Extracting Bimanual Synergies with Reinforcement Learning

Kevin Sebastian Luck[1] and Heni Ben Amor[1]

***Abstract*— Motor synergies are an important concept in human motor control. Through the co-activation of multiple muscles, complex motion involving many degrees-of-freedom can be generated. However, leveraging this concept in robotics typically entails using human data that may be incompatible for the kinematics of the robot. In this paper, our goal is to enable a robot to identify synergies for low-dimensional control using trial-and-error only. We discuss how synergies can be learned through latent space policy search and introduce an extension of the algorithm for the re-use of previously learned synergies for exploration. The application of the algorithm on a bimanual manipulation task for the Baxter robot shows that performance can be increased by reusing learned synergies intra-task when learning to lift objects. But the reuse of synergies between two tasks with different objects did not lead to a significant improvement.**

I. INTRODUCTION

The ability to manipulate objects in the environment is an important skill for humans and robots and has attracted a large body of research. Motor skills for reaching and grasping, for example, allow robots to physically interact with their immediate surroundings. Yet, generating control policies for these tasks is highly challenging due to the high number of involved degrees-of-freedom (DOF) and the inherent uncertainty.

An important concept in motor control that has helped to overcome these challenges, is the concept of motor synergies [1]; joint co-activations of a set of muscles from a smaller number of neural commands. A combination of a small number of synergies, leads to a large range of different possible movements. In humans, such synergies reduce the dimensionality of the control task and, in turn, reduce the cognitive effort during learning and execution [2], [3], [4]. Similarly, in robotics, synergies have been shown to improve grasp planning performance, while at the same time reducing the computational complexity [5], [6]. However, it is unclear how to identify and extract such synergies for different robot types and morphologies. Existing approaches typically rely on human demonstrations recorded through motion capture systems. Yet, synergies highly depend on the underlying kinematics and mechanics of the system and may not be easily transferred between a human and a robot.

Going beyond uni-manual manipulation, there are also many tasks that require the coordinated use of multiple limbs. Research in human motor control has presented evidence for the existence of bimanual synergies during object manipulation [7]. Different motor synergies may span both arms at the same time, or each arm separately. The ability to identify synergies that affect one or multiple groups of variables at the same time, would allow robots to efficiently learn bimanual manipulation tasks, e.g., pouring, turning a valve, or picking up a box. In this paper, we present a reinforcement learning method that jointly learns motor synergies, as well as control policies for bimanual robot manipulation. Based on our previous discussion of *Group Factor Policy Search* (GrouPS) [8], we will show how sample-efficient reinforcement learning can be performed on a physical robot, without the need for potentially inaccurate simulations. In particular, we will show how GrouPS can autonomously learn dual-arm lifting of objects (see Fig. 1), without relying on prior human demonstrations. The algorithm extracts custom-made synergies that best fit the current robot and task. This is achieved through a combination of dimensionality reduction and policy search. Both, synergies and control policies, are updated while learning, thus exhausting the information provided by the sampling set executed in each iteration. The result is a fast motor skill learning method for tasks that involve the coordination of multiple limbs. In our experiments, the robot autonomously learned object lifting strategies within a relatively small number of trials, i.e., about 1 hour of training time.

The presented method also allows visualizing the extracted bimanual synergies. Visualizing motor synergies enables

Fig. 1: A bimanual robot learns to lift an object while simultaneously identifying synergies among the control variables.

[1]Kevin S. Luck and Heni Ben Amor are with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85281, USA mail@kevin-luck.net, hbamor@asu.edu

users to better understand the couplings between control variables. Additionally, extracted synergies typically form basic movement "building blocks" which can be superimposed to generate a large variety of different behaviors.

In the remainder of this paper, we will first introduce our policy search method and subsequently present its application to bimanual manipulation tasks.

II. RELATED WORK

Human beings and animals are capable of producing a wide variety complex behaviors and motions that involve the coordinated activation of a large number of muscles. This ability poses the question of whether each muscle, or degree-of-freedom is independently controlled by the brain and the central nervous system. Research in neuroscience indicates that groups of muscles may be organized in a modular fashion to form *muscle synergies*. Activating a synergy will jointly co-activate all involved muscles and related joints. The result is a significant reduction in the number of controlled DOFs. In the case of grasping, Santello et al. [2] showed that $\approx 90\%$ of the variance during human grasping can be explained using only three synergies. To this end, human demonstration of grasps were first collected and, then, processed using Principal Component Analysis (PCA). Dimensionality reduction techniques, such as PCA, can uncover the lower-dimensional manifold in which the recorded data points are embedded. Using a similar strategy, other researcher teams have found evidence for synergies in walking [3], running [9], or balancing [4]. Safavynia et al. [10] showed that the acquisition of new motor skills can enable the formation of new synergies. Hence, the composition of synergies is *not static* but can change as a result of repeated practice and reinforcement learning. This also means, that synergies across different subjects may converge towards the same optimal composition that is induced by the task constraints. Hug et al. [11] reported evidence for the formation of similar muscles synergies across expert cyclists over time.

In robotics, motor synergies such as Eigengrasps [5] are typically generated by applying PCA on a set of training data. In the field of grasping and manipulation, this methodology has found wide spread application such as in [12], [6], [13], [14] since it significantly reduces the number of control parameters, while at the same time generating interpretable principal components. Besides grasping, dimensionality reduction was also used to extract synergies for various other robotics tasks. In [15], linear and non-linear manifold learning techniques are used to extract postural synergies for walking and standing-up. The majority of these approaches relies on a large training set of (approximate) solutions, prior simulations, or human demonstrations to perform dimensionality reduction. Even if such data exists, it may drastically bias the search by limiting it to the subspace of initially provided solutions. Especially human demonstrations may be ill-suited for identifying robot synergies. In [16], an approach is present in which robot manipulator can learn synergies from random movements. However, synergies are often required for a specific task at hand. Hence, methods are needed that can generate a set of synergies from a task specification. In [17], we presented a first approach in which synergies can be learned through reinforcement. However, the approach did not allow for the specification of groups of variables within a synergy. In contrast to that, the work presented in the remainder of this paper allows for users to identify specific connected groups of variables, e.g., left arm vs. right arm. Providing this structural information, the algorithm generates synergies that can both model inter-group, as well as intra-group correlations [8]. This is particularly useful for tasks that involve multiple limbs. Synergies can be used to seed the learning algorithm with information about the structure of the manifold to explore. In the case of Group Factor Analysis, a first approach for transfer learning was introduced in [18]. Although, we build upon the same general idea of reusing previously learned factors, we focus in this paper on a reinforcement learning setup rather than a supervised learning setup. Furthermore, we investigate the use of learned synergies for exploration in similar tasks.

III. EXTRACTING SYNERGIES WITH POLICY SEARCH

Synergies for robot motions are typically generated through the application of dimensionality reduction methods on existing data, e.g., joint angles recorded from a human subject. Our approach uses Group Factor Analysis (GFA) as introduced by Klami et al. [19]. However, in contrast to other work that relies on training data, we derive a reinforcement learning method that inherently performs factor analysis. In this section, we will introduce Group Factor Analysis briefly, describe its properties, and then proceed to show how Group Factor Analysis and Policy Search can be combined to yield the Group Factor Policy Search (GrouPS) algorithm. We close this section with the introduction of a new prior distribution for the transformation matrix in GrouPS, which enables the re-use of learned synergies for exploration.

A. Group Factor Analysis for Synergies

Based upon Factor Analysis, GFA assumes that the dimensions of a dataset can be split into groups of variables. The approach inherently assumes the existence of a strong correlation between variables of the same group, e.g., because they form a logical unit such as the leg of a robot, a regions of the brain, or a gene set [19]. The model equation of GFA for M groups reads

$$\mathbf{a}^{(m)} = \mathbf{W}^{(m)}\mathbf{z} + \boldsymbol{\mu}^{(m)} + \boldsymbol{\epsilon}^{(m)}, \tag{1}$$

where $\mathbf{W}^{(m)}$ is the transformation matrix, $\boldsymbol{\mu}^{(m)}$ the mean vector, and isotropic noise $\boldsymbol{\epsilon}^{(m)} \sim \mathcal{N}\left(\mathbf{0}, \tilde{\tau}_{(m)}^{-1}\mathbf{I}\right)$ defined by the precision $\tilde{\tau}_{(m)}$. The random vector $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the same for all groups while the action $\mathbf{a}^{(m)}$ contains the dimensions of the m-th group. Klami and colleagues introduced prior distributions over the parameters of the model equation given above. The most important prior distribution

is

$$p\left(\mathbf{W}|\boldsymbol{\alpha}\right) = \prod_{m=1}^{M}\prod_{k=1}^{K}\prod_{d=1}^{D_m}\mathcal{N}\left(w_{d,k}^{(m)}\middle|0,\alpha_{m,k}^{-1}\right), \qquad (2)$$

which defines a normal distribution over each entry of the transformation matrix $\mathbf{W}$, such that the matrix becomes structurally sparse. The parameter $\alpha_{m,k}$ is specific to each group m and component k and is given by the log-linear model $\log\boldsymbol{\alpha} = \mathbf{U}\mathbf{V}^{\mathrm{T}} + \boldsymbol{\mu}_u\mathbf{1}^{\mathrm{T}} + \mathbf{1}\boldsymbol{\mu}_v^{\mathrm{T}}$. The two matrices $\mathbf{U} \in \mathbb{R}^{M\times R}$ and $\mathbf{V} \in \mathbb{R}^{K\times R}$ are distributed according to a normal distribution with zero mean and λ precision. The rank factor $R \leq \min(M,K)$ influences how sensitive the components are to inter-group correlations. For $R = \min(M,K)$ the log-linear model is equal to a gamma distributed model assuming independent groups [20]. In order to compute the parameters of GFA given a data set, Variational Inference is used while assuming a factorization of parameters according to $p\left(\theta\right) = q(\mathbf{W})q(\boldsymbol{\tau})q(\mathbf{U})q(\mathbf{V})\prod_t^T q(\mathbf{z}_t)$ with q being the approximated distributions. For $q(\mathbf{U})$ and $q(\mathbf{V})$, point estimators are chosen in order to compute the parameters with an optimization method such as L-BFGS [21].

Input: Reward function $R\left(\cdot\right)$ and initializations of parameters. Choose number of latent dimension n and rank r. Set hyper-parameter and define groupings of actions. Set $\hat{\mathbf{W}}$ either to a previously learned synergy or to zero.

while *reward not converged* **do**
- **for** *h=1:H* **do** # Sample H rollouts
  - **for** *t=1:T* **do**
    - $\mathbf{a}_t = \mathbf{W}_i\mathbf{Z}\boldsymbol{\phi} + \mathbf{M}\boldsymbol{\phi} + \mathbf{E}\boldsymbol{\phi}$ with $\mathbf{Z} \sim \mathcal{N}\left(\mathbf{0},\mathbf{I}\right)$ and $\mathbf{E} \sim \mathcal{N}\left(\mathbf{0},\tilde{\boldsymbol{\tau}}\right)$, where $\tilde{\boldsymbol{\tau}}^{(m)} = \tilde{\tau}_m^{-1}\mathbf{I}$
    - Execute action $\mathbf{a}_t$
  - Observe and store reward $R\left(\boldsymbol{\tau}\right)$
- Initialization of q-distribution
- **while** *not converged* **do**
  - Update $q\left(\mathbf{M}\right)$, $q\left(\mathbf{W}\right)$, $q\left(\tilde{\mathbf{Z}}\right)$, $q\left(\boldsymbol{\alpha}\right)$ and $q\left(\tilde{\boldsymbol{\tau}}\right)$
- $\mathbf{M} = \mathbb{E}_{q(\mathbf{M})}\left[\mathbf{M}\right]$
- $\mathbf{W} = \mathbb{E}_{q(\mathbf{W})}\left[\mathbf{W}\right]$ with Eq. (9)
- $\boldsymbol{\alpha} = \mathbb{E}_{q(\boldsymbol{\alpha})}\left[\boldsymbol{\alpha}\right]$ with Eq. (11-14)
- $\tilde{\boldsymbol{\tau}} = \mathbb{E}_{q(\tilde{\boldsymbol{\tau}})}\left[\tilde{\boldsymbol{\tau}}\right]$

Result: Linear weights $\mathbf{M}$ for the feature vector $\boldsymbol{\phi}$, representing the final policy. The columns of $\mathbf{W}$ represent the factors of the latent space.

Algorithm 1: Outline of the Group Factor Policy Search (GrouPS) algorithm. The algorithm is compatible to the previous version presented in [8] since setting $\hat{\mathbf{W}}$ to zero results in the original update equations.

B. Group Factor Policy Search

In its traditional form, GFA requires a dataset of examples in order to extract a low-dimensional manifold. However, in our case we would like to uncover the low-dimensional manifold without prior access to any such dataset. Instead, our goal is to enable a robot to identify synergies for low-dimensional control using trial-and-error only. To this end, we derive a reinforcement algorithm that jointly estimates parameters for dimensionality reduction, as well as a control policy [8].

In our framework, a trajectory consisting of actions $\mathbf{a}$ and states $\mathbf{s}$ is defined by

$$\boldsymbol{\tau} = \left(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T, \mathbf{s}_{T+1}\right) \qquad (3)$$

where T is the number of time steps, and $\mathbf{s}_{T+1}$ is the final state. The objective of policy search is to maximize the expected reward over all possible trajectories given by

$$\mathbb{E}_{p(\boldsymbol{\tau})}\left[r=1\right] = \iint p(\boldsymbol{\tau},\theta)p(r=1|\boldsymbol{\tau})\mathrm{d}\theta\mathrm{d}\boldsymbol{\tau}, \qquad (4)$$

where the reward r is defined as a binary variable with probability $p(r=1|\boldsymbol{\tau}) \propto \exp(-c(\boldsymbol{\tau}))$ and the cost function $c(\cdot)$ [22]. The parameters of the policy are defined by θ. Assuming the Markov property, the probability $p(\boldsymbol{\tau},\theta)$ of the trajectory can be written as

$$p(\boldsymbol{\tau},\theta) = p(\theta)p(\mathbf{s}_1)\prod_{t=1}^{T}p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)\pi(\mathbf{a}_t|\mathbf{s}_t,\theta), \qquad (5)$$

with a prior distribution $p(\theta)$ over the parameters θ, state probabilities $p(\mathbf{s}_1)$ and $p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)$ and the stochastic policy $\pi(\mathbf{a}_t|\mathbf{s}_t,\theta)$.

The model equation for our algorithm Group Factor Policy Search (GrouPS) reads similar to Group Factor Analysis with

$$\mathbf{a}_t^{(m)} = \left(\mathbf{W}^{(m)}\mathbf{Z}_t + \mathbf{M}^{(m)} + \mathbf{E}_t^{(m)}\right)\boldsymbol{\phi}(\mathbf{s}_t,t), \qquad (6)$$

where $\mathbf{a}_t^{(m)} \in \mathbb{R}^{D_m}$ represents the joint or action vector and $\boldsymbol{\phi}(\mathbf{s}_t,t) \in \mathbb{R}^p$ the feature vector containing the basis functions in its entries. The actual values of the basis functions depend on the current state or time step. For notational clarity, we are going to omit states and actions for the feature vector $\boldsymbol{\phi}$ in the remainder of the paper. The matrices $\mathbf{Z}_t$ and $\mathbf{E}_t^{(m)}$ are distributed according to matrix-variate normal distributions: each entry of the latent matrix $\mathbf{Z}_t$ is sampled from a standard normal distribution, whereas the entries of $\mathbf{E}_t^{(m)}$ model the isotropic noise with $\mathcal{N}\left(0,\tilde{\tau}_m^{-1}\right)$. The mean policy is given by matrix $\mathbf{M}^{(m)} \in \mathbb{R}^{D_m\times p}$ whose parameters, i.e. entries, have to be estimated. The transformation matrix is given by $\mathbf{W}^{(m)}$ and contains the extracted synergies in its columns. As shown in [17], the term $\mathbf{Z}_t\boldsymbol{\phi}$ can be rewritten as $\tilde{\mathbf{z}}_t = \mathbf{Z}_t\boldsymbol{\phi} \sim \mathcal{N}\left(\mathbf{0},\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{\phi}\mathbf{I}\right)$, indicating that the noise depends on the values of the basis functions. In the case of normalized basis functions, i.e. $\parallel\boldsymbol{\phi}\parallel_2 = 1$, this term is distributed according to a standard normal distribution. Finally, given above distributions, the stochastic policy $\pi(\mathbf{a}_t|\mathbf{s}_t,\theta)$ (Eq. 5) of Group Factor Analysis can be found with

$$\prod_{m=1}^{M}\mathcal{N}\left(\mathbf{a}_t^{(m)}\middle|\mathbf{W}^{(m)}\tilde{\mathbf{z}}_t + \mathbf{M}^{(m)}\boldsymbol{\phi}, (\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{\phi})\tilde{\tau}_m^{-1}\mathbf{I}\right). \qquad (7)$$

As stated in [8], Group Factor Policy Search does not perform Factor Analysis for each group separately. Rather,

Eq. 7 in combination with the prior distribution of $\mathbf{W}$ from Eq. 2 allows us to uncover components, i.e. columns, in $\mathbf{W}$ with a strong correlation among the groups. For the computation of the parameters we utilize a Variational Inference approach and determine approximated distributions given by the factorization $q(\boldsymbol{\theta}) = q(\tilde{\mathbf{Z}})q(\mathbf{W})q(\tilde{\boldsymbol{\tau}})q(\mathbf{M})q(\boldsymbol{\alpha})$. The final update equations of the algorithm (Alg. 1) and more details regarding Group Factor Policy Search can be found in [8].

C. Transfer Learning with GrouPS

One possibility to incorporate the idea of transfer learning into Group Factor Policy Search is to use the latent space found in one experiment as prior information for a subsequent experiment. The prior of the latent space is given by the normal distribution $p(w_{d,k}^{(m)}|\alpha_{m,k}) = \mathcal{N}(w_{d,k}^{(m)}|0, \alpha_{m,k}^{-1})$. for each entry $w_{d,k}^{(m)}$ of the transformation matrix $W = (W^{1\mathrm{T}}, W^{2\mathrm{T}}, \ldots)^{\mathrm{T}}$. The parameter $\alpha_{m,k}$ is the variance parameter controlling the inter-group flexibility of each dimension, i.e. column of $\mathbf{W}$. We can now incorporate a previously learned latent space by changing the prior of $\mathbf{W}$ to

$$p\left(w_{d,k}^{(m)}|\alpha_{m,k}\right) = \mathcal{N}\left(w_{d,k}^{(m)}\middle|\hat{w}_{d,k}^{(m)}, \alpha_{m,k}^{-1}\right) \tag{8}$$

with $\hat{w}_{d,k}^{(m)}$ being the entries of $\hat{\mathbf{W}}$ learned in a previous experiment. This new prior changes the update equations for both α and $\mathbf{W}$ for the Variational Bayes update. For the transformation matrix $\mathbf{W}$ only the update of the mean [8, Eq. 21] has to be changed to

$$\mu_{m,j}^{W} = \boldsymbol{\Sigma}_{m}^{W} \cdot \mathbb{E}_{p(\boldsymbol{\tau})}\left[\frac{p(r=1|\boldsymbol{\tau})}{\hat{R}} \sum_{t=1}^{T}\left(\frac{\left(a_{t,j}^{(m)} - \mathbb{E}_{\mathbf{M}}\left[\mathbf{m}_{j,:}^{(m)}\right]\boldsymbol{\phi}\right)\mathbb{E}_{\tilde{\mathbf{z}}}\left[\tilde{\mathbf{z}}_{t}^{\mathrm{T}}\right]}{\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{\phi}\mathbb{E}_{\tilde{\tau}_m}\left[\tilde{\tau}_m\right]^{-1}} - \hat{\mathbf{w}}_{j,:}^{(m)}\overline{\overline{\boldsymbol{\alpha}}}_{m,K}\right)\right] \tag{9}$$

while for the log-linear model of $\boldsymbol{\alpha}$ the derivatives change (see Appendix). Introducing this new prior offers the possibility to infuse the algorithm with transformation matrices from different runs or tasks (Alg. 1). The intuition behind sparsity for the transformation matrix $\mathbf{W}$ is now slightly different: instead of driving the entries of the transformation matrix to zero the sparsity prior is trying to maintain the mean and $\boldsymbol{\alpha}$ controls if deviations are added to one group or several groups per column of $\mathbf{W}$.

IV. EXPERIMENTS

In order to evaluate the ability of the GrouPS algorithm to extract meaningful synergies during reinforcement learning, we performed experiments with bimanual manipulation tasks on the Baxter robot. In the following experiments, the robot was tasked to learn how to lift an object with both arms. The goal is to lift the object as high as possible while retaining stability. The initial policy consist of a zeroed M-matrix, i.e., the robot performs no action.

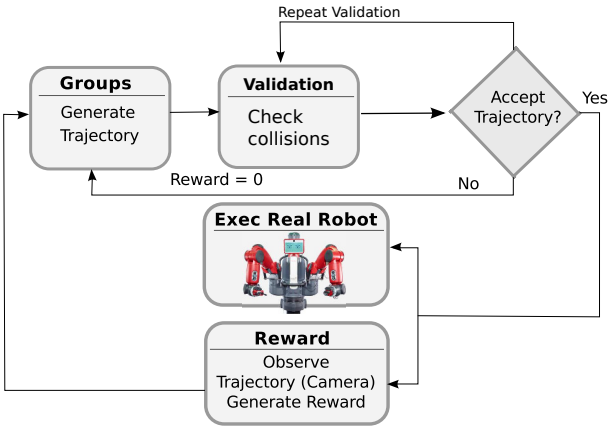


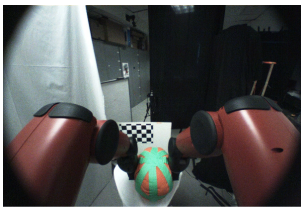
Fig. 2: In each iteration a created sample, i.e. trajectory, is first simulated and to be accepted by the process (or a human operator) for execution. Then the trajectory is executed on the real robot and a reward is generated, which will be used by GrouPS to compute the updates of the policy parameters.

1) Experimental Setup: Each Object was placed on a table of height 77cm in front of a Baxter robot with the same initial position of the arms. Then the GrouPS algorithm was executed for five latent dimensions and rank three over ten iterations. In each iteration but the first, ten samples were newly generated and executed. Each sample constitutes a full lifting trajectory. The samples were included in a sample set of size twenty, from which only the ten best samples were selected for processing while the others were discarded. This is similar to the importance sampling process used in [8], [17], [23]. In the very first iteration, twenty samples were generated. The motivation behind this approach is to allow failure during learning in the real world and compensate for any noise in the reward function. The actions $\mathbf{a}_t$ represent velocities in joint angles of the Baxter robot, thus the action space has 14 dimensions. The whole learning process was performed with real executions on the robot only, and using only rewards generated from the real world (Fig. 2). A kinematic validation process was only used for the purpose of detecting hazardous trajectories before execution (Validation). If a trajectory is deemed dangerous, the process can deny its execution and assign a reward of zero to the trajectory, thus effectively removing the trajectory from the set used for the estimation of the parameters.

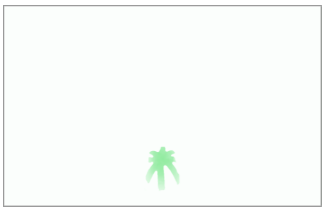
2) Groups: For the experiments on the Baxter robot, four groups were chosen in total, two for each arm. The first group for each arm contains four joints with all rotational and one twisting joint, while the second group consists of three twisting joints (see Table I).

3) Used Basis Functions: As basis functions ϕ, eight radial basis functions, i.e. Gaussian distributions, with a variance of three were used. The mean values were equidistant distributed over the 15 time steps of the trajectory starting with time step -3 and ending with time step 18.

4) Reward Function: As input for the reward function, we chose the height of the object in the picture delivered by



(a) The 800x1280 image delivered by Baxter's head camera.



(b) The image after using color and median filtering.

Fig. 3: The current height of an object is approximated by color filtering.

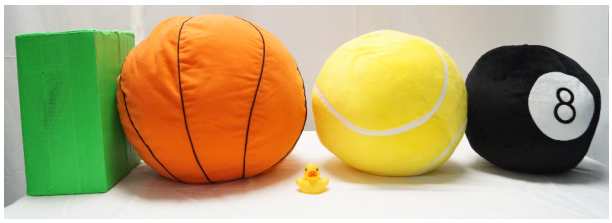


Fig. 4: The four different objects lifted next to each other with the ICRA duck as size reference.

the integrated head camera of the Baxter robot. In order to detect the object during lifting, we used basic color filtering and reflective green tape on the objects (Fig. 3).

For each time step in the trajectory, we create and process one image and detect the maximal height of green pixels in the image. Thus, our reward function does not use the actual height of the object, but the pixel height h in the projected 2D image. Since we employed episodic rewards in our experiments, we used the sum of the exponential cost function resulting in equation $\sum_{t=1}^{T} \exp(-(1-\frac{h_t}{800}))$ for the reward. The height h_t is here normalized with the image height of 800 pixel. Due to the angle of the camera, the reward function is sensitive to horizontal movements of the objects, thus leading to noticeable noise in the generated reward values.

5) Objects: Experiments were performed with four different objects: An orange ball with diameter 33cm, a yellow ball with diameter 27cm, a black ball with diameter 22cm, and a cardboard box with dimensions $26\text{cm} \times 13.5\text{cm} \times 18\text{cm}$ (Fig. 4). While the cardboard box is a rigid object, all balls are soft, non-rigid and deformable. This property makes them particularly challenging to handle for the robot. While the evaluations concentrate on comparisons between the orange ball and box, final results of the remaining two balls will be shown to demonstrate the general capability of GrouPS to solve the task.

6) Time and Sample Size: In all but the first iterations, ten samples were generated while in the very first iteration twenty samples were produced. Thus, the total sample size is 110 samples for one experiment. Each execution of a sample requires about 25 seconds and one complete iteration about four minutes. Accordingly, one experiment requires approximately one hour.

7) Reproducibility: All involved items are internationally available through the company IKEA®. The cardboard box is a standard parcel size with dimensions $26\text{cm} \times 13.5\text{cm} \times 18\text{cm}$ and is used by several international logistics companies. The Matlab code for this experiment together with a connection interface to the Baxter robot is available on our website[1] including all seeds for the random number generators.

8) Experiments: While not the main scope of this paper, a comparison was performed between GrouPS and the *Policy Learning by Weighting Exploration with the Returns* algorithm (PoWER) [23] on the task of lifting the orange ball (Fig. 6). PoWER was used in a configuration with a full covariance matrix over the number of basis functions. PoWER and GrouPS are naturally two very similar algorithms based on stochastic search. While PoWER makes use of an Expectation Maximization framework, the GrouPS algorithm is based on Variational Inference. Also, exploration in PoWER is solely in the high dimensional space without exploiting latent structures for directed exploration while GrouPS incorporates this feature due to its more complex model. Both algorithms made use of the same number of samples over ten iterations.

In order to evaluate the introduced modification of initializing GrouPS with previously learned synergies, three experiments were conducted: First, GrouPS was initialized with synergies found while learning to lift the orange ball (Fig. 9) and then executed four times on the same task. Then, the same initialization was used to learn to lift the box with the Baxter robot. Finally, Groups with random initialization and without pre-initialized mean was applied on the box lifting task and the learned synergies used to initialize GrouPS for lifting the orange ball. All above described experiments were performed four times each.

V. RESULTS

The GrouPS algorithm (without extension) was able to find trajectories for lifting non-rigid objects of different sizes such as the orange ball (Fig. 6), as well as for a rigid cardboard box (Fig. 5). All of the trajectories resulted in a stable final position holding the object in a higher position (Fig. 8). One sequence of synergy matrices $\mathbf{W}$ is shown in Figure 9, where the color of the squares indicates whether the values are negative (gray) or positive (black) and their size correspond to their absolute value. The depicted transformation matrices were computed during an experiment aiming to learn how to lift the orange ball. The extracted synergies can now be found in the columns of $\mathbf{W}$ and replayed directly on the robot for evaluation. Figure 10 shows two synergies found by GrouPS during the learning process which encodes movements for both arms. The first synergy is an opening and closing movement of both arms, while the second synergy showcases a movement to up or down. Both synergies can be combined to generate more complex movements like an upward, closing movement.

[1] http://interactive-robotics.engineering.asu.edu/project/bimanual-synergies/

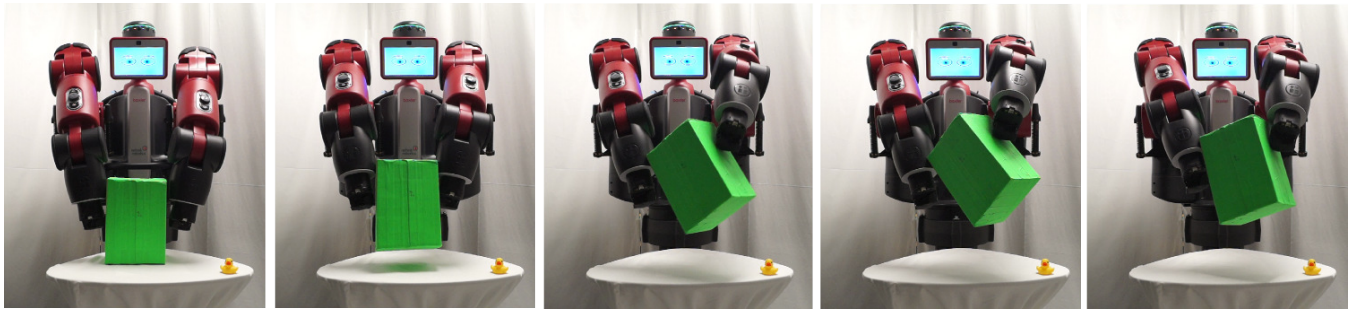


Fig. 5: The final sequence of actions for lifting a cardboard box found by GrouPS.

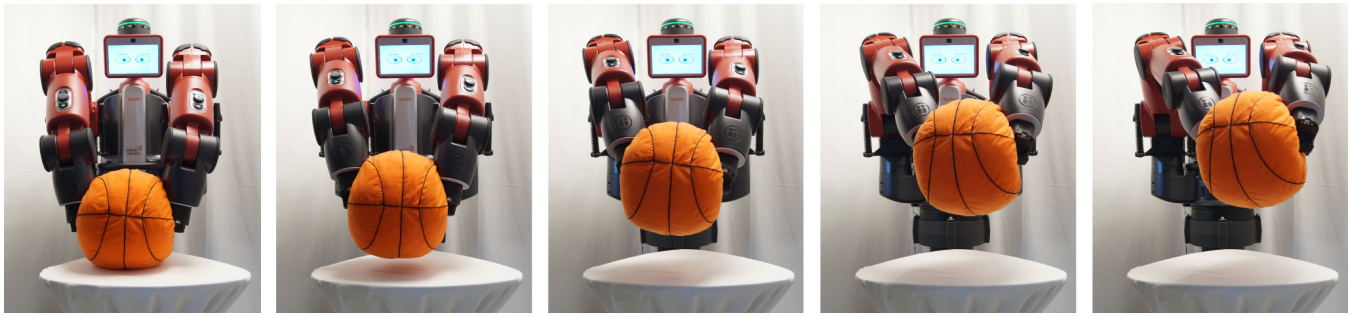


Fig. 6: One final sequence of actions for lifting the orange ball found by GrouPS.

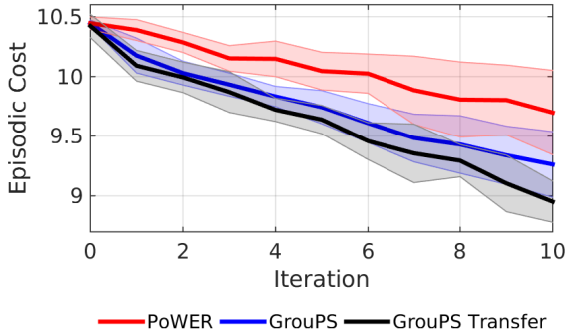


Fig. 7: A comparison between the GrouPS and PoWER algorithm on a lifting task with the orange ball. The presented variant of GrouPS using a synergy matrix *from the same task* outperforms both GrouPS and PoWER. Each algorithm was executed four times and the mean and standard variance were calculated. The vertical axis shows the total cost over 15 time steps and is based on the height of the ball.

Fig. 7 depicts a comparison of results between GrouPS, GrouPS using synergies, and PoWER on the task of lifting the orange ball. While GrouPS outperforms PoWER, pre-initializing GrouPS with learned synergies from the same task leads to another increase in performance. The comparison between using synergies learned from different tasks and GrouPS without modification is presented in Table II which shows that the differences between both variants are not significant.

(a) Orange Ball (b) Yellow Ball

(c) Box (d) Black Ball

Fig. 8: The final pose of trajectories for lifting an object.

VI. DISCUSSION

The purpose of the experiments presented above was to evaluate the ability of the Group Factor Policy Search algorithm to extract not only a successful policy [8], but also uncover latent synergies specific to the task and robot during the learning process. It was found that Group Factor Policy Search is in fact able to uncover synergies (Fig. 9

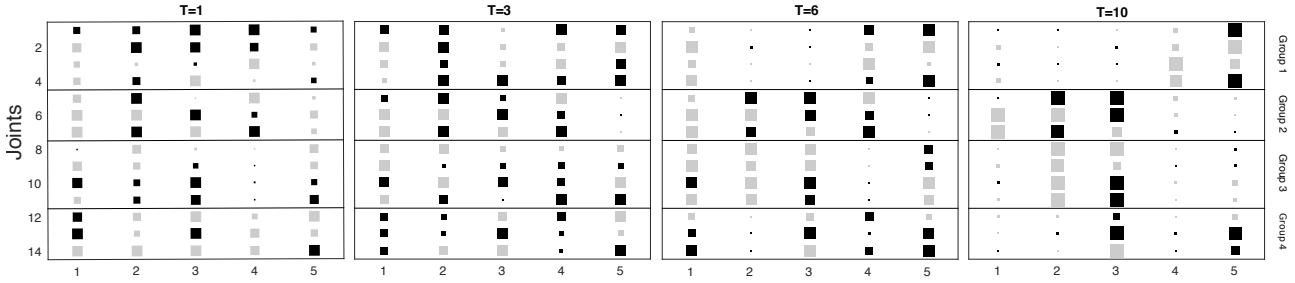


Fig. 9: A sequence of transformation matrices $\mathbf{W}$ computed in each iteration t. The transformation matrix contains the uncovered synergies and is forced to be sparse by the prior distribution on $\mathbf{W}$. The color of the squares represent the sign of the value for this entry: Gray color means negative values and black positive values. The size of the squares are corresponding with the absolute value in such a way that a square is small for small values and the opposite for bigger values.

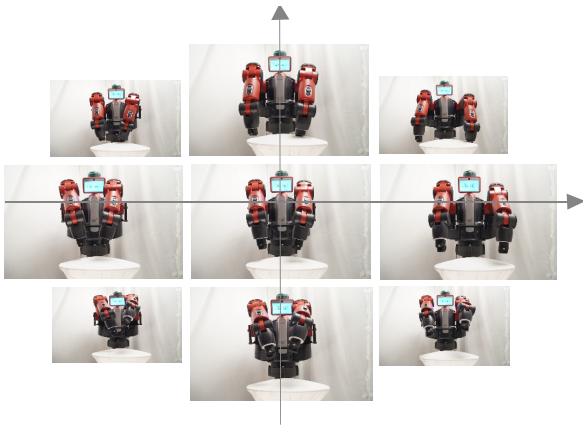


Fig. 10: Two synergies and their combinations found during the execution of the GrouPS algorithm for learning to lift the orange ball. The horizontal axis is showing a synergy of closing and opening motions whereas the vertical axis represents a up- and downwards movement of both arms. Different combinations of those synergies are shown in the four corners of this figure. The video attachment accompanying this paper shows the full execution of those synergies.

TABLE I: The groups for the joints of the Baxter robot chosen for the experiments presented in this paper. The joint names correspond to the technical documentation by Rethink Robotics

Group	Left Arm	Group	Right Arm
1	W1	3	W1
1	S1	3	S1
1	E0	3	E0
1	E1	3	E1
2	W0	4	W0
2	S0	4	S0
2	W2	4	W2

TABLE II: Comparison of GrouPS with the proposed modification for reusing synergies *from different tasks*. For the task of lifting the orange ball one set of synergies, the transformation matrix $\mathbf{W}$, learned in the eight iteration on the box lifting task were used and vice-versa for the box-lifting task. The table shows the final mean cost and standard deviation.

Algorithm	**Cost (Orange Ball)**
GrouPS	9.55 ± 0.19
GrouPS initialized with Synergies	9.47 ± 0.22
Algorithm	**Cost (Box)**
GrouPS	9.26 ± 0.27
GrouPS initialized with Synergies	9.28 ± 0.25

and Fig. 10) while developing a successful policy for a bimanual lifting task. Figure 10 also demonstrates, that these synergies can be combined in a meaningful way to produce new or adapted motions which may be used for similar tasks or initializations of other learning algorithms. Interestingly, the algorithm uncovers two synergies which resemble the nature of the task very well (Fig. 10): While different values of one synergy lead to an opening or closing movement, the second synergy controls the vertical movement of both hands.

An analysis of the transformation matrices in Figure 9 shows that weak correlations between groups disappear over time and strong ones reinforce. However, it can be noted that the transformation matrix is thinning out towards the end of the learning process, very likely due to the convergence to an optimal policy. Thus, it is more likely to find useful synergies in earlier iterations. The most usable synergies were be found in iterations seven and eight.

An variant of the GrouPS algorithm was presented which can make use of uncovered synergies for directed exploration. While it was found that GrouPS initialized with synergies *learned from the same task* indeed leads to an increase in performance (Fig. 7) it is surprising to find that is not the case when using synergies *learned from another task*. This applies for both directions: using synergies from lifting the orange ball to learn to lift the box, and using synergies learned from lifting the box to learn to lift the orange ball.

Both objects, box and ball, are naturally different in shape and so is the optimal strategy for lifting them. Especially the box posed challenges for the robot, since the endeffectors can slide easily along the sides of the box. However, the robot learned to exploit this property over time in order to change the orientation of the box such that one corner of the box points upwards.

VII. CONCLUSIONS

In this paper, we presented a methodology and algorithm for extracting synergies for motor skill learning in robots and using them to accelerate learning. The approach does not require any prior data from human demonstrations or other sources. Instead, we presented a reinforcement learning method that naturally combines dimensionality reduction and policy search. We have shown in experiments with a real-world robot that this combination leads to sample-efficient reinforcement learning. In addition, we have discussed how the generated synergies can be visualized in order to introspect the learning process and better understand the generated coupling of joints.

The potential for speeding up learning in inter- or intra-task transfer using synergies was evaluated. It was found that the presented variant of GrouPS can outperform the base algorithm when reusing synergies from the same task. However, using synergies from a different task did not lead to an increased performance. In future work we will investigate if this insight applies to the general case of transfer learning with GrouPS.

APPENDIX

Since the prior distribution of the transformation matrix $\mathbf{W}$ depends on $\boldsymbol{\alpha}$, the update rule for the log-linear model has also to be updated. Changing the prior of $\mathbf{W}$ leads to a slightly different log-likelihood for the optimization of the parameters as stated in [19] with $\boldsymbol{\Gamma}$ being

$$\boldsymbol{\Gamma} = \mathbb{E}_{\mathbf{W}}\left[(\mathbf{w}_{k,:}^{(m)} - \mathbf{w}_{k,:}^{(m)\prime})(\mathbf{w}_{k,:}^{(m)} - \mathbf{w}_{k,:}^{(m)\prime})^{\mathrm{T}}\right]. \tag{10}$$

The final gradients are then given with

$$\begin{aligned}\frac{\partial L_{\mathbf{U},\mathbf{V}}(\theta)}{\partial \mathbf{U}_{m,:}} &= 2\lambda \mathbf{U}_{m,:} + \sum_{k=1}^{K} D_m \mathbf{V}_{k,:} \\ &- \sum_{K=1}^{K} \boldsymbol{\Gamma} \exp\left(\mathbf{U}_{m,:}\mathbf{V}_{k,:}^{\mathrm{T}} + \boldsymbol{\mu}_{\mathbf{U}_m} + \boldsymbol{\mu}_{\mathbf{V}_k}\right) \mathbf{V}_{k,:},\end{aligned} \tag{11}$$

$$\begin{aligned}\frac{\partial L_{\mathbf{U},\mathbf{V}}(\theta)}{\partial \mathbf{V}_{m,:}} &= 2\lambda \mathbf{V}_{k,:} + \sum_{m=1}^{M} D_m \mathbf{U}_{m,:} \\ &- \sum_{m=1}^{M} \boldsymbol{\Gamma} \exp\left(\mathbf{U}_{m,:}\mathbf{V}_{k,:}^{\mathrm{T}} + \boldsymbol{\mu}_{\mathbf{U}_m} + \boldsymbol{\mu}_{\mathbf{V}_k}\right) \mathbf{U}_{m,:},\end{aligned} \tag{12}$$

$$\begin{aligned}\frac{\partial L_{U,V}(\theta)}{\partial \boldsymbol{\mu}_{\mathbf{U}_m}} &= D_m K \\ &- \sum_{K=1}^{K} \boldsymbol{\Gamma} \exp\left(\mathbf{U}_{m,:}\mathbf{V}_{k,:}^{\mathrm{T}} + \boldsymbol{\mu}_{\mathbf{U}_m} + \boldsymbol{\mu}_{\mathbf{V}_k}\right),\end{aligned} \tag{13}$$

$$\begin{aligned}\frac{\partial L_{U,V}(\theta)}{\partial \boldsymbol{\mu}_{\mathbf{V}_k}} &= D_m M \\ &- \sum_{m=1}^{M} \boldsymbol{\Gamma} \exp\left(\mathbf{U}_{m,:}\mathbf{V}_{k,:}^{\mathrm{T}} + \boldsymbol{\mu}_{\mathbf{U}_m} + \boldsymbol{\mu}_{\mathbf{V}_k}\right).\end{aligned} \tag{14}$$

REFERENCES

[1] N. A. Bernstein, *The co-ordination and regulation of movements*. Pergamon Press, 1967.

[2] M. Santello, M. Flanders, and J. Soechting, "Postural hand synergies for tool use," *The Journal of Neuroscience*, vol. 18, no. 23, 1998.

[3] X. Wang, N. O'Dwyer, and M. Halaki, "A review on the coordinative structure of human walking and the application of principal component analysis," *Neural Regeneration Research*, vol. 8, no. 7, pp. 662–670, 2013.

[4] G. Torres-Oviedo and L. H. Ting, "Subject-specific muscle synergies in human balance control are consistent across different biomechanical contexts," *Journal of Neurophysiology*, vol. 103, no. 6, pp. 3084–3098, 2010.

[5] M. T. Ciocarlie and P. K. Allen, "Hand posture subspaces for dexterous robotic grasping," *The International Journal of Robotics Research*, vol. 28, no. 7, pp. 851–867, 2009.

[6] H. Ben Amor, G. Heumer, B. Jung, and A. Vitzthum, "Grasp synthesis from low-dimensional probabilistic grasp models," *Computer Animation and Virtual Worlds*, vol. 19, no. 3-4, pp. 445–454, 2008.

[7] N. Kang and J. H. Cauraugh, "Force control improvements in chronic stroke: bimanual coordination and motor synergy evidence after coupled bimanual movement training," *Experimental Brain Research*, vol. 232, no. 2, pp. 503–513, 2014.

[8] K. S. Luck, J. Pajarinen, E. Berger, V. Kyrki, and H. B. Amor, "Sparse latent space policy search," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[9] S. Hagio, F. M., and K. M., "Identification of muscle synergies associated with gait transition in humans," *Frontiers in Human Neuroscience*, vol. 9, no. 48, 2015.

[10] S. A. Safavynia, G. Torres-Oviedo, and L. H. Ting, "Muscle synergies: implications for clinical evaluation and rehabilitation of movement," *Topics in spinal cord injury rehabilitation*, vol. 17, no. 1, p. 16, 2011.

[11] F. Hug, N. A. Turpin, A. Guével, and S. Dorel, "Is interindividual variability of emg patterns in trained cyclists related to different muscle synergies?" *Journal of Applied Physiology*, vol. 108, no. 6, pp. 1727–1736, 2010.

[12] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis-a survey," *Trans. Rob.*, vol. 30, no. 2, pp. 289–309, Apr. 2014.

[13] R. Deimel and O. Brock, "A novel type of compliant and underactuated robotic hand for dexterous grasping," *Int. J. Rob. Res.*, vol. 35, no. 1-3, pp. 161–185, Jan. 2016.

[14] S. Andrews and P. G. Kry, "Technical section: Goal directed multi-finger manipulation: Control policies and analysis," *Comput. Graph.*, vol. 37, no. 7, pp. 830–839, Nov. 2013.

[15] H. Ben Amor, E. Berger, D. Vogt, and B. Jung, *Kinesthetic Bootstrapping: Teaching Motor Skills to Humanoid Robots through Physical Interaction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 492–499.

[16] K. C. D. Fu, F. D. Libera, and H. Ishiguro, "Extracting motor synergies from random movements for low-dimensional task-space control of musculoskeletal robots," *Bioinspiration and Biomimetics*, vol. 10, no. 5, p. 056016, 2015.

[17] K. S. Luck, G. Neumann, E. Berger, J. Peters, and H. B. Amor, "Latent space policy search for robotics," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 1434–1440.

[18] E. Leppäaho, "Transfer Learning with Group Factor Analysis," Master's thesis, Aalto University, Finland, 2013.

[19] A. Klami, S. Virtanen, E. Leppaaho, and S. Kaski, "Group Factor Analysis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 9, pp. 2136–2147, 2014.

[20] S. Virtanen, A. Klami, S. A. Khan, and S. Kaski, "Bayesian group factor analysis." in *AISTATS*, 2012, pp. 1269–1277.

[21] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.

[22] M. Toussaint, "Robot trajectory optimization using approximate inference," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 1049–1056.

[23] J. Kober and J. R. Peters, "Policy search for motor primitives in robotics," in *Advances in neural information processing systems*, 2009, pp. 849–856.