Learning to Put On a Knit Cap in a Head-centric Policy Space

Lukas Twardon and Helge Ritter

***Abstract*—Robotic manipulation of such highly deformable objects as clothes is a challenging problem. Robot-assisted dressing adds even more complexity as the garment motions must be aligned with a human body under conditions of strong and variable occlusion. As a step toward solutions for the general task, we consider the example of a dual-arm robot with attached anthropomorphic hands that learns to put a knit cap on a styrofoam head. Our approach avoids modeling the details of the garment and its deformations. Instead, we demonstrate that a head-centric policy parameterization, combined with a suitable objective function for determining the right amount of contact between the cap and the head, enables a direct policy search algorithm to find successful trajectories for this task. We also show how a toy problem that mirrors some of the task constraints can be used to efficiently structure hyperparameter search. Additionally, we suggest a point cloud based algorithm for modeling the head as an ellipsoid which is required for defining the policy space.**

***Index Terms*—Learning and Adaptive Systems, RGB-D Perception, Human Detection and Tracking**

I. Introduction

In recent years, robot-assisted dressing has attracted increasing interest. The task is extremely challenging because of the complex dynamics of the involved objects, together with the difficulty to control the contact-rich interactions between a garment and the body part to be dressed. While solutions for the general task are still out of reach, there is a growing number of works that demonstrate how important parts of the problem can be solved. In most cases, this is done by exploiting simplifications gained through focusing on a narrow subdomain [1]–[11].

The fact that there is a lot of contact between the garment, the robot, and the human body complicates the task. Therefore, we believe that it is important to consider methods which, although not completely task-agnostic, avoid the need to model the item of clothing and its complex interplay with the other physical entities in detail. In this spirit, our previous work [1], [2] focused on reduced topological representations of clothes rather than exhaustive geometric models. We were able to show how this can aid bringing both the robot and the garment into a suitable starting pose for a dressing task. However, occlusion and self-occlusion make it difficult to track the garment during the subsequent contact-rich interactions.

Manuscript received: September 10, 2017; Revised December 7, 2017; Accepted January 3, 2018.

This paper was recommended for publication by Editor Dongheui Lee upon evaluation of the Associate Editor and Reviewers' comments. This work was funded by the German Research Council (DFG), grant EXC 277.

The authors are with the Neuroinformatics Group, EXC Cognitive Interaction Technology (CITEC), Bielefeld University, Germany {ltwardon, helge}@techfak.uni-bielefeld.de

Digital Object Identifier (DOI): see top of this page.

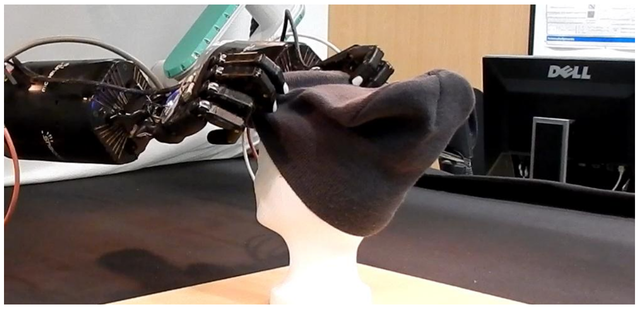


Fig. 1. A robot putting a knit cap on a styrofoam head.

This motivates the present work which explores an extreme strategy that neglects any information about the configuration of the garment *during* task execution and instead uses reinforcement learning to optimize an objective function that is based solely on information about the task performance gathered *after* a policy rollout.

In our study, we focus on the specific scenario of a robot putting a knit cap on a head. This exemplary task involves many of the aforementioned challenges that characterize robot-assisted dressing. As a simplification, we use a styrofoam head model firmly attached to a wooden base. This bypasses most of the safety issues and additional difficulties that would arise from reactive movements of a real human head. It also simplifies systematic studies of the learning approach which relies on repeated interaction with the environment. Nevertheless, the task remains very challenging because the robot is neither provided with a model of the garment nor with the full geometry of the head. Moreover, the robot receives no visual or force feedback during task execution and has to coordinate two kinematically redundant arms with attached five-fingered hands.

Our approach shifts the focus from *online* perception to (i) constructing a suitable policy space *before* and (ii) defining an objective function that is evaluated *after* task execution. For the movement policies to be meaningful in the context of the task, we assume that the robot is initially in a certain configuration and grasps the garment in a defined way. The policy space is then a space of robot trajectories expressed in a reference frame that is based on an ellipsoidal model of the head. Hence, a robot vision algorithm is required that provides an estimate not only of the head pose but also of the scale parameters.

Designing the objective function is challenging as it is not obvious how the distance from a successful reference outcome can be defined. This is because failures to properly finish the execution of a planned trajectory may be frequent in the early stages of learning. The robot could try to maximize the distance covered along the path before failing. This would however favor strategies that avoid garment-head contact completely. Therefore, we use an objective function that enables the robot to learn a trade-off between establishing contact and minimizing the risk of early failure.

The paper makes three mainly conceptual contributions: (i) Definition of a head-centric policy space (using spherical coordinates, safe manipulator poses, and a *B-spline* trajectory parameterization). (ii) A novel objective function for finding the right amount of garment-head contact. (iii) The use of a suitably designed toy problem to support hyperparameter search and a structured analysis of the optimization process without the need for time-consuming real-world samples.

Additionally, the paper provides two algorithmic contributions: (i) A method for head pose and scale estimation with point cloud data employing texture-independent facial features, head proportion heuristics, and ellipsoid fitting. (ii) An algorithm for direct policy search combining *Active-CMA-ES* optimization with a simple surrogate model.

II. Related work

One of the key distinguishing characteristics between different works on robot-assisted dressing is the objective function used because it determines what is to be optimized or learned. Colomé et al. [3] proposed a reinforcement learning framework for the problem of wrapping a scarf around the neck of a mannequin. Since no tight garment openings are involved in this particular task, it was possible to minimize a rather simple objective function using the spatial distance of the scarf from a reference position.

The work from [4] shares with ours the goal to find suitable trajectories for a more complex task. The authors have suggested a path optimization algorithm for putting on a jacket without sleeves. In this task domain, an objective function that aims at avoiding large external forces and, in this way, external resistance has proven to be effective.

While in many works the reference frame is either the garment or the body part to be dressed, Tamei et al. [5] modeled the relationship between a T-shirt's neck opening (equipped with markers) and a mannequin's head through so-called topology coordinates which were used to define a reward function. In [6], the authors have reported some success in markerless estimation of the coordinates.

Contact between a garment and the human body can often be inferred only indirectly from feedback at the robot's end effector. Kapusta et al. [7] distinguish three possible outcomes of the task of pulling a sleeve over a person's forearm: The hand misses the opening to the sleeve, the hand or forearm gets caught in the fabric, or the full forearm successfully enters the sleeve. Their algorithm was able to classify these three outcomes using forces measured at the end effector. Yu et al. [8] employed a simulator to reduce both the classification error and the amount of required real-world samples. Erickson et al.

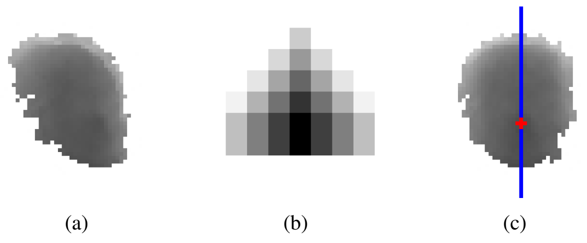


Fig. 2. Detection of the symmetry plane of the face and the tip of the nose using point cloud data. (a) Example range image. (b) Nose template. (c) Optimal axis of symmetry (blue) and nose tip (red).

[9] were even able to infer the areas of contact between the sleeve and the arm.

Furthermore, we should mention that not all works on robot-assisted dressing make use of stochastic optimization or learning. Chance et al. [10] argue that simple HRI strategies could be used instead of complex machine learning methods. The robot in [11] inferred a knowledge base of user constraints and used it for placing a stiff hat on the head.

The reader is referred to [12] for a survey of computer vision methods for human head pose estimation. Many of the existing approaches are texture-based. The styrofoam head in our example does not have much texture, though. The technique described in [13], by contrast, only uses 3D landmark structures which can be extracted from depth data.

III. Ellipsoidal model of the head

We require a geometric representation of the part of the head that is to be covered by the knit cap. For this purpose, we model the styrofoam head as an ellipsoid which we will use for a policy parameterization based on spherical coordinates (Section IV). Our algorithm aims to find the ellipsoid whose upper hemiellipsoid best fits the upper part of the (hairless) head. We propose a point cloud based single-view approach which is (i) independent of texture, shadows, and skin color; and (ii) designed to be used in a realistic robotic setup including a single depth camera with a downward diagonal viewing direction.

In general, an ellipsoid is defined through nine parameters: three for position, three for orientation, and three for scaling. Throughout the paper, $C = (C_x, C_y, C_z)^T$ denotes the ellipsoid center. The axes u, v, w of the ellipsoid object (depicted as red, green, and blue lines in Fig. 3 and 4) are represented by unit vectors $\vec{u}$, $\vec{v}$, $\vec{w}$ and lengths a, b, c. The angles of rotation w.r.t. an extrinsic coordinate system about the intrinsic axes are denoted by α (about u), β (about v), and γ (about w) and applied in reverse order. The difficulty with the single-view setup is that the point cloud of the head is incomplete due to self-occlusion, so that there is no unique solution to the ellipsoid fitting problem. We employ a cascade of heuristics to initialize an ellipsoid whose parameters are then optimized to fit the point cloud.

A. Ellipsoid initialization using landmark structures and head proportions

One way to resolve head pose ambiguities is to detect landmark structures of the face. However, there are often only two stable 3D features: the tip of the nose, and the symmetry plane through the nose. We make the assumption that somewhat more than half of the face is visible to the depth camera. Then, we can use the obtained 3D point cloud to detect both features. In order to do so, we basically follow Spreeuwers' approach [13].

First, we extract those points which belong to the head by detecting the biggest connected region of points in a predefined area. As a second step, we create a set of range images (Fig. 2(a)) by projecting the reduced point cloud to a vertical plane which is rotated in steps of one degree about the vertical axis through the center of mass of the points. To find the symmetry plane, we calculate a symmetry measure for each rotation and each shift along the horizontal axis of the range image (for details, refer to [13]). Then, for all local minima of the symmetry measure, *Normalized Cross Correlation* based nose template matching (Fig. 2(b)) is performed along the vertical mirror axis to find an optimal symmetry plane / nose tip pair (Fig. 2(c)).

We are now able to initialize the model parameter tuple $(C_x, C_y, C_z, \alpha, \beta, \gamma, a, b, c)$ using several heuristics. We set γ to the optimal rotation of the symmetry plane. It can be assumed that the styrofoam head is not tilted to the side, i.e., there is no rotation β. To estimate the remaining orientation parameter α, we employ another technique from [13]. We fit a cylinder with a fixed radius to the points within a defined region around the nose. To this end, we vary both the shift of the cylinder along v and the rotation about u, and set α to the rotation angle that minimizes the mean squared distance of the points from the cylinder.

To obtain initial values for the position and scale parameters, we consider the head proportion heuristics illustrated in Fig. 3. We see that the head can be modeled as an oblate spheroid, i.e., $b = c$. Let P_{nose} be the position of the tip of the nose. We define a plane spanned by $\vec{u}$ and $\vec{v}$ and going through P_{nose}. Then, we compute the distances of the head points above the nose tip from that plane, define d_{vert} to be a high percentile of the distances, and set $b = c = \frac{3}{4} d_{vert}$. The scale parameter a is set to a high percentile of all horizontal distances of the head points from the symmetry plane, and the center of the ellipsoid is initialized as follows:

$$C = P_{nose} - b\vec{v} + \frac{1}{3}c\vec{w} \qquad (1)$$

B. Parameter optimization through point cloud fitting

We must be careful which parameters to optimize when fitting the ellipsoid model to a noisy and incomplete point cloud. The back part of the head, for instance, is usually not represented, so we rely on the heuristic estimates of the scale parameters b and c, and only optimize a. As mentioned above, we assume that $\beta = 0$. The rotation angle α is used to define the coordinate frame of the ellipsoid object, but rotating about u does not change the shape of the oblate spheroid model. Therefore, we can only improve γ. However, the tip of the nose should be on the symmetry plane, so we optimize a rotation angle γ' about an axis w' with direction $\vec{w}$ and going through P_{nose} instead. Moreover, we do not optimize the

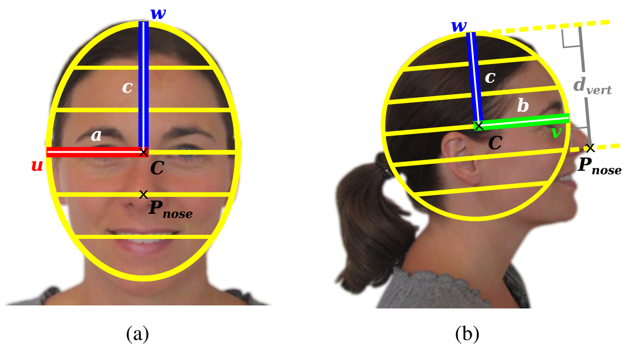


Fig. 3. Head proportion heuristics used for initialization of the ellipsoid model parameters. (a) Front view. (b) Side view.

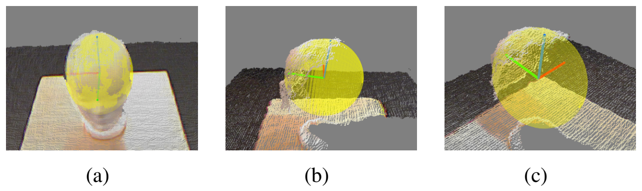


Fig. 4. Ellipsoid model fitted to the noisy and incomplete point cloud of a styrofoam head. The intrinsic coordinate frame is depicted as red (u-axis of length a), green (v-axis of length b), and blue lines (w-axis of length c). (a) Front view. (b) Side view. (c) Diagonal back view.

global position C of the ellipsoid center, but two parameters C_v and C_w representing translation along this new symmetry plane.

The parameter tuple (a, γ', C_v, C_w) is optimized using *Active-CMA-ES* (see Section V). As objective function, we use the mean squared distance of the upper head points (those lying above the old uv-plane) from the ellipsoid. The distances are computed iteratively using the algorithm from [14] because no closed-form solution is known to the point-ellipsoid distance problem. Fig. 4 shows the fitted ellipsoid.

IV. HEAD-CENTRIC POLICY SPACE

When defining the head-centric policy space, we assume a certain garment pose relative to the robot, but we do not model it explicitly. Specifically, the fingers of both hands are assumed to grasp around the boundary of the fabric with the thumbs pointing inside the knit cap (Fig. 5(a) and 5(b)). The robot has to infer strategies to bring the garment into the desired configuration from interaction with the environment. Therefore, the policy space should (i) be restricted to only allow robot poses and actions which are safe for the robot arms and hands, the garment, and the head; and (ii) allow fast learning, e.g., by keeping the dimensionality low.

We encode the end effector position (the contact point of the thumb and the forefinger) in spherical coordinates (θ, ϕ, Δ) w.r.t. the upper hemiellipsoid of the head model. The polar coordinate θ specifies the angle between $-\vec{v}$ and the position vector of the end effector w.r.t. C. The azimuth coordinate ϕ defines the angle between the position vector's orthogonal projection on the uw-plane, and $-\vec{u}$ or $+\vec{u}$ (such that $\phi \leq \frac{\pi}{2}$). To put it less formally, the polar angle $\theta \in [0, \pi]$ runs from the back pole to the front pole of the ellipsoid, whereas the azimuth angle $\phi \in [0, \frac{\pi}{2}]$ determines the "meridian" on which the end effector is located at a given θ. Both coordinates are stretched according to the values of a, b, and c. The third coordinate Δ represents the distance of the finger tips from the ellipsoid model.

To obtain the cartesian coordinates Q of the end effector, we employ

$$\begin{aligned} Q = C &\pm (a + \Delta) sin(\theta) cos(\phi) \vec{u} \\ &- (b + \Delta) cos(\theta) \vec{v} \\ &+ (c + \Delta) sin(\theta) sin(\phi) \vec{w}. \end{aligned} \tag{2}$$

The $\pm$ operator indicates the symmetry of the left and right hand w.r.t. the vw-axis. Restricting the hand motions to be symmetric halves the dimensionality of the policy space. We enforce a minimum distance of 5 cm between the hands to prevent collisions.

The hand orientations are predefined for given positions to ensure safe poses and motions. In robotic grasping, end effector orientations are often specified by two unit vectors: an approach vector, and an orientation vector. We follow a similar convention, replacing the approach vector by a pull vector. The position Q, the orientation vector $\vec{o}$, the pull vector $\vec{p}$, and the normal vector $\vec{n} = \vec{o} \times \vec{p}$ form the intrinsic coordinate frame of the hand (Fig. 5(c)).

A safe end effector orientation can be achieved by choosing the vectors $\vec{o}$ and $\vec{p}$ in such a way that they span a tangent plane to the head ellipsoid, with $\vec{p}$ being oriented along the "meridian" through Q (one of the lines running from the back pole to the front pole). This ensures that (i) the robot hands do not collide with the head (as the palms are roughly tangential to the head), and (ii) when moving the hands from the back to the front of the head, the fingers do not get caught in the fabric, but, in case of too much external resistance, the cap slips out between the thumb and the forefinger in a controlled (orthogonal) way.

The described hand orientations correspond to the following normalized gradients:

$$\begin{aligned} \vec{o} = -\frac{\nabla_\phi Q}{|\nabla_\phi Q|} = \Big[&\pm (a + \Delta) sin(\theta) sin(\phi) \vec{u} \\ &- (c + \Delta) sin(\theta) cos(\phi) \vec{w} \Big] \Big/ |\nabla_\phi Q| \end{aligned} \tag{3}$$

$$\begin{aligned} \vec{p} = \frac{\nabla_\theta Q}{|\nabla_\theta Q|} = \Big[&\pm (a + \Delta) cos(\theta) cos(\phi) \vec{u} \\ &+ (b + \Delta) sin(\theta) \vec{v} \\ &+ (c + \Delta) cos(\theta) sin(\phi) \vec{w} \Big] \Big/ |\nabla_\theta Q| \end{aligned} \tag{4}$$

The vectors $\vec{o}$ and $\vec{p}$ can be made perfectly orthogonal by rotating them in opposite directions about $\vec{n}$. In practice, the elbows or other parts of the robot arms would in some cases exceed the workspace limits to reach the proposed end effector poses. Therefore, we have to partly give up the conditions on the hand orientations. In the back part of the head, we limit the upward angle of the pull vector, but try to keep it on the tangent plane to avoid robot-head collisions. In the front part (where robot-head collisions are not a major issue), the downward angle of the pull vector is limited in such a way that the orientation vector remains essentially unchanged. Moreover, we restrict the horizontal angle between the forearms. The details of the constraint handling procedure are however out of the scope of the paper.

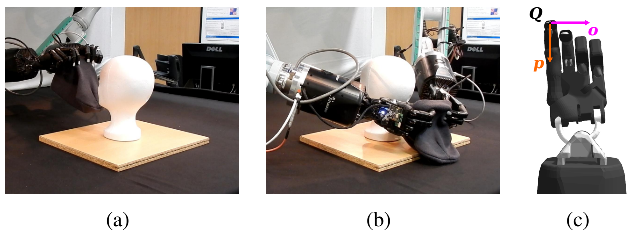


Fig. 5. Configurations of the three entities involved in the example task of putting a knit cap on a styrofoam head: a robot, a garment, and a head. (a) The fingers of the anthropomorphic robot hands grasping around the boundary of the fabric with the thumbs pointing inside the cap. (b) Starting pose of the dressing trajectory. (c) Intrinsic coordinate frame of the hand.

Very low values of Δ involve the risk of collisions with the head, whereas too high values might lead to excessive stretching of the fabric. Therefore, we can further reduce the dimensionality by assuming a fixed distance $\Delta = 1$ cm of the finger tips from the head model. Then, a policy is a parameterized end effector trajectory from the back pole to the front pole in the (θ, ϕ) space. We disregard paths from the front to the back of the head in the present work.

We parameterize the trajectories as *B-splines* which have been used successfully in robotics (e.g., [15]). Specifically, we use *non-periodic uniform B-Splines* $B(t)$ of degree 3 which are fully specified by $N \geq 4$ control points that determine the shape of the path. Other authors have used policy parameterizations which do not directly depend on a parameter t (such as *Dynamic Movement Primitives* [16]). However, using *B-splines*, we can exploit the prior knowledge that the end effector should move continuously from the back to the front of the head, by making the spline a function of the polar angle θ instead of t. Thus, $B(\theta)$ expresses the behavior of the azimuth angle ϕ w.r.t. the running parameter θ. The number of one-dimensional control points N is the only hyperparameter of this parameterization.

V. Policy search

There are two main classes of approaches to reinforcement learning (i.e., learning to accomplish a task by optimizing an objective function through interaction with the environment): value-based approaches and methods for direct policy search [17]. The former approximate a value function which, in principle, has to cover the whole state space, and use this function to infer the optimal policy. The latter tend to be more sample efficient because they only search in the neighborhood of the current policy. Direct policy search methods fall into two groups: gradient-based and gradient-free approaches. Gradient-free policy search is simple and has been shown to be more robust than gradient-based methods regarding initialization, choice of hyperparameters, and noise [18].

In direct policy search, the objective function quantifies the performance feedback the robot gets after each rollout (trajectory execution). For our example task, we suggest the objective function below. The robot receives a fixed reward $r_{success}$ for putting on the knit cap successfully. In the other cases, the objective function aims to support the robot in finding a trade-off between minimizing the risk of early failure, and establishing contact between the fabric and the head. This is expressed as a linear combination of two variables: the polar angle θ_{fail} at which the robot fails (the knit cap slips out between the thumb and the forefinger), and the average azimuth ϕ_{mean} along the path. From evolutionary computation, we adopt the convention that the objective function f is to be minimized.

$$f(P_1,...,P_N) = \begin{cases} -r_{success}, & \text{if successful} \\ f_{shaped}(P_1,...,P_N), & \text{otherwise} \end{cases} \tag{5}$$

with

$$f_{shaped}(P_1,...,P_N) = c_{contact}\phi_{mean} - (1 - c_{contact})\theta_{fail} \tag{6}$$

where $P_1,...,P_N$ denote the control points of the *B-spline* policy parameterization.

The rationale behind the shaped objective function is as follows: The further down the robot hands go (i.e., the lower the azimuth angle ϕ is) when moving around the head, the more contact is established between the fabric of the cap and the head. This is a very rough heuristic contact estimate and could of course be replaced by a more sophisticated force-based measure. If ϕ_{mean} is very low, the first term becomes minimal, but the robot is likely to fail early. If ϕ is constantly high, the knit cap will probably drop down only when releasing it at the end of the trajectory. In this case, the second term is minimized, but the policy is outperformed by every other policy with the same outcome and a slightly lower path. The coefficient $c_{contact}$ is a hyperparameter which controls the relative importance of the objectives.

The objective function is minimized using a black-box optimizer, i.e., a gradient-free optimization algorithm that does not assume any particular knowledge on the structure of f. The *Covariance Matrix Adaptation Evolution Strategy (CMA-ES)* [19] is a general purpose optimizer that has become a quasi-standard in several areas of robotics. In fact, in the present work, we use it for such different optimization tasks as point cloud fitting (Section III-B) and policy search. Like most evolution strategies, *CMA-ES* is a generation-based algorithm. At the beginning of each generation, λ parameter vectors are sampled from a multivariate normal distribution. After that, the robot evaluates f for each sample by executing the trajectories. The weighted mean of the best samples is then considered the new optimal policy. We use the more sample efficient *Active-CMA-ES* variation of the algorithm [20]. The only free parameter of *Active-CMA-ES* is an initial step size parameter σ_0. Moreover, it has been shown that the sample efficiency can be further improved by using previous samples to approximate the landscape of f by means of a surrogate model [21]. We employ a simple k-nearest-neighbor regression model which can be used in different ways: (i) Before each generation, $\lambda_{pre} > \lambda$ samples are drawn from the distribution, and only the λ best samples according to the model are preselected for re-evaluation by the robot. (ii) A certain proportion of real-world rollouts per generation is replaced by surrogate function evaluations.

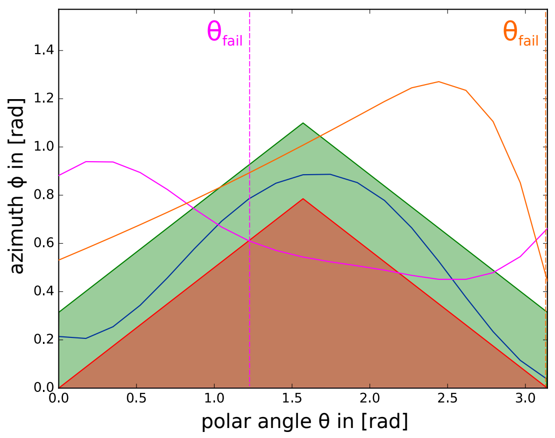


Fig. 6. Toy problem used for hyperparameter search. The regions of successful policies and early failure are depicted in green and red, respectively. The example paths depicted as colored lines represent success (blue), early failure (pink), and failure at the end of the trajectory (orange).

VI. A TOY PROBLEM FOR STRUCTURING HYPERPARAMETER SEARCH

Experiments with a robot are time-consuming and may lead to material wearout. Therefore, it is desirable to reduce the amount of real-world interaction needed for learning a task. Simulation can be useful, but tends to be overly complex, in particular in the absence of an accurate model of the involved materials. In our example, the exact physical properties of the garment, the head, and the fingertips of the robot are unknown. This is one of the reasons why we rely on real-world samples during the actual learning phase. However, hyperparameter search and analyzing the problem structure (e.g., the influence of delayed feedback) require a particularly large number of samples.

Therefore, we have designed a toy problem analogous to the real task, i.e., a problem with the same policy parameterization, identical parameter dimensionality and range, as well as a similarly structured objective function. The exact shape of the landscape of f in the real world is of course not known, but it is reasonable to assume a certain topology: There is a tube-shaped area (the green region in Fig. 6) in the (θ, ϕ) space which contains the successful policies (e.g., the path depicted in blue). Below that area, there is another area (depicted in red) that causes immediate failure when entered by a path (e.g., by the pink one). This represents the case when the knit cap slips out between the fingers because of too much contact between the cap and the head. All other policies (e.g., the path depicted in orange) fail when releasing the cap at the end of the trajectory because the established contact is not sufficient. We believe that the convergence behavior of the optimizer is largely determined by the overall structure of f, whereas the exact geometry plays a minor part. Hence, we are optimistic that hyperparameters found in the toy problem are also a good choice for the real-world problem. To be more realistic, we add a stochastic delay to θ_{fail}. This accounts for delayed sensor or user feedback, and the fact that failures are often caused by suboptimal behavior at an earlier stage of the trajectory.

VII. Experiments

A. Ellipsoid fitting

The first experimental setup included a styrofoam head placed on a table and a *Kinect* depth camera with a downward diagonal viewing angle of about $45°$ w.r.t. the tabletop. This was intended to simulate the situation where a human sits on a chair with the face roughly directed toward the robot. To test the performance of our ellipsoid fitting algorithm in multiple configurations, we randomly placed the head on the tabletop at ten slightly different positions and horizontal orientations (within a range of about ± 10 cm and $\pm 20°$ as measured from the vertical plane along the viewing direction of the camera). In doing so, we varied the structure and amount of self-occlusion of the head w.r.t. the depth camera. We were interested (i) in how much these variations affected the repeatability (standard deviation) of the estimation of the other model parameters (the rotation angle α about u and the scale parameters), and (ii) in how well the model fitted the visible and invisible parts of the upper head.

From visual inspection, we found that the model fitting was reasonable in all trials. The standard deviation of α was $0.95°$. The repeatability of the estimation of a (SD = 0.64 mm) was better than that of the other scale parameter $b = c$ which had a standard deviation of 2.3 mm. Fig. 7 shows the evolution of the root mean squared distance (RMSD) of the visible upper head points from the ellipsoid model, averaged over the ten trials. After 30 *Active-CMA-ES* generations, it has converged to a value of 2.61 mm (SD = 0.27 mm).

To investigate the accuracy of the model w.r.t. both the visible and invisible parts of the head, we created a baseline mesh model of the styrofoam head in advance employing a high-precision laser scanner. Then, we used *ICP* [22] to align the mesh (including the wooden base) with the point cloud, and measured the distances of the upper head vertices from the ellipsoid model. The RMSD averaged over the ten trials was 3.6 mm (SD = 0.75 mm).

B. Toy problem

The toy problem described in Section VI was repeatedly solved to find suitable hyperparameters for the real dressing task, and to analyze the effect of delayed feedback on the optimization algorithm. The considered parameters were: N (the number of control points in the *B-spline* policy parameterization), $c_{contact}$ (the weighting coefficient of the objective function), σ_0 (the initial step size used by the *Active-CMA-ES* optimizer), k (the free parameter of the surrogate model), and the number of surrogate function evaluations per generation (if any). We did not perform an automated hyperparameter search because (i) the performance criterion was not clear (speed of learning, probability of convergence, or something else), and (ii) there was a risk of overfitting the hyperparameters to the simplified toy problem.

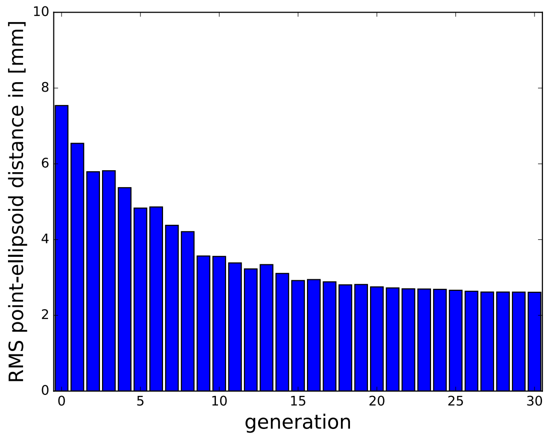


Fig. 7. Root mean squared distance (RMSD) of the upper head points from the ellipsoid model, averaged over ten trials. Each generation comprises $\lambda = 8$ objective function evaluations.

Instead, we performed a manual search and varied one parameter after the other to illustrate and discuss the choices made. We set $r_{success} = 10$ and conducted 1000 sessions (i.e., learning cycles from initialization until convergence) per hyperparameter variation. At each session, the stopping criterion for the optimizer was whether the current optimal policy was within the defined success area (green in Fig. 6). In Fig. 8, we show the amount of successfully finished sessions after a given number of rollouts. We introduced a limit assuming that sessions which were not solved after 300 rollouts had erroneously converged to a local minimum.

The choice of N heavily influenced the speed of convergence. Learning in low-dimensional policy spaces was fast, but also prone to premature convergence (Fig. 8(a)). Besides, the *B-spline* trajectory parameterization can only represent $N - 2$ changes of direction, so the number of control points required to specify successful policies in the real-world is not known in advance. Choosing $N = 6$ was considered a reasonable trade-off. We set $c_{contact} = 0.1$, but interestingly, the parameter had little influence on the convergence behavior, as long as $c_{contact} \leq 0.5$ (Fig. 8(b)). As can be seen from Fig. 8(c), the optimizer was very robust regarding σ_0. We set $\sigma_0 = \frac{\pi}{8}$. Fig. 8(d) shows that using the surrogate model for preselection ($\lambda_{pre} = 60$) greatly speeded up learning. Replacing real objective function evaluations by surrogate evaluations drastically increased the risk of premature convergence to a local minimum, so we decided for a pure preselection strategy. The choice of k in the k-nearest-neighbor regression model played a minor part, but local models (e.g., $k = 2$) performed slightly better (Fig. 8(e)). Our method was robust to moderate Gaussion delay ($\sigma_{delay} \leq \frac{\pi}{4}$) added to θ_{fail} (Fig. 8(f)).

C. Real robot

The robot setup in the dressing experiments consisted of two *Mitsubishi PA-10* arms with attached *Shadow Dexterous Hands* and a *Kinect* depth camera. Before each rollout, the human experimenter placed the cap between the thumbs and the forefingers of the robot in such a way that the fabric

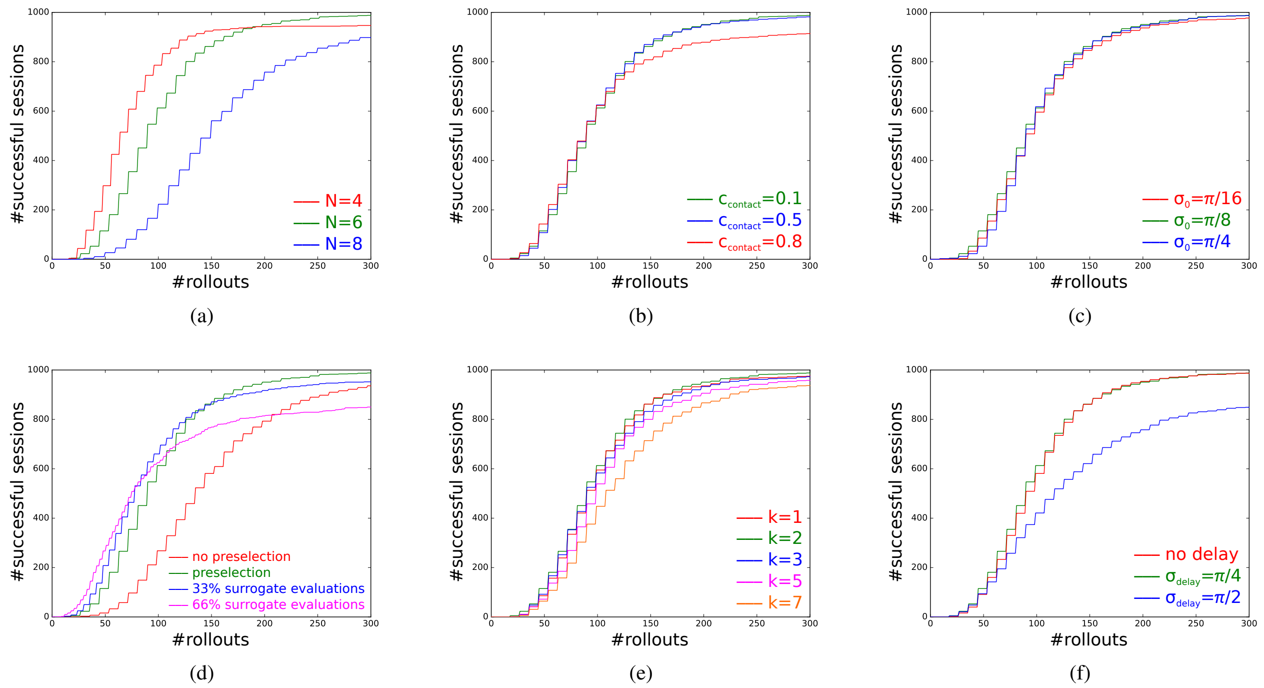


Fig. 8. Amount of successful learning sessions after a given number of rollouts under several hyperparameter variations in the toy problem.

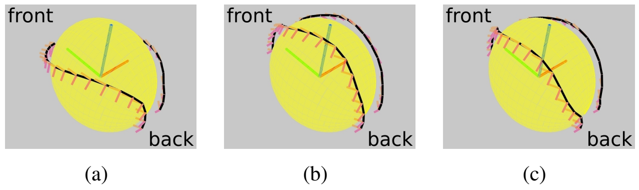


Fig. 9. End effector trajectories for the task of putting a knit cap on a styrofoam head. (a) Example of a path which induces too much object-head contact. (b) Example of a path which avoids object-head contact. (c) Optimized path.

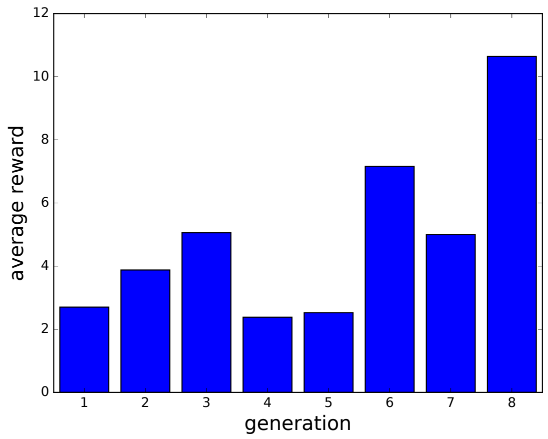


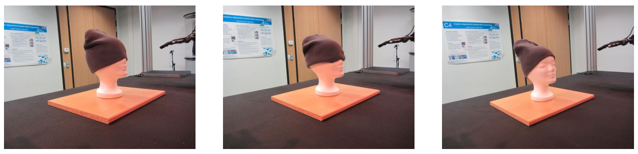
Fig. 10. Average reward $(-f)$ gained per generation. Each generation comprises $\lambda = 9$ trajectory rollouts.

covered the first two phalanges of the thumbs (Fig. 5(a)). This step could be automated in the future using the methods from [1] and [2]. The robot closed the hands and moved to the starting pose at the back of the styrofoam head (Fig. 5(b)). The learning algorithm provided the parameters to be explored next, and the robot started to execute the corresponding trajectory. The tendons in the fingers followed a predefined force profile protecting them against wearout [23], but no forces were applied to counteract slippage of the fabric. If the cap slipped out of the robot's hands, the experimenter immediately pressed a button to stop the execution. After each rollout, the experimenter decided whether it was a *failed*, *successful* (the cap was placed firmly on the head; $r_{success} = 10$), or *perfect* run (the cap was placed firmly on the head, and the edge of the cap was not folded inward; $r_{success} = 20$).

As expected, two types of failure occurred during learning: (i) There was too much contact between the fabric of the cap and the head because the hands went too low (e.g., Fig. 9(a)), and the cap slipped out between the thumb and the forefinger. (ii) The robot was not able to establish enough object-head contact because the path was too high (e.g., Fig. 9(b)).

Fig. 10 shows the average reward $(-f)$ the robot gained per generation. Learning was stopped when more than half of the rollouts of one generation were *successful* or *perfect*. The objective function does not represent the different forms of deformation which occur in this region of the policy space. Therefore, adding more iterations would not help the robot learn to reliably prevent the fabric from folding inward.

After eight generations (72 rollouts), the robot has learned the trade-off visualized in Fig. 9(c). To evaluate the result, we conducted ten trials in which the robot followed the optimized policy. In half of the trials, the knit cap was perfectly placed on the head (Fig. 11(a)), whereas in the other trials, parts of the edge were folded inward (Fig. 11(b)).



(a) (b) (c)

Fig. 11. Possible configurations of the knit cap after an optimal policy rollout. (a) A *perfect* run. The cap was placed firmly on the styrofoam head. (b) A *successful* run. The cap was placed firmly on the head, but the edge was folded inward. (c) Configuration after a finalizing robot motion to uncover the eyes.

VIII. Discussion

We have shown that reinforcement learning in a carefully designed policy space allowed our robot to find successful trajectories for an exemplary but simplified dressing task. Although we did not model the configuration of the garment explicitly, the robot learned to put a knit cap on a styrofoam head. To avoid undesired folds of the fabric, we would however need to adjust the objective function such that deformations are represented in some way.

Since the general dressing assistance problem is still far from being solved, we have considered a very constrained subproblem. As a consequence, our method has a number of limitations. Applying the technique to real humans would pose additional challenges such as handling heads with hair, motion tracking, and robot compliance. Therefore, our study should be seen only as a proof of concept. We were able to reduce the complexity of the problem to a tractable level, but the proposed learning approach might miss some possibly successful policies because we assume a certain grasp pose and limit the search space to paths between two fixed poles.

We used a very problem-specific heuristic for estimating areas of contact, but we speculate that objective functions with similar trade-off structures can be found for other dressing tasks. In the future, we would like to use tactile or force measurements both during learning and to make the optimized system more robust to variation in friction.

Another shortcoming is the fact that the robot tends to pull the cap too far over the face of the styrofoam head. To show that this is not a major limitation, we have implemented a heuristic to uncover the eyes subsequently (Fig. 11(c)): The robot grasps the knit cap at the highest intersection point between the fabric and the symmetry plane of the face, and pulls it backward along a line whose angle and length have been derived empirically.

Thus far, our method does not generalize over caps or heads. However, we believe that spherical coordinates facilitate the design of algorithms with such generalization properties because they scale with the head model. In future work, the robot could learn how a baseline trajectory must be transformed depending on the ellipsoid parameters and the size and deformability of the knit cap opening.

Acknowledgments

We would like to thank Guillaume Walck and Robert Haschke for their helpful input and support, and Jascha Achenbach for creating the baseline mesh model of the head.

References

[1] L. Twardon and H. Ritter, “Interaction skills for a coat-check robot: identifying and handling the boundary components of clothes,” in *ICRA*, 2015, pp. 3682–3688.

[2] ——, “Active boundary component models for robotic dressing assistance,” in *IROS*, 2016, pp. 2811–2818.

[3] A. Colomé, A. Planells, and C. Torras, “A friction-model-based framework for reinforcement learning of robotic tasks in non-rigid environments,” in *ICRA*, 2015, pp. 5649–5654.

[4] Y. Gao, H. J. Chang, and Y. Demiris, “Iterative path optimisation for personalised dressing assistance using vision and force information,” in *IROS*, 2016, pp. 4398–4403.

[5] T. Tamei, T. Matsubara, A. Rai, and T. Shibata, “Reinforcement learning of clothing assistance with a dual-arm robot,” in *Humanoids*, 2011, pp. 733–738.

[6] 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,” in *IROS*, 2015, pp. 3464–3469.

[7] 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*, 2016, pp. 451–458.

[8] W. Yu, A. Kapusta, J. Tan, C. C. Kemp, G. Turk, and C. K. Liu, “Haptic simulation for robot-assisted dressing,” in *ICRA*, 2017, pp. 6044–6051.

[9] Z. Erickson, A. Clegg, W. Yu, G. Turk, C. K. Liu, and C. C. Kemp, “What does the person feel? learning to infer applied forces during robot-assisted dressing,” in *ICRA*, 2017, pp. 6058–6065.

[10] G. Chance, A. Camilleri, B. Winstone, P. Caleb-Solly, and S. Dogramadz, “An assistive robot to support dressing - strategies for planning and error handling,” in *IEEE International Conference on Biomedical Robotics and Biomechatronics*, 2016, pp. 774–780.

[11] S. D. Klee, B. Q. Ferreira, R. Silva, J. P. Costeira, F. S. Melo, and M. Veloso, “Personalized assistance for dressing users,” in *International Conference on Social Robotics*, 2015, pp. 359–369.

[12] E. Murphy-Chutorian and M. M. Trivedi, “Head pose estimation in computer vision: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 607–626, 2009.

[13] L. Spreeuwers, “Fast and accurate 3d face recognition,” *International Journal of Computer Vision*, vol. 93, no. 3, pp. 389–414, 2011.

[14] D. Eberly, “Distance from a point to an ellipse, an ellipsoid, or a hyperellipsoid,” *Geometric Tools, LLC*, 2013.

[15] M. Ruchanurucks, S. Nakaoka, S. Kudoh, and K. Ikeuchi, “Generation of humanoid robot motions with physical constraints using hierarchical b-spline,” in *IROS*, 2005, pp. 1869–1874.

[16] S. Schaal, “Dynamic movement primitives - a framework for motor control in humans and humanoid robotics,” in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[17] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[18] V. Heidrich-Meisner and C. Igel, “Evolution strategies for direct policy search,” in *International Conference on Parallel Problem Solving from Nature*, 2008, pp. 428–437.

[19] N. Hansen, S. D. Müller, and P. Koumoutsakos, “Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es),” *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[20] G. A. Jastrebski and D. V. Arnold, “Improving evolution strategies through active covariance matrix adaptation,” in *IEEE International Conference on Evolutionary Computation*, 2006, pp. 2814–2821.

[21] Y. Jin, “Surrogate-assisted evolutionary computation: Recent advances and future challenges,” *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.

[22] P. Besl and N. D. McKay, “A method for registration of 3-d shapes,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[23] G. Walck, R. Haschke, M. Meier, and H. Ritter, “Robot self-protection by virtual actuator fatigue: Application to tendon-driven dexterous hands during grasping,” in *IROS*, 2017, pp. 2200–2205.