Haptic Simulation for Robot-Assisted Dressing

Wenhao Yu, Ariel Kapusta, Jie Tan, Charles C. Kemp, Greg Turk and C. Karen Liu

***Abstract*— There is a considerable need for assistive dressing among people with disabilities, and robots have the potential to fulfill this need. However, training such a robot would require extensive trials in order to learn the skills of assistive dressing. Such training would be time-consuming and require considerable effort to recruit participants and conduct trials. In addition, for some cases that might cause injury to the person being dressed, it is impractical and unethical to perform such trials. In this work, we focus on a representative dressing task of pulling the sleeve of a hospital gown onto a person's arm. We present a system that learns a haptic classifier for the outcome of the task given few (2-3) real-world trials with one person. Our system first optimizes the parameters of a physics simulator using real-world data. Using the optimized simulator, the system then simulates more haptic sensory data with noise models that account for randomness in the experiment. We then train hidden Markov Models (HMMs) on the simulated haptic data. The trained HMMs can then be used to classify and predict the outcome of the assistive dressing task based on haptic signals measured by a real robot's end effector. This system achieves 92.83% accuracy in classifying the outcome of the robot-assisted dressing task with people not included in simulation optimization. We compare our classifiers to those trained on real-world data. We show that the classifiers from our system can categorize the dressing task outcomes more accurately than classifiers trained on ten times more real data.**

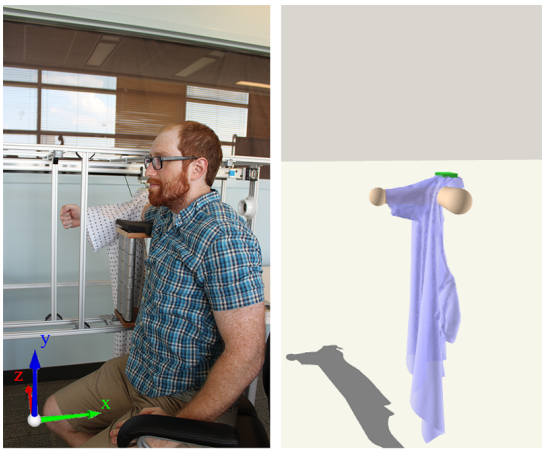
I. INTRODUCTION

Many people do not have the ability to dress themselves, and thus require assistance with this task. According to some studies, other than bathing/showering assistance, dressing is the most received assistance for older adults among all activities of daily living, and dressing assistance is needed by over 80% of people in skilled nursing facilities [1]. In this work, we focus on robot-assisted dressing, which is a growing area of study that has the potential to alleviate this problem. We work on a specific assistive dressing task, where a robot pulls the sleeve of a hospital gown onto a person's forearm, as illustrated in Fig. 1 (a). Kapusta *et al.* [2] demonstrated that for this dressing task, the haptic data collected from the robot end effector holding the hospital gown can be used to train a classifier that accurately categorizes whether the person's arm is going to 1) successfully go into the sleeve, 2) miss the sleeve opening completely or 3) get caught by the gown. This indicates that haptic sensing can provide valuable information for robots to infer the relationship between clothing and the human body. However, obtaining this real-world haptic data requires extensive amounts of time and human labor. Also, it is difficult to obtain data for failure cases, especially for those that might cause injury to the person.

Wenhao Yu, Jie Tan, Greg Turk and C. Karen Liu are with the School of Interactive Computing, Georgia Institute of Technology, Atlanta, USA

Ariel Kapusta and Charles C. Kemp are with the Healthcare Robotics Lab, Georgia Institute of Technology, Atlanta, USA

Wenhao Yu is the corresponding author (wyu68@gatech.edu).



(a) (b)

Fig. 1: The assistive dressing task (a) demonstrated by an experimenter and (b) simulated by our system.

In this research, we extend the work of Kapusta *et al.* [2] by incorporating physical *cloth simulation* techniques into the data generation pipeline, thus reducing the required number of real-world experiments. Physical cloth simulation usually models cloth as a series of vertices and advances their state (e.g. position and velocity) by solving the equations of motion. Cloth simulation has the potential to generalize to cases that we haven't seen in real-world experiments. Our system is divided into three stages: *optimize*, *simulate*, and *classify*. Our system *optimizes* the parameters of a physics simulator by matching the simulated data to two to three haptic data sequences for one participant from real-world experiments. The resulting optimized physics engine is then used to *simulate* new haptic data. Finally, we train hidden Markov Models (HMMs) with the simulated haptic data and demonstrate that these HMMs can achieve high accuracy in classifying the outcome of the task for real-world test data from participants not included in the dataset used for optimizing the simulation. Fig. 1 illustrates the dressing task in the real world and in simulation.

To evaluate the effectiveness of our system, we compare the classifiers trained by our system to classifiers trained with real-world data. We also evaluate the importance of the arm height offset in the simulator optimization formulation.

II. RELATED WORK

A. Robot-Assisted Dressing

Most of the prior work in robot-assisted dressing has been focused on vision and kinematics based methods. Tamei *et al.* demonstrated using a robot to put a mannequin's head into a T-shirt. They used topological coordinates to represent the

relationship between the T-shirt and the mannequin in low-dimensional space and applied reinforcement learning to learn the arm motion for the robot to dress the mannequin [3]. Gao *et al.* focused on user modeling, where they used a depth camera to estimate the user pose and modeled the joint movement space with GMMs. They presented results in which a Baxter robot dresses a person with sleeveless jacket[4]. Klee *et al.* introduced a turn-taking approach where the robot coordinates with a human during assistive dressing and learns the person's mobility limitations using a vision module. They demonstrated putting a hat onto a person's head [5].

There have also been works focusing on visual perception of cloth during dressing. Koganti *et al.* used a depth sensor to estimate a human cloth topological relationship and showed the feasibility of using it to replace the more complex motion capture system [6]. Further, an offline learning method was proposed in the more recent work by Koganti *et al.* , in which they fitted a cloth dynamics model with multi sensor data and applied it to track human-cloth relationships in real-time using a depth sensor [7].

During robot-assisted dressing, vision-based methods usually suffer from occlusion of task-related objects from cloth and the robot itself. More recently, there have been works that incorporated haptic information into robot assisted dressing. Yamazaki *et al.* demonstrated pulling of a bottom along the legs using a life-size humanoid robot, where they applied both visual and force sensing to detect failure cases and re-plan the robot motion [8]. Gao *et al.* introduced a stochastic path optimization method to search for the optimal personalized dressing path for the robot using the estimated human pose and haptic data from the sensor attached at the robot's end effector [9]. To evaluate the usefulness of haptic data for robot-assisted dressing, Kapusta *et al.* conducted a user-study that had a robot pull a hospital gown's sleeve onto a person's arm. They showed that Hidden Makove Models (HMMs) trained on force data from the robot's end effector can achieve high accuracy in classifying outcomes of this robot-assisted dressing task [2]. Our work builds on the work by Kapusta *et al.* [2] and aims to reduce the number of real-world trials needed to train a high-accuracy classifier.

B. Anomaly Detection

Our work is also related to anomaly detection from force or trajectory data,[10],[11],[12]. Park *et al.* presented an online anomaly detection framework using multimodal sensory inputs. They demonstrated their methods on anomaly detection of a robot performing various tasks including door pushing and assistive yogurt feeding. [13]. Rodriguez *et al.* collected force data during part assembly and trained an SVM to detect failed assemblies [14]. In the work by Pastor *et al.* , they applied reinforcement learning for robots to acquire new motor skills from demonstration. In their system, they trained a prediction model from real-world sequences to detect failure outcomes [15]. On the other hand, our work focuses on detection of multiple outcomes during robot-assisted dressing, instead of general anomaly detection.

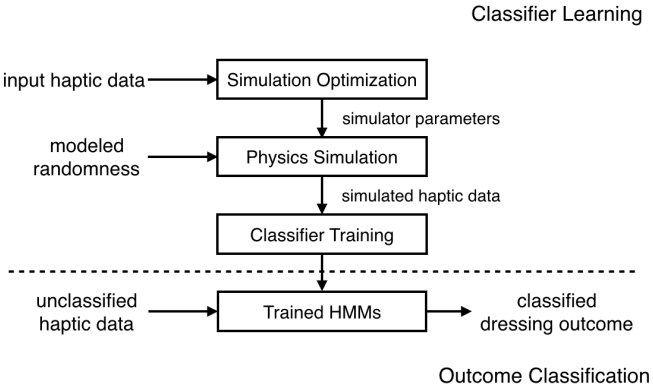


Fig. 2: Overview of our system.

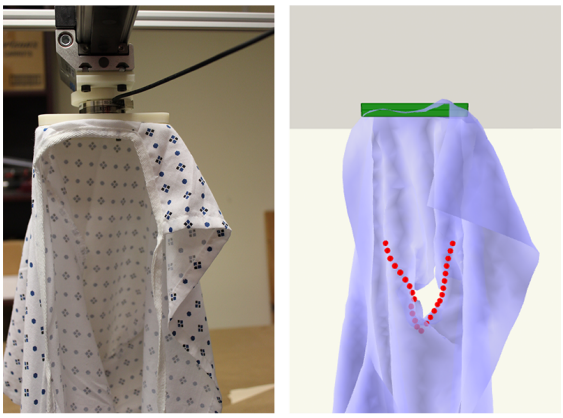
C. Physics Simulation in Robotics

Physics simulation has been used to debug or validate algorithms before applying them to real robots [16], [17], [18]. In addition, a few works have investigated integrating physics simulation into the robot learning or control algorithm. Haidu *et al.* presented an interactive game of cooking pancakes using rigid-body and fluid simulation. They collected data from a human performing the cooking task and trained classifiers on the data to predict failure outcomes [19]. In a more recent work by Haidu *et al.* , they demonstrated predicting failure cases for the same task on a real robot, in which they learned a classifier by constructing envelopes for the data collected from the interactive game [20]. Jie *et al.* proposed an iterative method for balancing a humanoid robot. They used physics simulation to perform system identification and trajectory optimization [21]. Wu *et al.* demonstrated using a physics simulator to infer the properties of rigid objects in videos or images and to predict the physical events in the scene. They further learned a direct mapping from visual features to the object properties using deep learning techniques [22]. In the work by Cusumano-Towner *et al.* , they used an Hidden Markov Model to infer where the garment is being grasped and identify the state of the garment. Based on this and a physical cloth simulator, they planned the motion for a robot to bring a garment into a desired configuration [23].

III. OVERVIEW

Our system trains Hidden Markov Models (HMMs) that classify the outcome of a dressing task using a small amount of haptic data from real-world measurements. We demonstrate our system on the same dressing task investigated by Kapusta *et al.* [2]— pulling the sleeve of a hospital gown onto a person's arm. This task is representative of various assistive dressing tasks for two reasons: 1) a hospital gown is a common clothing article with which health professionals provide assistance and 2) pulling the sleeve onto a person's arm represents a general dressing class, where a cloth tube is pulled onto the human body.

Fig. 2 shows an overview of our system. The inputs to our system are two or three sequences of forces measured at the robot end-effector (Fig. 3 (a)). The system first optimizes the parameters of a physics simulator based on the input data sequences. Once optimized, the physics simulator is



(a) (b)

Fig. 3: The hospital gown being grasped by the robot's end effector (the rectangular piece) in (a) experiment and (b) simulation. The red dots shown on the simulated gown denotes the lower half of the sleeve opening, which are used to automatically label the simulated assistive dressing trials.

used to generate a large number of synthetic haptic data sequences. Finally, the system trains HMMs using forces from the synthetic sequences. Only two components of the forces are used during training: in the end-effector direction of movement and in the direction of gravity. The trained HMMs are used to predict in real world whether the arm will successfully go into the sleeve, miss the sleeve completely, or get caught by the gown.

IV. SIMULATOR OPTIMIZATION

The heart of our system is a physics simulator capable of producing haptic data, to be used for learning a model that classifies and predicts the outcome of the dressing task. To simulate the dressing task, we create a triangle mesh from measurements of a hospital gown, and we use capsules to model the person's arm. The modeled arm has uniform friction coefficient and is fixed during the simulation. The haptic data in our experiment consist of the force-torque information measured at the robot's end-effector, where the gown is held. Simulating this 6-dimensional data involves complex computations on cloth dynamics and collisions, which are sensitive to numerous simulation and model parameters, as well as the dynamic integration scheme and collision handling method used. As such, the success of haptic data simulation depends on the choice of the initial simulator, as well as identifying the right model parameters.

A. Initial cloth Simulator

An ideal cloth simulator should be fast, stable, and accurate. Existing cloth simulators, however, can only achieve one or two of these three desirable properties. In order for a robot to acquire skills for assistive dressing through physics simulation, many simulated trials under various conditions would be especially useful. Therefore a relatively accurate but slow cloth simulator, such as one based on Finite Element Methods (FEMs) [24], [25], is impractical for many machine learning methods that require large amounts of training data. Moreover, FEM-based cloth simulation tends to be unstable when large contact forces or large deformations occur, which are common events in a dressing scenario.

In this work, we choose to use the cloth simulator in Nvidia PhysX 3.3.4 [26] due to its efficiency and robustness at handling large contact forces (e.g. when the sleeve is caught on the hand). From our test, PhysX runs at 0.2x real-time on an Intel Core i5 CPU for our hospital gown model consisting of 20k triangles, which is 60 times faster than the FEM-based cloth simulator from Narain *et al.* [25]. PhysX is based on Position-based Dynamics (PBD), which calculates position changes directly instead of through force integration. Being able to avoid solving and integrating the forces and to modify position directly makes PBD more stable, controllable, and efficient than many alternative methods.

These desirable properties make PhysX suitable for applications that require an efficient and stable simulator. However, the standard PhysX software is not sufficient for producing reliable haptic data for our purpose. We implemented additional functions in PhysX to calculate haptic data and improve the accuracy of the friction handling. These changes are described below.

1) Collision Force Calculation: Because forces are not explicitly calculated in the PBD formulation, we need to extract the force information from the positions of each vertex on the cloth. During simulation, the position of a vertex at timestep $t+1$ is calculated as:

$$x_{t+1} = x_t + \Delta t \dot{x}_t + \frac{(\Delta t)^2 f_{ext}}{m} + \delta x_{int} + \delta x_{col}, \qquad (1)$$

where x_t is the position at timestep t, $\dot{x}$ is the velocity, f_{ext} is the external force, m is the mass of the vertex, δx_{int} is the position change due to internal constraints and δx_{col} is the position change due to collision. Our goal is to compute a collision force f_{col} that, when applied to the vertex, would achieve the same position change as δx_{col}. To do this, we equate δx_{col} to the position change due to f_{col} and solve for the collision force:

$$f_{col} = \frac{\delta x_{col} m}{\Delta t^2}. \qquad (2)$$

2) Improved Friction Scheme: In the original PBD friction scheme, the calculated friction impulses are added to the velocity of the vertex at the next timestep. In this scheme, tangential position changes caused by internal force or gravity force before the friction solving are not corrected, resulting in underestimation of friction force. In fact, we find that in our experiments, even the maximum friction coefficient allowed by PhysX is not able to generate enough friction to match the real-world data. To address this problem, we add the calculated friction impulse to the current position, in addition to the velocity for the next timestep. The new scheme improves the accuracy of the friction solver and enables us to match the real world data.

3) Cloth-Cloth Friction: The default PhysX implementation does not model friction forces for cloth self-collisions. The lack of friction forces often make the simulated cloth

appear overly slippery without wrinkles or folds. We implemented cloth-cloth friction using the same algorithm for cloth-object friction as described in IV-A.2.

B. Optimization

PhysX provides a flexible framework that allows simulating a wide range of cloth materials (e.g. silk or denim). However, the parameters in PhysX do not have a direct mapping to real-world cloth materials, preventing us from directly measuring the parameters from real garments. Instead, we use an optimization procedure to search for suitable simulator parameters. The input to our optimization procedure includes a few sequences of real-world haptic data and an initial set of cloth parameters for the simulator. The optimizer then searches for a set of cloth parameters that minimize the difference between real and simulated forces in the direction of gravity (Y) and end-effector movement (X). By fitting the simulation parameters to the real-world sequences, we expect the optimized simulator to generalize to other situations relevant to our classification problem, such as different arm heights and initial configurations. However, given so few real-world sequences, it is unlikely that the optimized simulator can generalize to all conditions.

1) Input Sequence Selection: We use the haptic data acquired by Kapusta *et al.* [2]. The dataset consists of 360 sequences from 12 participants. For each participant, 30 sequences were collected with their arm positioned at three heights. For all trials at the highest height, the arm successfully goes into the sleeve, and at the lowest height, the arm misses the sleeve. For the middle height, either the arm misses the sleeve or the arm gets caught by the gown. Each sequence contains 6-dimensional force and torque measurements at the robot's end-effector, but only the forces in the horizontal direction of end-effector movement (X) and in the gravity direction (Y) are used. Our system uses two or three sequences as the input, as chosen by the policy given below.

For the participant of interest, we randomly select one sequence from each of the three heights, excluding all the *caught* sequences. For a participant whose middle height only contains *caught* sequences, we end up selecting only two sequences for optimization. We do not include *caught* sequences because the rapid increase of force in *caught* sequences lead to sub-optimal simulators with an unrealistically large friction coefficient. Our results show that the exclusion of *caught* sequences during parameter optimization does not affect the ability of the simulator to generate cases when the arm is caught by the gown.

2) Simulation Parameters: PhysX provides a large set of free parameters for cloth simulation (24 in total). To avoid overfitting to a small set of examples, we identify six parameters directly related to the cloth materials for optimization, namely the stiffness of vertical stretching, horizontal stretching, bending, shearing, and the coefficients of friction and self-friction. We further assume isotropic material for our cloth model and thus combine vertical and horizontal stretching stiffness into one parameter. We use an iteration number of 20 for the constraint solver and use default values for the rest of the parameters[1]. In addition, the measured dataset includes variations in the precise height of the arm due to human factors, therefore we include a parameter that models the deviation between the actual height of the arm and the controlled height (the height of the armrest). We include this parameter for each of the controlled heights. The final dimension of the parameter vector is either seven or eight, depending on whether a middle height sequence is used.

3) Objective Function: We define the objective function as the difference of forces in X and Y direction between simulated and input sequences. For each input sequence, the system runs a complete simulation at the corresponding arm height and records haptic data from the simulated robot end-effector. For force data in each direction, we compute the error $e(S, R)$ between synthetic data $s_i \in S = [s_0, s_1, \cdots, s_m]$ and input data $r_i \in R = [r_0, r_1, \cdots, r_n]$:

$$e(S, R) = \sum_{i}^{max(n,m)} d(s_i, r_i). \tag{3}$$

The per-frame error metric $d(s_i, r_i)$ is defined as:

$$d(s_i, r_i) = \begin{cases} \Delta x |s_i - r_i| & \text{if } i \leq m \text{ and } i \leq n \\ w\Delta x |s_i - r_n| & \text{if } i \leq m \text{ and } i > n \\ w\Delta x |s_m - r_i| & \text{if } i > m \text{ and } i \leq n \end{cases}, \tag{4}$$

where Δx is the distance traveled by the end-effector between two consecutive frames and $w = 20$ is a weight to penalize differences in sequence length.

The error function is an approximation of the unsigned area difference between two curves, with a penalty for difference in sequence length. The objective function is then defined as:

$$L(R, \theta) = \sum_{k=1}^{N} (e(S_k^Y(\theta), R_k^Y) + e(S_k^X(\theta), R_k^X)), \tag{5}$$

where θ is the set of simulation parameters to be optimized. The subscript k is the index of the input sequence ($N = 2$ or 3) and the superscript X or Y is the direction of the force.

4) Solving The Optimization: We can now formulate our optimization problem as:

$$\theta^* = \underset{\theta}{\text{argmin}}\ L(R, \theta). \tag{6}$$

However, it is nontrivial to use gradient-based optimization methods because the objective function is highly nonlinear and non-differentiable in θ. To solve this problem, we use Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [27], a stochastic algorithm for solving nonlinear optimization problems. We run CMA-ES for 50 iterations with 32 samples per iteration. The resulting objective value of the optimal solution represents how close the physics simulation is able to match the real-world data.

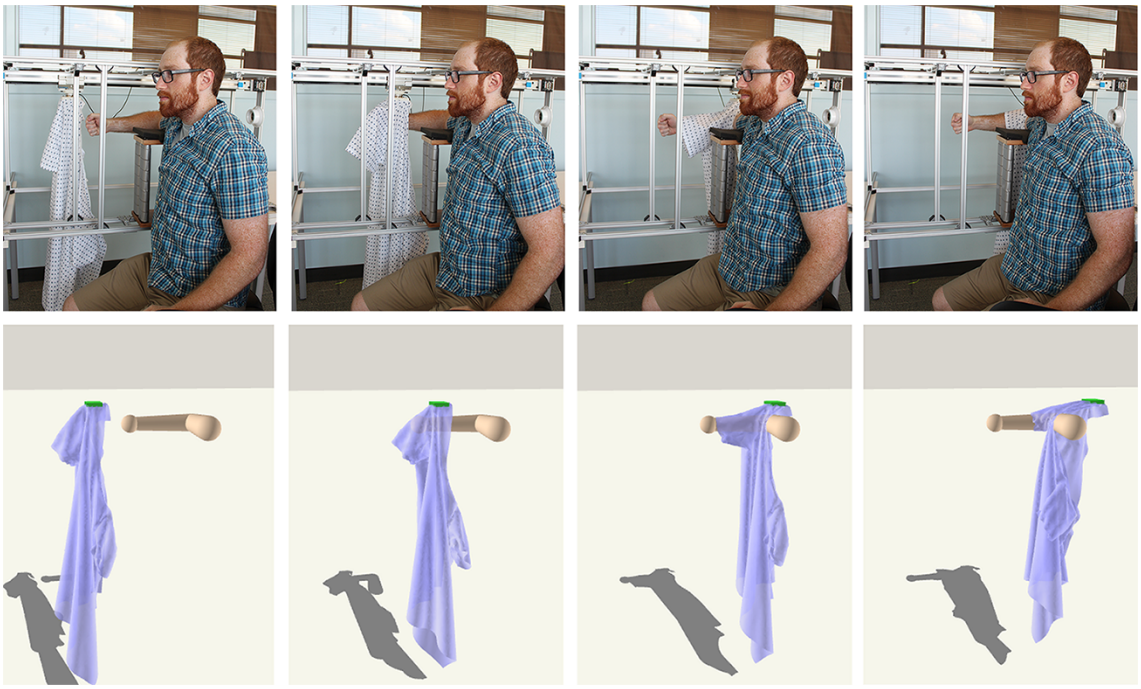


Fig. 4: Successfully dressing one sleeve in real-world and in simulation.

V. CLASSIFIER TRAINING

A. Haptic Data Simulation

With the optimized simulator, we are able to generate haptic data similar to the input sequences. However, the success of classification also depends on modeling the distribution of the real world data accurately. In our dressing task, two major factors, among many others, that cannot be precisely controlled during data acquisition are the initial state of the hospital gown and the actual arm height of the participant. As such, we model these two factors as probabilistic distributions when synthesizing haptic data using the optimized simulator.

We perturb the initial state of the cloth by adding a random external force equally distributed to all the cloth vertices. The random force is sampled from a uniform distribution in the X and Z direction with the maximum magnitude being 6N. We apply the force to the cloth for 0.5 seconds and wait for the gown to settle before the end-effector starts moving.

Similarly, we randomize the arm height around the highest, the lowest, and the middle arm heights used in the experiments [2]. For the highest and the lowest arm heights, which produce *good* and *missed* outcomes respectively, we add a uniformly distributed noise $\sigma = [-1.5, 1.5]$cm to the optimized arm height. For the middle arm height, which is expected to generate *caught* or *missed* outcomes, we add a larger uniformly distributed noise $\sigma = [-6, 0]$cm to increase the chance of generating both *caught* and *missed* outcomes. Note that for the middle arm height, we directly add the noise to the middle arm height because not all participants have an optimized middle arm height. Further, we only add *negative* noise to compensate the gravity effect.

[1]We refer readers to http://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/Cloth.html for details in PhysX cloth parameters.

We run the simulation at 300 Hz and record simulated haptic data once every three frames to match the sample rate of the force-torque sensor used in the user-study [2]. We generate 100 sequences for each height. Using the same protocol as the user-study, half of the sequences are generated with the end effector moving at 0.1 m/s and the other half at 0.15 m/s. The simulation is terminated when either the end-effector travels 0.85 meters or the force in the X or Y direction reaches 10N. The simulated haptic data is automatically labeled according to the following policy.

If the end-effector travels the full distance of 0.85 meters, we label the resulting sequence as *missed*. Otherwise, we compute the distance between the vertices on the lower half of the sleeve opening (Fig. 3 (b)) and the center of the person's elbow in the X direction and find the maximum value d_{max}. If $d_{max} \leq 7$ cm, we label this sequence as *good*, otherwise we label it as *caught*. Fig. 4 shows the simulated dressing task and a real-world demonstration where the sleeve is successfully pulled onto the arm.

B. Hidden Markov Model

We train three hidden Markov Models (HMMs) to classify the outcome of the assistive dressing task given the simulated haptic data. We use the General HMM library[28] to train and run left-right HMMs. One HMM was trained for each of the three task outcomes, with the simulated data set labeled as *missed*, *good*, or *caught*. Similar to Kapusta *et al.* [2], we use bivariate HMMs trained on force data in the X and Y direction. When testing with a new haptic sequence, each of the trained HMMs predicts the probability of the new sequence being in its class. The classifier then assigns the category label corresponding to the HMM with the highest likelihood.

VI. RESULTS

In this section, we present the performance of our Simulation Trained Classifiers (STCs). We tested the accuracy of STCs on the labeled data from the user-study [2]. The accuracy is defined as the percentage of test sequences that are correctly classified into one of the three classes: "good", "missed", and "caught". For all experiments presented here, we trained bivariate HMMs with 10 hidden states, similar to Kapusta *et al.* [2].

A. Classification Accuracy of STCs

For each of the 12 participants in the dataset, we trained one STC using our system and tested it on the data from the remaining 11 participants. This yields 3960 classification results in total. We present this data in a confusion matrix, as shown in Fig. 5 (a). The STCs achieves 92.83% accuracy on average. For comparison, a random classifier would result in 33.3% accuracy and a majority classifier would achieve 40%. Below, we compare STCs with baseline classifiers trained on different amounts of real-world data. HMMs are used in all baseline classifiers.

B. Comparison with Real-World Data Trained Classifiers

1) Comparison with Baseline Classifiers: For each of the 12 participants, we trained a Baseline Classifier (BC) directly with the input sequences (N = 2 or 3) used to optimize the simulator. Recall that during the simulator optimization we chose not to include sequences labeled as *caught*. To conduct the comparison, we added one *caught* sequence, randomly chosen from each participant, to the training set of BC so that it can learn to classify all three classes. BCs achieved an overall accuracy of 75.13% as illustrated in Fig. 5 (b).

This confirms that our system does have a positive effect by generating haptic data. However, this is not surprising, since a classifier that is trained on such a small amount of data is not likely to perform well.

2) Comparison with Extended Baseline Classifiers: Our next comparison is against Extended Baseline Classifiers (EBCs). For each participant, we trained one EBC on his/her entire data set (N=30). We again present average classification accuracy for EBCs. The resulting EBCs achieved 88.01% accuracy, as shown in Fig. 5 (c). Out of the 3960 test samples, the STCs from our system were able to classify 191 more samples correctly than EBCs.

3) Performance With Limited Distance: In our experiment, the robot stops moving when it travels 0.85m in distance or when a force (in the X or Y direction) of 10N or above is detected. During the assistive dressing task, it will be beneficial if we can predict the outcome earlier, before the task is complete. To evaluate how well STCs predict the outcome of our dressing task, we truncated the test sequences at various distances and used the truncated sequences to test the performance of STCs and EBCs. Fig. 6 shows the comparison of the two sets of classifiers with regard to travel distance of the robot's end-effector. It can be seen that the EBCs perform better earlier in the process, while our STCs are able to catch up and surpass them later on.

VII. DISCUSSION AND LIMITATIONS

We have shown that our Simulation Trained Classifiers outperform Extended Baseline Classifiers on average. To further evaluate the performance of the individual STCs, we collected the accuracy of all the STCs for each participant, as shown in Table I. Note that most of the classifiers achieved reasonable results (9 out of 12 were over 90% and 11 were over 85%), but the classifier trained on participant 11 achieved only 71.52% accuracy. The significantly lower accuracy of participant 11 is also observed in EBCs (68.79% accuracy). This suggests that the data collected from participant 11 do not represent well the haptic data distribution for this dressing task, resulting in a sub-optimal physics simulator that generates biased data.

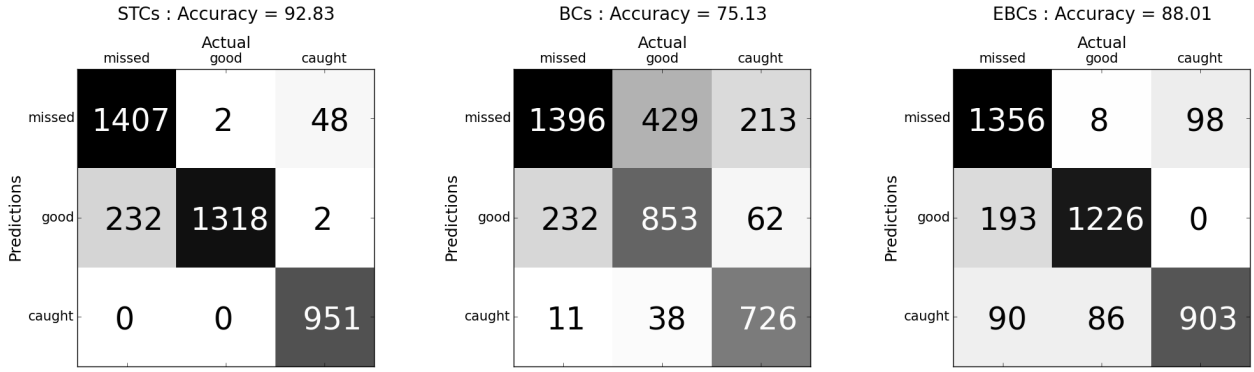
Fig. 7 and Fig. 8, visualize the simulated data and the real-world data for the most and the least accurate STCs respectively. The data for *good* and *missed* are shown in bold curves while the data for *caught*, which was not included in the input sequence set, is shown in dashed curves. In general, for *good* and *missed* outcomes, the simulated data tend to be close to the real data from the same participant, with larger variance. This fits our expectation: the simulator optimization would attract the synthetic data towards the real-world data, and simulating large amounts of haptic data (10 times the real-world data) with the noise model would introduce greater variance. Note that we were able to reliably synthesize data for *caught* even though it is not used during the simulator optimization. This suggests the potential for simulation-trained classifiers to detect new events during robot-assisted dressing.

One insight that we gain from our experiments is that modeling the variance in the real world accurately is important to the success of classification using synthetic data. In this work, we identified two types of randomnesses in the experiments [2], the initial gown configuration and the arm height deviation. We evaluated the importance of these two factors through the following experiments.

We first trained and tested classifiers from simulated data that were generated without the initial gown configuration perturbation. This resulted in an average classification accuracy of 89.17%, which is about 3.5% lower than STCs. Similarly, we trained and tested classifiers from simulated data without arm height perturbation. The resulting average accuracy was 81.77%, which outperforms the Baseline Classifiers, but is substantially lower than the performance of STCs. These results confirm that modeling both source of randomnesses during data synthesis is valuable to achieve accurate classification.

TABLE I: The accuracy of the STCs trained for each participant using our system.

Participant	Accuracy	Participant	Accuracy
1	99.09%	7	99.7%
2	98.18%	8	94.24%
3	97.88%	9	85.76%
4	91.82%	10	91.52%
5	98.48%	11	71.52%
6	88.18%	12	97.58%



(a) (b) (c)

Fig. 5: Confusion matrix showing the performance of (a) STCs, (b) BCs and (c) EBCs.

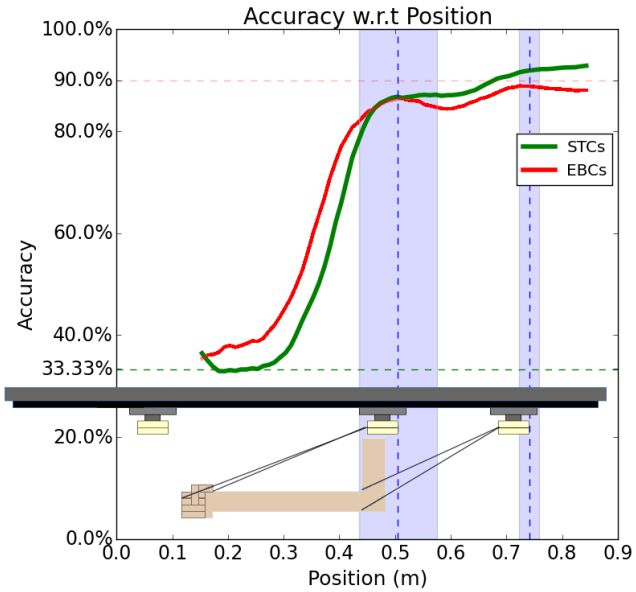


Fig. 6: Comparison of performance in predicting the outcome earlier in the process with STCs and EBCs. The bottom part of the figure illustrates the robot end-effector pulling a gown onto a human arm with average length. From left to right, the dotted lines depict the average stop position for the *caught* event and average stop position for the *good* event.

To demonstrate the importance of modeling the arm height deviation during simulator optimization, we optimized two simulators with the same input sequences. We optimized for the arm height deviation for one of them while fixing it to zero for the other. As shown in Fig. 9, the green dashed curve representing the data from the simulator optimized with arm height deviation fits the real-world data (red solid curve) notably better than the blue dotted curve generated without optimizing for arm height deviation.

There are a few limitations in our system. Although the STCs from our system achieved better performance than EBCs when tested with full data, it did not perform as well when trying to predict the outcome early in the process (roughly before the robot's end effector reaches the person's elbow), as shown in Fig. 6. A possible reason is that the objective function that we used measures the overall similarity between

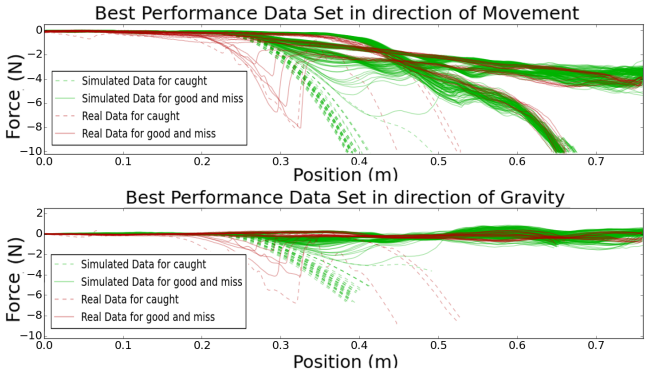


Fig. 7: Data visualization for the most accurate STC overlaid with real-world data from the same participant. The classifier trained on this simulated data set achieved 99.7% accuracy.

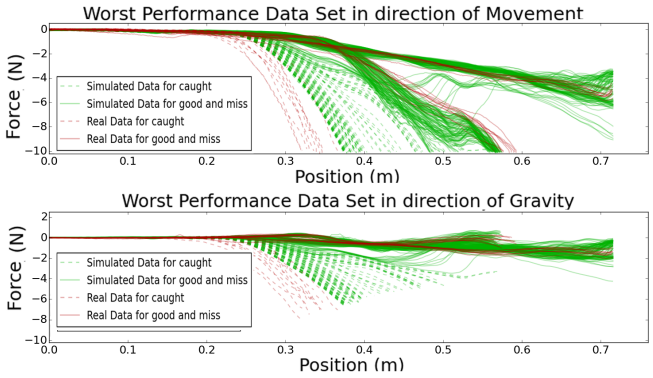


Fig. 8: Data visualization for the least accurate STC overlaid with real-world data from the same participant. The classifier trained on this simulated data set achieved 71.52% accuracy.

two sequences, which might neglect the details when the force is relatively low.

In addition, optimizing the parameters for the simulator is still time consuming despite the fast cloth simulation enabled by PBD. In our experiments, it took about 16.8 hours on a 2.4 GHz Intel Core i5 CPU to optimize a simulator with two input sequences. The optimization parameter size also increases with the number of input sequences due to the arm height deviation parameter. These factors make it difficult for our method to deal with many inputs.

In the work by Kapusta *et al.* [2], they trained classifiers

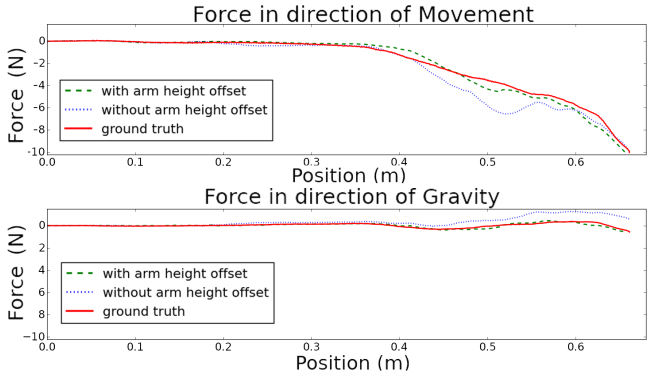


Fig. 9: Optimized simulator with and without arm height deviation for the *good* outcome from participant 7.

on the data of 11 participants and tested on one, resulting in average classification accuracy of 98.61%. We tested a similar procedure where we trained classifiers on simulated data for 11 participants and tested on the real-world data of the other and achieved on average 97.78% accuracy. This is a promising result for including data from multiple participants. However, optimizing 11 simulators with our system currently takes about 10 days. Therefore, how to best take advantage of input sequences from multiple participants efficiently still remains an open question.

VIII. CONCLUSIONS

We have presented a system that learns a classifier to categorize the outcome of a robot pulling the sleeve of a hospital gown onto a person's arm from only two or three real-world sequences. The key components of our system are an optimization scheme to tune a physics simulator with the input sequences, and using the optimized simulator to synthesize haptic data with noise models that account for real-world variations. We demonstrate that the optimized simulator can synthesize outcomes that are not seen during the optimization and that our classifiers achieve high accuracy in categorizing real-world data. Our results suggest that physics simulations can improve robot-assisted dressing by enabling robots to learn from fewer trials with real people and also learn about riskier situations without real-world data.

ACKNOWLEDGMENT

We thank Alexander Clegg and Zackory Erickson for their help with this work. This work is supported by NSF award IIS-1514258 and the National Institute on Disability, Independent Living, and Rehabilitation Research (NIDILRR), grant 90RE5016-01-00 via RERC TechSAge.

REFERENCES

[1] T. L. Mitzner, T. L. Chen, C. C. Kemp, and W. A. Rogers, "Identifying the potential for robotics to assist older adults in different living environments," *International Journal of Social Robotics*, pp. 1–15, 2013.

[2] A. Kapusta, W. Yu, T. Bhattacharjee, C. K. Liu, G. Turk, and C. C. Kemp, "Data-driven haptic perception for robot-assisted dressing," in *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2016*.

[3] T. Tamei, T. Matsubara, A. Rai, and T. Shibata, "Reinforcement learning of clothing assistance with a dual-arm robot," in *Int. Conf. on Humanoid Robots (Humanoids)*. IEEE, 2011, pp. 733–738.

[4] Y. Gao, H. J. Chang, and Y. Demiris, "User modelling for personalised dressing assistance by humanoid robots."

[5] S. D. Klee, B. Q. Ferreira, R. Silva, J. P. Costeira, F. S. Melo, and M. Veloso, "Personalized assistance for dressing users," in *Social Robotics*. Springer, 2015, pp. 359–369.

[6] N. Koganti, T. Tamei, T. Matsubara, and T. Shibata, "Estimation of human cloth topological relationship using depth sensor for robotic clothing assistance," in *Proceedings of Conference on Advances In Robotics*. ACM, 2013, pp. 1–6.

[7] N. Koganti, J. G. Ngeo, T. Tomoya, K. Ikeda, and T. Shibata, "Cloth dynamics modeling in latent spaces and its application to robotic clothing assistance," 2015.

[8] K. Yamazaki, R. Oya, K. Nagahama, K. Okada, and M. Inaba, "Bottom dressing by a life-sized humanoid robot provided failure detection and recovery functions," in *Int. Symposium on System Integration (SII)*, 2014, pp. 564–570.

[9] Y. Gao, H. Chang, and Y. Demiris, "Iterative path optimisation for personalised dressing assistance using vision and force information." IEEE, 2016. [Online]. Available: http://hdl.handle.net/10044/1/39009

[10] C. Piciarelli and G. L. Foresti, "On-line trajectory clustering for anomalous events detection," *Pattern Recognition Letters*, vol. 27, no. 15, pp. 1835–1842, 2006.

[11] O. Rosen and A. Medvedev, "An on-line algorithm for anomaly detection in trajectory data," in *American Control Conference (ACC), 2012*. IEEE, 2012, pp. 1117–1122.

[12] A. Jain and C. C. Kemp, "Improving robot manipulation with data-driven object-centric models of everyday forces," *Autonomous Robots*, vol. 35, no. 2-3, pp. 143–159, 2013.

[13] D. Park, Z. Erickson, T. Bhattacharjee, and C. C. Kemp, "Multimodal execution monitoring for anomaly detection during robot manipulation," in *Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2016.

[14] A. Rodriguez, D. Bourne, M. Mason, G. F. Rossano, and J. Wang, "Failure detection in assembly: Force signature analysis," in *Automation Science and Engineering (CASE), 2010 IEEE Conference on*. IEEE, 2010, pp. 210–215.

[15] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *Int. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 3828–3834.

[16] S. Ha and C. K. Liu, "Multiple contact planning for minimizing damage of humanoid falls," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, Sept 2015, pp. 2761–2767.

[17] Y. Bai and C. K. Liu, "Dexterous manipulation using both palm and fingers," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 1560–1565.

[18] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[19] A. Haidu, D. Kohlsdorf, and M. Beetz, "Learning task outcome prediction for robot control from interactive environments," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, USA, 2014.

[20] ——, "Learning action failure models from interactive physics-based simulations," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015.

[21] J. Tan, Z. Xie, B. Boots, and C. K. Liu, "Simulation-based design of dynamic controllers for humanoid balancing," in *Proceedings of The IEEE Conference on Intelligent Robots and Systems (IROS-2016)*, 2016.

[22] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum, "Galileo: Perceiving physical object properties by integrating a physics engine with deep learning," in *Advances in Neural Information Processing Systems 28*.

[23] M. Cusumano-Towner, A. Singh, S. Miller, J. F. O'Brien, and P. Abbeel, "Bringing clothing into desired configurations with limited perception," in *Int. Conf. on Robotics and Automation (ICRA)*, May 2011, pp. 3893–3900.

[24] J. F. O'Brien and J. K. Hodgins, "Graphical modeling and animation of brittle fracture," in *ACM SIGGRAPH*, 1999, pp. 137–146.

[25] R. Narain, A. Samii, and J. F. O'Brien, "Adaptive anisotropic remeshing for cloth simulation," *ACM transactions on graphics (TOG)*, vol. 31, no. 6, p. 152, 2012.

[26] "PhysX physics engine." www.geforce.com/hardware/technology/physx, [Online].

[27] N. Hansen, "The CMA evolution strategy: a tutorial (2009)," *Technische Universitat Berlin, TU Berlin*.

[28] "The general hidden markov model library." http://ghmm.org/, [Online].