signals are weak or ambiguous. This may also aid the control policy in forming a relationship between increased reward and following the task vector provided in observation. Given the task vector observation O_t, we define this reward as:

$$r_t(\mathbf{s}) = (\mathbf{p}_t(\mathbf{s}) - \mathbf{p}_{t-1}(\mathbf{s})) \cdot O_t \quad (16)$$

where t is the current simulation step and $\mathbf{p} = \mathbf{p}_0 - \mathbf{p}_3$ is the end-effector position relative to the shoulder position.

7.5 Target Position

The final term in the reward function attracts the character to a target position which can be described by a goal pose, a desired end-effector location, or a geometric relationship between body parts:

$$r_r(\mathbf{s}) = -\|\mathbf{q}(\mathbf{s}) - \bar{\mathbf{q}}\| - \alpha_1 f_{grip}(\mathbf{s}) + \alpha_2 f_{tuck}(\mathbf{s}) \quad (17)$$

where $\mathbf{q}(\mathbf{s})$ is the current pose of the character and $\bar{\mathbf{q}}$ is the goal pose, derived from the mean of the next control policy's initial state distribution ($\boldsymbol{\mu}^{i+1}$). Optionally, we add an end-effector target penalty for the subtasks involving gripping:

$$f_{grip}(\mathbf{s}) = \|(\mathbf{p}_0(\mathbf{s}) - \bar{\mathbf{p}}_0(\mathbf{s}))\| \quad (18)$$

where $\bar{\mathbf{p}}_0(\mathbf{s})$ is a target position such as the current location of the grip feature during the grip and re-grip subtasks. For tucking subtasks, we define a binary reward for intersection of any limb segment with a triangle defined by one corner at the gripping end effector and the other two at the shoulders:

$$f_{tuck}(\mathbf{s}) = \begin{cases} 1 & \text{limb not contained,} \\ 0 & \text{otherwise.} \end{cases} \quad (19)$$

This term simply encourages the limb to remain between the grip point and the body while seeking to satisfy other reward terms and is used exclusively in tucking subtasks. The parameters α_1 and α_2 are listed in Table 1.

8 RESULTS

We choose to examine two different self-dressing tasks: dressing a t-shirt and dressing a jacket. Both of them require cooperation of two arms and multiple subtasks. In addition, we also evaluate our method on an assistive dressing task. Please watch the accompanying video to see animation of these tasks.

We simulate the character dynamics using DART [Lee et al. 2018], and cloth dynamics with NVIDIA PhysX, which implements position-based dynamics [Macklin et al. 2014]. [Clegg et al. 2017] investigated neural network control of a sphere that is navigating a cloth tube via translational motion. Using the same friction model as that which is used in this work, they found that friction coefficient values greater than 0.6 resulted in reduced performance as the cloth began sticking to the sphere during tangential motion. Based on these results, we chose a friction value of 0.5 out of a maximum of 1.0 for all examples. To more accurately model the complexity of actual human upper body motion, we apply a recently developed data driven joint limiting approach in place of traditional joint limits for the shoulders and elbows [Jiang and Liu 2017]. Our character model is displayed in Figure 3 and consists of 22 degrees of freedom. The root of the character model is fixed to the origin such that no translation is permitted.

All policies are represented by fully connected neural networks consisting of 2 hidden layers of 64 nodes each and tanh activations with a final linear output layer. We optimize the control policies using Trust Region Policy Optimization (TRPO) [Schulman et al. 2015a] implemented in OpenAI rllab [Duan et al. 2016]. Policies were trained for $1,000 - 3,000$ TRPO iterations with $10,000 - 20,000$ simulation steps per iteration depending on the complexity of the subtask. We use a simulation step of 0.01s and query the control policy every 4 steps for the t-shirt sequence and every 5 steps for the jacket and assistive tasks. Each subtask spans $4 - 6$s. Our simulation environment runs at interactive rates, but not realtime rates, producing a sample in about 0.05 seconds. As such, training a single subtask requires about 24 hours of simulation time on the 36 vCPU compute nodes we used.

8.1 T-Shirt

In this task, a t-shirt is initialized on the character's shoulders with the character's neck contained within the collar. To randomize the initial garment state, we apply a random impulse force with fixed magnitude to all the garment vertices at the beginning of the simulation. We allow the garment to settle for 1s before the character begins to move.

The first control policy completes the task of moving the right end effector into gripping range of the specified grip feature. The policy attempts to match a given position and orientation target in the garment feature space. Once the error threshold is reached, control transitions to an alignment policy designed to "tuck" the left end effector and forearm under the waist feature of the garment in preparation for dressing the arm. This policy attempts to contain the arm within the triangle formed by the gripping hand and the shoulders. This heuristic approximates the opening of the garment waist feature. In addition, this policy is rewarded for contact with the garment interior and penalized for geodesic distance from a selected point on the interior of the garment. Once interior contact is detected, and the arm is within the heuristic triangle, control is transitioned to the left sleeve dressing controller which attempts to minimize the end effector contact geodesic distance from the end feature of the sleeve and maximize the containment depth of the arm within the sleeve entrance feature. A task vector is provided which indicates the direction the end effector should move to decrease its contact geodesic distance (or points to the garment feature if not

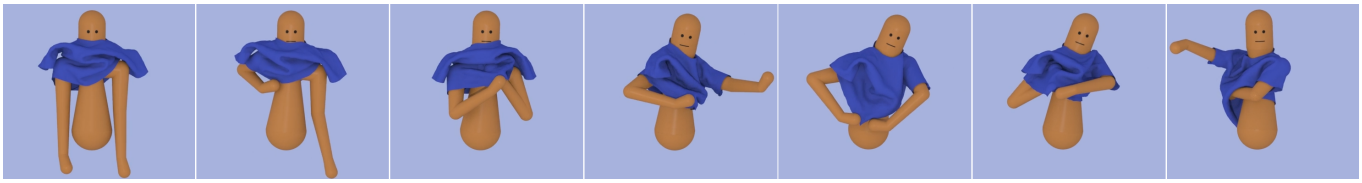

Fig. 5. A virtual human dresses a t-shirt.

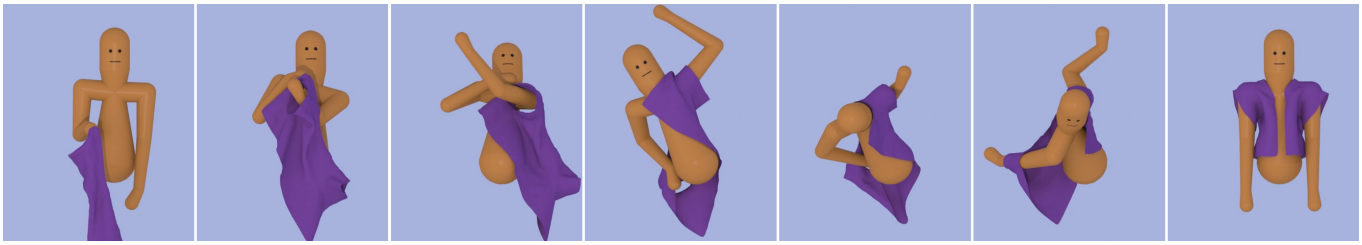

Fig. 6. A virtual human dresses a jacket.

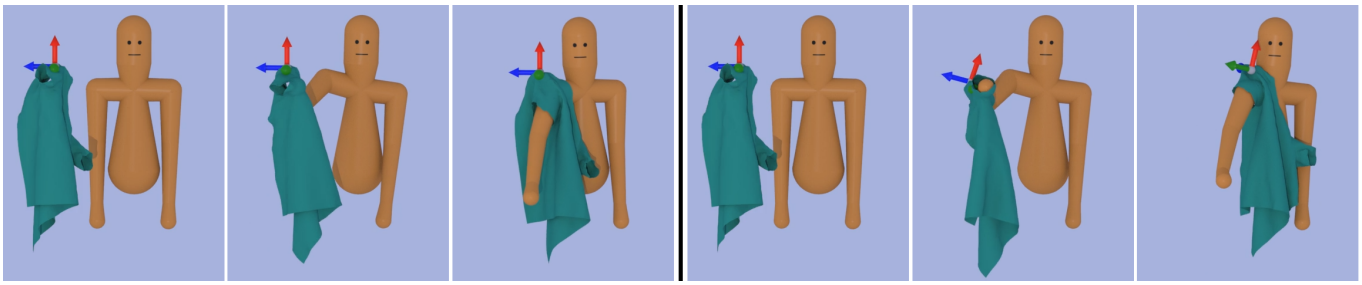

Fig. 7. A virtual human dresses a hospital gown carried by a linear track (left) and a trained assistive gripper policy (right).

in contact). Once the limb has passed a threshold distance through the sleeve, the re-grip controller directs the hands together into position to exchange grip from right hand to left. Once the left hand is within a threshold distance of its gripping target, the grip exchange is triggered and control is transitioned to the second "tuck" control policy with the same purpose and transition criteria as the first. The second sleeve policy is then run to pass the right arm through the right sleeve. At this point, the seventh and final policy is used to guide the character back to its start pose while avoiding garment tearing.

8.2 Jacket

In this task, a jacket is initialized in front of the character, already gripped in the right hand. The control sequence begins immediately with the task of pulling the left sleeve of the jacket onto the left arm. Once the right sleeve feature contains the arm at a depth of 0.75, control switches to a phase transition control policy with the task of reaching a desired transition pose with the right hand behind the back and as close as possible to a fixed point in the local space of the garment while maintaining the dressed left arm. Once the character is within a threshold distance of the task goal, the control policy is transitioned once again to the task of inserting the right arm into the right sleeve, resulting in a "windmill" motion of the arm as it scoops the sleeve onto the arm. Once again, a sleeve containment depth threshold is used to transition to the final, feed forward, SPD controller in order to move the body into a rest pose.

8.3 Assistive Hospital Gown

In order to demonstrate the potential value of modeling human dressing behavior in the assistive robotics domain, we train a control policy to dress one arm of a hospital gown which is suspended from a linear track. The gown is initialized randomly within a specified region in front of the character and moves linearly at a constant rate toward the location of the character's shoulder when the task began. The character learns to lift its arm to insert a hand into the garment and then avoid tearing the garment as it executes the remainder of its traversal.

Once this phase is complete, we fix the character control policy and detach the garment from the linear track, instead fixing it to a robotic gripper with 6 dof torque control provided by a second neural

network policy. We train this policy by continuing to randomly initialize the gown in the starting region and applying the same reward function we used to train to the initial human control policy. The robotic gripper policy is given only the pose of the character and its own 6 dof pose as policy input. After training, the robotic gripper learns to cooperatively seek the human's hand and follow its arm, even turning at the end of the task to settle the gown on the human's shoulder. As shown in Figure 9, this cooperatively trained assistive policy completes the arm dressing task much faster than the human was capable of alone with the original linear track. This opens many possible avenues of research for robot assisted dressing applications.

9 EVALUATION

In order to evaluate our approach, we first show the necessity of feedback control by generating a successful tshirt sequence trajectory with fixed random seed 0 and then replaying this trajectory for random seeds [1,10]. Even for the fairly narrow initial distribution of this task, the trajectory replay strategy fails 80% of the time on test seeds. Our control policy sequence, however, succeeds on 100% of initial seeds [0,10] and 79% of initial seeds [0,100].

To analyze the necessity of the various reward and observation terms we propose in our approach, we conduct an ablation study to evaluate the impact of removing individual terms. Specifically, we train the first sleeve dressing task control policy from a single initial state over 3000 TRPO iterations with the following four variations: our complete proposed set of terms, ablating the oracle from observation and reward, ablating the contact geodesic reward term, and ablating both the contact ID and haptic features from the observation. We exclude the deformation penalty and limb progress from this study because they are directly related to completing the task. As shown in figure 8, the complete policy trained on our proposed combination of reward and observation features out-performs all ablated policies significantly, being the only policy to successfully dress the sleeve. Both the Haptic and Task ablation policies manage to navigate to the sleeve feature, but are unable to reach the arm through the feature. The Geodesic ablation policy, however, is unable to reach the sleeve feature and instead finds an oscillatory cycle enabling endless accumulation of small reward by moving in the direction of the task vector. While these data strongly support our choice of reward and observation features, we encourage the reader to view the accompanying video results for a better understanding of the behaviors exhibited by these control policies.

To demonstrate the necessity of our policy sequencing algorithm (Algorithm 1), we replace the final, tuned control policy for the 1st sleeve dressing subtask of the t-shirt sequence with a policy trained only on a narrow initial state distribution (first pass of the algorithm). We then run the modified t-shirt sequence policy on seeds [1,10] and observe a 90% failure rate on the sleeve dressing subtask. This can be attributed to the mismatch in the end state distribution of the tuck sub-task and the initial distribution of the un-tuned sleeve dressing subtask controller.

Additionally, we examine the potential for a reinforcement learning framework to aid in the development of assistive robotics control policies. Section 8.3 details the training of a human behavior policy

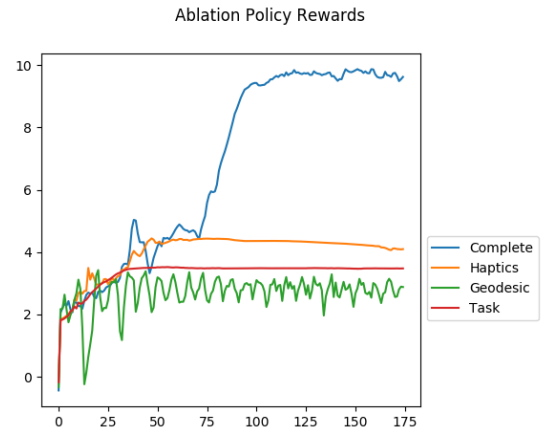

Fig. 8. Final ablation study policies' performance evaluated on the complete reward function averaged over 10 trajectories with timesteps on the x-axis.

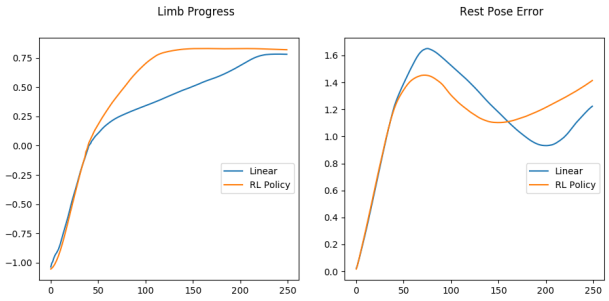

Fig. 9. Comparison of the limb progress metric and rest pose error on the MDP horizon, averaged over 10 trajectories for the linear track and RL trained assistive policies with timesteps on the x-axis.

for inserting one arm into the sleeve of a hospital gown gripped by an assisitive robot with two control modes. In the first control mode, the robotic gripper moves along a fixed linear track toward the shoulder. In the second control mode, gripper motion is controlled by a neural network policy with a six dof torque based action space composed of three dimensional translation and rotation. Figure 9 shows a comparison of the performance of these control modes measuring limb insertion progress and character rest pose error over the full task horizon, averaged over ten trajectories. These plots clearly show that the learned gripper policy accumulates limb progress and completes the task (>= 0.75 limb progress) much faster than the linear policy (step 130 vs. 220) and results in lower rest pose error before task completion. This indicates that a reinforcement learning framework can successfully be applied to modeling human dressing behavior and developing successful control policies for robotic dressing assistants.

10 CONCLUSIONS

We have presented a system that learns to animate a character that is putting on clothing through the use of reinforcement learning and physics simulation. After the dressing task has been divided into manageable sub-tasks, the system learns each sub-task individually, connecting them with a state machines and matching the input state distribution of each sub-task to the output distribution of the prior sub-task. We have found that careful selection of the cloth observations and the reward functions are important to the success of this approach. The result of our approach is not just a single dressing sequence, but a character controller that can successfully dress under various initial conditions.

Despite the success of this system on several dressing tasks, there is always room for improvement. Our system currently performs upper body tasks, and extension to lower body dressing would require incorporating balance into the controller. Although our cloth observation space is sufficient for the demonstrated tasks, it would be interesting to see if an end-to-end controller could be trained that makes use of simulated vision to determine the cloth state. Our reduced haptic observation serves to span the divide between efficiency and representative power, but a more complete model of human haptic perception could prove useful for a variety of applications. Finally, the use of a control policy architecture with memory may prove to reduce the number of necessary subtasks and allow greater generalization of learned skills.

ACKNOWLEDGMENTS

We want to thank the reviewers for their feedback, Charlie Kemp, Zackory Erickson, Ari Kapusta and Henry Clever for their insight and all members of the Graphics Lab at Georgia Tech. This work was supported by NSF Graduate Research Fellowship under Grant No. DGE-1650044, NSF award IIS-1514258, Samsung Global Research Outreach grant and AWS Cloud Credits for Research.

REFERENCES

Yunfei Bai, Wenhao Yu, and C Karen Liu. 2016. Dexterous manipulation of cloth. In *Computer Graphics Forum*, Vol. 35.

Benjamin Balaguer and Stefano Carpin. 2010. Motion planning for cooperative manipulators folding flexible planar objects. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 3842–3847.

Dmitry Berenson. 2013. Manipulation of deformable objects without modeling and simulating deformation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 4525–4532.

Alexander Clegg, Jie Tan, Greg Turk, and C. Karen Liu. 2015. Animating Human Dressing. *ACM Trans. Graph.* 34, 4, Article 116 (July 2015), 9 pages. https://doi.org/10.1145/2766986

Alexander Clegg, Wenhao Yu, Zackory M. Erickson, C. Karen Liu, and Greg Turk. 2017. Learning to Navigate Cloth using Haptics. *CoRR* abs/1703.06905 (2017). http://arxiv.org/abs/1703.06905

Martin de Lasa, Igor Mordatch, and Aaron Hertzmann. 2010. Feature-based Locomotion Controllers. *ACM Trans. Graph.* 29, 4, Article 131 (July 2010), 10 pages. https://doi.org/10.1145/1778765.1781157

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. 2016. Benchmarking Deep Reinforcement Learning for Continuous Control. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16)*. JMLR.org, 1329–1338. http://dl.acm.org/citation.cfm?id=3045390.3045531

Tom Erez, Yuval Tassa, and Emanuel Todorov. 2015. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 4397–4404.

Thomas Geijtenbeek, Michiel van de Panne, and A. Frank van der Stappen. 2013. Flexible Muscle-based Locomotion for Bipedal Creatures. *ACM Trans. Graph.* 32, 6, Article 206 (Nov. 2013), 11 pages. https://doi.org/10.1145/2508363.2508399

Nicolas Heess, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, Ali Eslami, Martin Riedmiller, et al. 2017. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286* (2017).

Edmond SL Ho and Taku Komura. 2009. Character motion synthesis by topology coordinates. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 299–308.

Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. 1995. Animating Human Athletics. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. ACM, New York, NY, USA, 71–78. https://doi.org/10.1145/218380.218414

Sumit Jain, Yuting Ye, and C. Karen Liu. 2009. Optimization-Based Interactive Motion Synthesis. *ACM Transaction on Graphics* 28, 1 (2009), 1–10.

Yifeng Jiang and C. Karen Liu. 2017. Data-Driven Approach to Simulating Realistic Human Joint Constraints. *CoRR* abs/1709.08685 (2017). arXiv:1709.08685 http://arxiv.org/abs/1709.08685

Jeongseok Lee, Michael Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha Srinivasa, Mike Stilman, and C Karen Liu. 2018. DART: Dynamic Animation and Robotics Toolkit. 3 (02 2018), 500.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

Libin Liu and Jessica Hodgins. 2017. Learning to schedule control fragments for physics-based characters using deep q-learning. *ACM Transactions on Graphics (TOG)* 36, 3 (2017), 29.

Libin Liu, Michiel Van De Panne, and Kangkang Yin. 2016. Guided Learning of Control Graphs for Physics-Based Characters. *ACM Trans. Graph.* 35, 3, Article 29 (May 2016), 14 pages. https://doi.org/10.1145/2893476

Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. 2014. Unified Particle Physics for Real-time Applications. *ACM Trans. Graph.* 33, 4, Article 153 (July 2014), 12 pages. https://doi.org/10.1145/2601097.2601152

Eder Miguel, Andrew Feng, Yuyu Xu, Ari Shapiro, Rasmus Tamstorf, Derek Bradley, Sara C Schvartzman, Bernhard Thomaszewsky, Bernd Bickel, Wojciech Matusik, et al. 2014. Towards cloth-manipulating characters. In *Computer Animation and Social Agents*, Vol. 3.

Stephen Miller, Jur Van Den Berg, Mario Fritz, Trevor Darrell, Ken Goldberg, and Pieter Abbeel. 2012. A geometric approach to robotic laundry folding. *The International Journal of Robotics Research* 31, 2 (2012), 249–267.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

Kourosh Naderi, Joose Rajamäki, and Perttu Hämäläinen. 2017. Discovering and Synthesizing Humanoid Climbing Movements. *ACM Trans. Graph.* 36, 4, Article 43 (July 2017), 11 pages. https://doi.org/10.1145/3072959.3073707

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *ACM Transactions on Graphics (Proc. SIGGRAPH 2018 - to appear)* 37, 4 (2018).

John Schulman, Sergey Levine, Philipp Moritz, Michael I Jordan, and Pieter Abbeel. 2015a. Trust region policy optimization. *CoRR, abs/1502.05477* (2015).

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015b. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature* 550, 7676 (2017), 354.

Tomoya Tamei, Takamitsu Matsubara, Akshara Rai, and Tomohiro Shibata. 2011. Reinforcement learning of clothing assistance with a dual-arm robot. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*. IEEE, 733–738.

Jur Van Den Berg, Stephen Miller, Ken Goldberg, and Pieter Abbeel. 2010. Gravity-based robotic cloth folding. In *Algorithmic Foundations of Robotics IX*. Springer, 409–424.

He Wang, Kirill A Sidorov, Peter Sandilands, and Taku Komura. 2013. Harmonic parameterization by electrostatics. *ACM Transactions on Graphics (TOG)* 32, 5 (2013), 155.

KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. SIMBICON: simple biped locomotion control. *ACM Trans. on Graphics (SIGGRAPH)* 26, 3 (2007), 105.

Wenhao Yu, Greg Turk, and C. Karen Liu. 2018. Learning Symmetric and Low-energy Locomotion. *ACM Transactions on Graphics (Proc. SIGGRAPH 2018 - to appear)* 37, 4 (2018).