Learning to Map Degrees of Freedom for Assistive User Control

Towards an Adaptive DoF-Mapping Control for Assistive Robots

Felix Ferdinand Goldau
German Research Center for Artificial Intelligence (DFKI)
Bremen, Germany
felix.goldau@dfki.de

Udo Frese
University of Bremen
Bremen, Germany
ufrese@uni-bremen.de

ABSTRACT

This paper presents a novel approach to shared control for an assistive robot by adaptively mapping the degrees of freedom (DoFs) for the user to control with a low-dimensional input device. For this, a convolutional neural network interprets camera data of the current situation and outputs a probabilistic description of possible robot motion the user might command.

Applying a novel representation of control modes, the network's output is used to generate individual degrees of freedom of robot motion to be controlled by single DoF of the user's input device. These DoFs are not necessarily equal to the cardinal DoFs of the robot but are instead superimpositions of those, thus allowing motions like diagonal directions or orbiting around a point. This enables the user to perform robot motions previously impossible with such a low-dimensional input device.

The shared control is implemented for a proof-of-concept 2D simulation and evaluated with an initial user study by comparing it to a standard control approach. The results show a functional control which is both subjectively and objectively significantly faster, but subjectively more complex.

CCS CONCEPTS

• **Computer systems organization** → **Robotic control**; Neural networks; • **Human-centered computing** → *Interaction devices*; *Interaction techniques*; • **Social and professional topics** → People with disabilities; **Assistive technologies**.

KEYWORDS

Assistive Robotics, Convolutional Neural Network (CNN), Deep Learning (DL), Human Machine Interface (HMI), Human Robot Interface (HRI), Shared User Control

ACM Reference Format:
Felix Ferdinand Goldau and Udo Frese. 2021. Learning to Map Degrees of Freedom for Assistive User Control: Towards an Adaptive DoF-Mapping Control for Assistive Robots. In *The 14th PErvasive Technologies Related to Assistive Environments Conference (PETRA 2021), June 29-July 2, 2021, Corfu, Greece.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3453892.3453895

PETRA 2021, June 29-July 2, 2021, Corfu, Greece

ACM ISBN 978-1-4503-8792-7/21/06…$15.00
https://doi.org/10.1145/3453892.3453895

1 INTRODUCTION

The general understanding of autonomy and technical systems is something akin to using a computer program to independently control the actuators of a machine or robot to solve a given task. While this might be appropriate for the default industrial scenario, it stands in vast contrast to applications of assistive robots, such as the Kinova Jaco [17, 20], which aim to (re-)enable a person to perform activities of daily living themselves, instead of having them performed by another person or program. However, the manual control of such devices can be very exhausting and taxing for the user due to the complexity of the system and the user's impairments, thus generating a necessity for easier and more accessible methods of control [5].

Some previous work has been done with the aim to automate or ease specific activities of daily living [6, 8, 24]. However, a study investigating the performance and satisfaction of spinal cord injured users of a wheelchair-mounted robotic arm with regards to manual and autonomous control modes showed a higher satisfaction for manual mode users, even though the autonomous mode required less effort [16]. The resulting call for more flexible interfaces coincides with findings by [21], who show the users' requirement to personalise their interaction such that personal standards and social norms are met. A situation with robotic assistance should be as similar as possible to a respective situation without impairments. Therefore, one should be very careful when applying fully automated solutions to such assistive scenarios.

The alternative to a system being controlled by a computer is usually to have it directly or indirectly controlled by a human using a form of Human Computer Interface (HCI) with a keyboard, joystick or similar input device. However, very few devices have sufficient Degrees of Freedom (DoFs) to directly control a robot like the Jaco and those that fulfil this specification require a significant dexterity from the user. For most users of assistive robots, this poses an impossible challenge due to their sicknesses or disabilities. In order to use the remaining mobilities of a user, specific HCIs have been developed [13, 19, 23, 23] which, due to the specifications and limitations, mostly cannot compare to the default control interfaces when it comes to their output DoFs. For example, the Jaco requires at least seven DoFs (three for positioning, three for rotation, and one for grasping), whereas input devices such as Eye-Trackers [23], Chin- or Tongue-Mouses [10] only provide two. Even the robot's joystick only provides a maximum of three DoF to be controlled at once, with buttons allowing to switch between different control modes (cf. [1, 13, 18]). An extensive literature review regarding functionality and performance of assistive robots concluded in a call to "develop a two-way user interface between higher dexterity

Authors´ accepted version. https://doi.org/10.1145/3453892.3453895

[robots] that could be operated by fewer [DoFs] from end-users", whilst keeping the users in control, as desired [3].

Various forms of shared user control exist, where the systems utilise a combination of input from the user and the output of a computer program. For example, [25] initially lets the user control only the translational DoFs of a robot arm, whilst automatically handling rotation. Close to a defined target, the system starts blending the user input with an automated grasp approach based on the user-defined position, until finally applying a fully automated grasp action. Based on a literature study on multiple systems using shared control, [2] identifies the detection of user intent as one of the largest problems within this area and calls out for more Machine Learning (ML) in shared control approaches. Following this call, [7] presents a shared control approach for an electric wheelchair passing small doorways, where the user can activate a blend of their commands with a pre-trained ML-generated control.

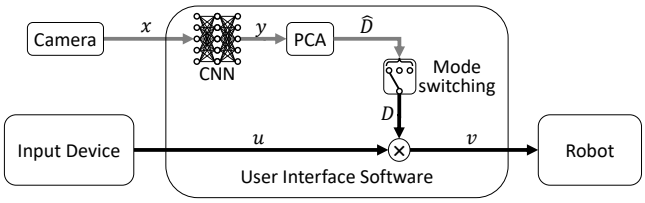

Figure 1: Control pipeline for a user-controlled assistive robot

This paper presents a proof of concept for a novel variation of shared control, where a Deep Learning (DL) based setup evaluates the current situation and adaptively proposes a set of high-dimensional DoFs of robot motion to be controlled by the user's low-DoF input device. Figure 1 shows the corresponding control pipeline: Usually, the user-generated input u is directly mapped to the robot-controlling input v (i.e. D is static), which enables the user to control a single cardinal DoF of the robot (i.e. x-axis, y-axis, z-axis, roll, pitch, yaw) with each DoF of their input device. In cases where the input device has fewer DoFs than the robot control, the user generally has the option to switch between pre-defined modes, thus changing the mapping from input device DoF to robot control DoF (i.e. exchange D). We break this static connection by using a Convolutional Neural Network (CNN) to describe the probabilistic distribution y of intended robot motion v given the camera data x (i.e. the current situation). A Principal Component Analysis (PCA) is applied to calculate a matrix D that adaptively maps the user-generated input to the robot motion, thus portraying modes of control.

The user stays in control; in particular a zero user command u always results in no motion. This eliminates much of the safety concerns of machine learning.

The presented approach enables the DL system not only to suggest the set of cardinal DoFs but also superimpositions of those, thus allowing motions previously impossible with a limited set of input DoFs, such as diagonal paths, orbiting around a point in space or approaching a goal at an angle (cf. Fig. 2). For this paper, the proof-of-concept scenario is limited to a simulated 2D environment with a robot defined by four cardinal DoFs (two positional, one rotational, and grasping). Figure 2 shows the robot with cardinal and adaptive DoFs, both represented by arrows.[1]

Figure 2: The simulated robot with two out of the four cardinal DoFs (left) and two adaptive DoFs (right)

The paper is organised as follows: After a review of previous research to handling the discrepancy of input to output DoFs in Section 2, Section 3 describes our approach in detail, with the simulation environment being described in Section 4. An initial user study is presented in Section 5, with Section 6 discussing the resulting implications and directions for future work.

This paper provides a proof of concept for adaptive DoF mapping in a 2D simulation environment.[2] Its contributions are

- the idea of a novel DL approach to shared control for an assistive robot arm,
- a general representation for DoF-based user control, optionally with modes,
- a 2D simulation environment for proof-of-concept of such methods, and
- an initial user study regarding the usability of such an approach to shared control.

2 RELATED WORK

The default method to controlling a high-DoF device using a low-DoF input device (e.g. controlling an assistive robot arm using a joystick) is mode switching. A single DoF of the input device controls a single cardinal DoF of the robot. Switching the selected mode changes this mapping, such that the same user input now controls a different cardinal DoF of the robot. To the best knowledge of the authors, no shared user control exists that allows the user to control a device along arbitrary online-defined DoFs. However, there are different approaches to mapping user input from a low-DoF input device to a high-DoF system, as well as ML setups that learn autonomous behaviours in a high dimensional environment.

For this paper we use *cardinal DoFs* to describe the set of DoFs defined by, and axis-aligned to, the Cartesian coordinate system of the robot, plus an additional DoF to handle closing the gripper. For a robot with at least six DoFs in 3D space, like the Kinova Jaco, this would be [X-Axis, Y-Axis, Z-Axis, Roll, Pitch, Yaw, Gripper].

Based on their method of user inclusion, it is possible to differentiate control approaches into two categories [11]: In one the user indicates targets and the autonomous system executes the action mostly without user interaction (cf. [26]). The other integrates the user as a direct source of movement control. If a user functions as a direct source of control input, they often have an HCI with low-DoF input device and different control modes. In experiments

[1] Video available at: http://www.informatik.uni-bremen.de/agebv2/downloads/videos/GoldauPetra21.m4v

[2] Resources available at: https://github.com/f371xx/adaptive_dof_mapping_2d

by [11] using an HCI with a standard button-based mode switching setup, more than one-sixth of the total execution time was spent changing the currently selected mode. Within a deterministic simulation environment and a predefined goal, they showed that an automatic mode switching approach already leads to an increase in user satisfaction.

Many manipulation actions require precise positioning. Therefore, when controlling a device towards a goal (e.g. grasping a cup), slight corrections in direction or orientation need to be made. Depending on the environment and perception of the user, this can be a difficult task. For the task of grasping a cup, this would be the precise positioning to not accidentally approach the cup off-center or tip it over with the fingers. Also, if applying a mode switching approach, these small adjustments generally require multiple mode switches, all with very small actual movements of the device within a single mode. To avoid this, research has shown remarkable success with control blending [5], which arbitrates the user's control input with computer generated control, thus allowing the computer to assist the user by avoiding obstacles or supporting with the final approach [4]. However a study has shown that the level of assistance should be customisable by the user to allow for perfect adjustment to the user's needs and abilities, as well as increase user satisfaction [14].

With more complex scenarios and non-deterministic users, multiple goal states can be possible in a given situation (e.g. multiple cups available from which the user can choose which to grasp). For these scenarios, [9] presents a different approach to assistive mode switching: The system isolates possible user intentions and chooses the control mode whose actions will maximise the arbitration of possible user goals in order to assist the ML System in identifying the underlying intention. Once a threshold certainty about the user's intent is surpassed, control blending is applied to assist the user. While this does show promising results, the user's control options are still limited to the cardinal DoFs.

Controlling more complex movements with a low-DoF interface has been realised by predefining sequences within a complex task and using autonomous planners to execute the task. Instead of directly controlling each cardinal DoF of the manipulator, the user utilises their low-DoF interface to define the velocity of the automation and switch between the automated trajectories [15].

A more general option of controlling a robotic device with an HCI is introduced by [22], who propose a neural network to map the sensory readings of an input device to the control signals for a robot. However, within their work they aim to learn an intuitive constant mapping per user and task, therefore restricting the mapping to be static and not adaptive to the situation.

3 MAPPING DEGREES OF FREEDOM

We want to not only do intelligent mode switching but instead loose the system's predefined definitions of DoFs and allow the user to control the robot along DoFs that are regularly redefined based on the current environment and situation.

3.1 Definitions

A DoF d is therefore not limited to the predefined set of cardinal DoFs but instead a vector $d \in \mathbb{R}^n, \|d\|_2 = 1$ in the cardinal coordinate space. This allows for DoFs that are not necessarily axis-aligned to the cardinal coordinate frame, such as moving diagonally or orbiting around a point. A 1-dimensional user input device (e.g. a 1D joystick) could therefore control a high-DoF robot along such an arbitrary n-dimensional DoF.

In the general case, given $u \in \mathbb{R}^m$ as the output of an m-dimensional user input device and $v \in \mathbb{R}^n$ as the n-dimensional robot motion, a matrix $D \in \mathbb{R}^{n \times m}, D = (d_0, d_1, \ldots, d_m)$ can be defined such that

$$v = D \cdot u, \tag{1}$$

where D linearly maps an individual robot motion DoF d_i to each DoF of the user input device (cf. Fig. 1).

As most input devices supply fewer DoFs than the system which they control ($m < n$), a form of mode switching is generally applied. In our notation, this would be equal to exchanging the DoF-mapping matrix D. As an example, Figure 3 shows the static DoF-mapping matrices of the three default control modes of the Kinova Jaco joystick, omitting Drinking mode and the two-finger grasp option.

	Translational mode	Wrist mode	Finger mode
X-Axis	$\begin{pmatrix}1&0&0\end{pmatrix}$	$\begin{pmatrix}0&0&0\end{pmatrix}$	$\begin{pmatrix}0&0&0\end{pmatrix}$
Y-Axis	$\begin{pmatrix}0&1&0\end{pmatrix}$	$\begin{pmatrix}0&0&0\end{pmatrix}$	$\begin{pmatrix}0&0&0\end{pmatrix}$
Z-Axis	$\begin{pmatrix}0&0&1\end{pmatrix}$	$\begin{pmatrix}0&0&0\end{pmatrix}$	$\begin{pmatrix}0&0&0\end{pmatrix}$
Roll	$\begin{pmatrix}0&0&0\end{pmatrix}$	$\begin{pmatrix}1&0&0\end{pmatrix}$	$\begin{pmatrix}0&0&0\end{pmatrix}$
Pitch	$\begin{pmatrix}0&0&0\end{pmatrix}$	$\begin{pmatrix}0&1&0\end{pmatrix}$	$\begin{pmatrix}0&0&0\end{pmatrix}$
Yaw	$\begin{pmatrix}0&0&0\end{pmatrix}$	$\begin{pmatrix}0&0&1\end{pmatrix}$	$\begin{pmatrix}0&0&0\end{pmatrix}$
Gripper	$\begin{pmatrix}0&0&0\end{pmatrix}$	$\begin{pmatrix}0&0&0\end{pmatrix}$	$\begin{pmatrix}1&0&0\end{pmatrix}$

Figure 3: The DoF mapping of the default control modes on the Kinova Jaco joystick

Assuming the use of an input device with sufficient DoFs ($m = n$) and corresponding DoF-map $\hat{D} \in \mathbb{R}^{n \times n}, \hat{D} = (d_0, d_1, \ldots, d_n)$ with linearly independent DoFs d_i and therefore $\text{rank}(\hat{D}) = n$, a user would have complete control of the system without the necessity of switching modes. We name such a mapping a complete DoF-set. If each DoF of an input device directly controls a single cardinal robot DoF using a *complete DoF-set*, $\hat{D}$ would be equal to the identity matrix. For an input device with $m < n$, the mapping for different modes can be generated based on a complete DoF-set by stacking m columns of $\hat{D}$, optionally using zero-padding if $m \nmid n$. This method ensures that the set of modes collectively gives the user the same complete control as an input device with $m = n$ if each column (i.e. DoF) of $\hat{D}$ is represented in at least one mode. For the Kinova Jaco joystick, the underlying identity matrix-shaped $\hat{D}$ can easily be seen in Figure 3.

3.2 Approach

Our approach is to adaptively calculate the mapping D for a low-DoF input device, such that the most likely direction of control is represented by the first DoF in D. We require that the DoFs are perpendicular to one another, such that each of the remaining columns represents the next most likely direction for arbitration. Assuming an optimal mapping, the first DoF should therefore enable the user to manoeuvre the manipulator to their desired position,

Authors´ accepted version. https://doi.org/10.1145/3453892.3453895

with the second DoF allowing them to adjust according to personal preferences. Further options of arbitration exist with the remaining DoFs.

For clarification, please see the following example: A user wants to pour water from an open bottle into a cup. Whilst approaching the bottle, the first DoF initially offers a 3D path command towards the cup, with the second DoF offering an adjustment in the z-direction, thus allowing to grasp the bottle higher or lower. Once in grasping range, these DoFs automatically switch to grasping and rotation around the bottle.

We generate the mapping D from a complete DoF-set $\hat{D}$. If the user wants to perform an action not represented by the current mapping, simple mode switching is applied as a fallback option to give the user the remaining modes for complete control. This can, for example, be automated by switching after a defined idle time, thus allowing to control a complex high-DoF system with a very low-DoF input device. Regarding the update rate of the mapping, internal tests showed the best results when keeping D static while the user is performing any action and, therefore, only updating D when the user gives no input (i.e. zero-input).

3.3 Learning Degrees of Freedom

In order to learn a mapping of DoFs given a certain situation, training data of robot motion is required. As we aim to extend the possibilities of control that are possible with a specific low-DoF input device, it is necessary to take advantage of more complex methods of control (i.e. high-DoF input devices) for the demonstration sequences. Therefore the control pipeline of the deployed implementation in Figure 1 differs from the training setup.

During data generation, using an m-dimensional input device to command an n-dimensional robot with $m \geq n$ allows maximum flexibility and avoids control-based restrictions of robot motions. Applying such a setup, the user interface software requires no mode switching and a simple identity matrix-shaped DoF-mapping D. For data generation and training, the control pipeline is therefore a direct link between input commands u and robot motions v. For our scenario, a joystick-equipped gamepad with continuous user input is used.

This setup allows to intentionally use able-bodied subjects with a very different method of control to generate training data, making it much easier to collect the dataset. Based on this, the CNN can learn a distribution y of arbitrarily complex robot motions v for a specific situation as described by the camera image x. This means for a situation as perceived by the camera image x, the CNN predicts which robot motions v are likely and unlikely to follow, expressed as a distribution of robot motions y.

3.4 Probabilistic view

We view the training data as samples from everyday activities performed by a robot arm. For the probabilistic view discussed here, an outcome of the considered probability space models a snapshot of a random moment of a random everyday activity.

Let X, Y and V be random variables, where X represents the image provided by the camera and V the robot motion. We are interested in the training distribution of V given X ($V|X = x$), i.e. what DoF the user will most likely command in the specific situation evident in the camera data $X = x$. This distribution shall be the basis for selecting an optimal DoF-mapping D and hence the output of the CNN.

Accordingly, we assume $P(V|X = x)$ to exist and follow a multivariate normal distribution $\mathcal{N}_n(\mu, \Sigma)$ with the mean vector $\mu \in \mathbb{R}^n$ and the symmetric, positive definite covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$. Y contains parameters describing μ, Σ and is therefore also a random variable, depending on X.

Treating the control commands in training sequences as samples of V, a feed-forward CNN is used to estimate Y given the camera input X. The link between Y describing the conditional distribution of V and the particular V in the training sample is made by a specific loss (see below), similar to a maximum likelihood loss. We moreover define $\mu = (0, \ldots, 0)^T$ to represent a zero-motion when having the respective zero-input from the user. The CNN therefore only needs to calculate the covariance matrix Σ.

Knowing the distribution of user commands in a given situation allows us to extract a representation of principal components and use these as DoFs for our mapping. We can therefore calculate a complete DoF-set $\hat{D}$ by generating a matrix where each column represents an eigenvector of Σ, sorted in descending order by their respected eigenvalues. Thus, the mode generated by taking the first m columns of $\hat{D}$ as D represents the smallest expected error between the expected (intended) robot motion V and what the user can command with the input device using u. This will be derived in the following.

3.5 Mathematical Derivation of Optimal D

Our DoF-mapping D in (1) has fewer rows n than columns m, hence not every v can be obtained by an appropriate u. However,

$$u = D^{+}v, \tag{2}$$

with D^{+} as the Moore-Penrose-inverse of D gives the input u that produces a robot motion Du as close to v as possible.

With this in mind, we want to obtain the best DoF-mapping $D \in \mathbb{R}^{n \times m}$ given that the intended user command V in this situation is distributed as $V \sim \mathcal{N}_n(0, \Sigma)$. We define *best* by the following requirements:

$$\|Du\|_2 \leq \|u\|_2 \quad \forall u \in \mathbb{R}^m \tag{3}$$

$$\text{minimize } \mathrm{E}\left(\|V - DD^{+}V\|_2^2\right) \tag{4}$$

$$\text{among (4)-optimal } D \text{ minimize } \mathrm{E}\left(\|D^{+}V\|_2^2\right) \tag{5}$$

Requirement (3) forbids too large amplification of the user input, which would make the system hard to control. It also avoids an infinite optimum for D in (5). Requirement (4) expresses our primary goal, namely to minimize the expected difference between the robot motion desired by the user V and the one $DD^{+}V$ that can be commanded via the input device. In general, there are several optimal solutions and among these, we prefer the one that minimizes the command (5).

Note that (4) depends only on the subspace spanned by the columns of D (span D), while (5) depends on D itself.

D can be singular-value decomposed as $D = A\,\mathrm{diag}(\sigma_1, \ldots, \sigma_m)B^T$, $A \in \mathbb{R}^{n \times m}$, $B \in \mathbb{R}^{m \times m}$, with orthonormal A and B. Due to (3),

$\sigma_i \leq 1 \,\forall\, i$, (5) can be rewritten in terms of the σ_i as

$$\mathrm{E}\left(\|D^+V\|_2^2\right) = \mathrm{E}\left(\|B\,\mathrm{diag}(\sigma_1^{-1},...,\sigma_m^{-1})A^TV\|_2^2\right) \tag{6}$$

$$= \mathrm{E}\left(\|\,\mathrm{diag}(\sigma_1^{-1},...,\sigma_m^{-1})A^TV\|_2^2\right) \tag{7}$$

$$= \mathrm{E}\left(\sum_{j=1}^{m}\left(\sigma_j^{-1}A_{\bullet j}^TV\right)^2\right) \tag{8}$$

$$= \sum_{j=1}^{m}\sigma_j^{-2}\,\mathrm{E}\left(\left(A_{\bullet j}^TV\right)^2\right) \tag{9}$$

Now D can be replaced by $D' = AB^T$ (equivalently $\sigma'_j = 1$) which is orthonormal, still meets (3), has the same span as D and hence the same (4). It has at least as large singular values as D and hence an equal or smaller (9). Thus it improves (5).

In conclusion, we can restrict our search for the optimal (4) to orthonormal D, because among the solutions equally good in (4), there is always an orthonormal one at least as good in (5).

We know, that DD^+V is the closest approximation of V in span D. Hence, $V - DD^+V$ is orthogonal to span D and DD^+V. It follows by the Pythagorean theorem, that

$$\|V - DD^+V\|_2^2 = \|V\|_2^2 - \|DD^+V\|_2^2 \tag{10}$$

$$= \|V\|_2^2 - \|D^+V\|_2^2 = \|V\|_2^2 - \|D^TV\|_2^2, \tag{11}$$

where the last two equations are because D is orthonormal. So (4) is equivalent to

$$\text{maximize}_{D\text{ orthonormal}}\,\mathrm{E}\left(\|D^TV\|_2^2\right) = \mathrm{tr}\,\mathrm{Cov}(D^TV) \tag{12}$$

$$= \mathrm{tr}\,D\Sigma D^T. \tag{13}$$

This is a well studied problem in linear algebra and as [12, Corollary 4.3.39] states, the maximum is obtained when D is chosen as orthonormal eigenvectors to the m largest eigenvalues.

This is the mathematical justification of our approach. It can be readily generated by defining the eigenvectors of Σ sorted by descending eigenvalues as a full DoF-set $\hat{D}$. First, D consists of the first m columns of $\hat{D}$. Should the desired robot motion not be (well) covered by these DoFs, the user can switch to the next m columns.

3.6 Implementation

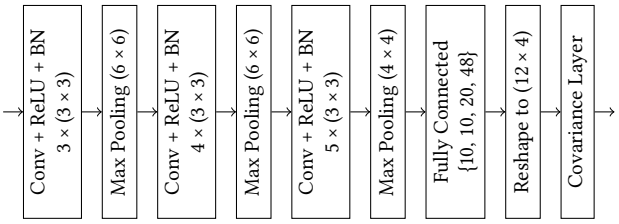

Figure 4: Neural Network

The structure of our CNN is shown in Figure 4. The image-shaped features are processed by Convolutional (Conv) layers with Rectifying Linear Units (ReLU), Batch Normalisation (BN) and max pooling such that fully connected layers can be applied on a flat feature vector. As the final layer, a sample-based method estimates the covariance matrix Σ, with

$$\hat{\Sigma} = \frac{1}{k}\sum_{i=1}^{k}\left(\frac{t_i}{\|t_i\|_2}\right)\left(\frac{t_i}{\|t_i\|_2}\right)^T, \tag{14}$$

$$\Sigma = \varepsilon\, I_n + \hat{\Sigma}, \tag{15}$$

where $\varepsilon > 0$, I_n is the n-dimensional identity matrix and $t_i \in \mathbb{R}^n$ are k samples generated by the previous layer. Each sample is normalised, such that

$$\mathrm{tr}(\hat{\Sigma}) = \sum_{i=1}^{n}\lambda_i = 1, \tag{16}$$

with $\lambda_i, i = 1\ldots, n$ being the eigenvalues of $\hat{\Sigma}$. This method functions as a novel output layer for neural networks, allowing to learn conditioned covariance matrices, guaranteed to be positive definite with defined trace.

We trained our neural network using the loss function $l(v, \Sigma)$

$$l(v, \Sigma) = v^T\Sigma^{-1}v \tag{17}$$

based on maximum log-likelihood loss, to learn a distribution such that the probability of the robot motion $v \in \mathbb{R}^n$ is maximised. In comparison to the standard maximum log likelihood loss, we have omitted constant scaling factors and offsets, as well as the term $\ln|\Sigma|$. Conceptually, this term penalises the covariance matrix for growing too large. As we limit this already by defining the trace of the matrix and internal tests showed better training results without this term, we chose to omit it.

4 SIMULATION ENVIRONMENT

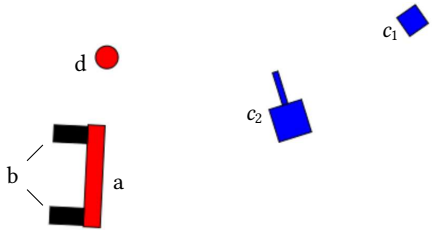

Figure 5: Element overview of the simulation environment

A simple 2D simulation environment was created to develop, test and evaluate the basic principles of adaptive DoF learning as a proof of concept. Figure 5 shows a section of the environment that includes all relevant features. To function as a minimal working example, the user-controlled device is a robotic manipulator (a) able to move forward and backward, sideways, rotate around its center, and close the gripper (b). This sums to a 4-dimensional setting, or 4 DoFs for the user to control. Two blue boxes (c_i) need to be grasped and moved towards the goal marker (d). The physics between the robot, gripper and boxes are handled by a Box2D JavaScript port[3], while the goal marker is solely visual and has no colliding component. At the start of an iteration, all components are positioned randomly. Optionally, the simulation can be toggled,

[3]https://github.com/hecht-software/box2dweb

such that the boxes have spikes on one side (cf. c_2), effectively adding an additional complexity to the scenario, as the gripper can now only grasp the boxes from the side opposite the spike.

Within this environment five options exist to control the robot:

(1) standard control using 8 binary buttons on the keyboard (2 per DoF, one positive and one negative) to control the robot along the cardinal DoFs, therefore allowing only a limited set of directions,
(2) standard control using 4 binary buttons and automated mode switching to cycle through all four cardinal DoFs,
(3) standard control using a joystick with multiple continuous inputs, thus fulfilling the requirement of a high-DoF input device in section 3,
(4) adaptive control using up to 4 binary buttons on the keyboard to steer the robot along up to two DoFs of the neural network-generated DoF-set, and
(5) adaptive control using a joystick with continuous input values based on the same DoF-set as 4.

Option 3 was used for data generation and options 2 and 4 for evaluation. Options 1 and 5 are used for testing and future work respectively.

A mode switching setup is used after five seconds without user input. The currently active DoFs are represented by colored arrows, showing the future state of the robot when following the respective DoF. Figure 2 shows an example situation, with the standard control shown on the left and the adaptive control on the right. When using adaptive control, a server evaluates the current state of the environment and generates the DoF-mapping matrix D for the simulation.

The simulation is implemented in JavaScript, therefore allowing quick and easy website deployment for user studies and evaluations. A variety of settings are customisable within a user interface and allow different deployment strategies for the changing DoFs, thus enabling us to evaluate how much DoF-variety, and therein complexity, users can handle. Internal tests showed the best results when not altering the DoF-set while the user enters any non-zero input and normalising the individual DoFs such that the largest component is always positive. While this prevents the neural network from constantly adjusting the DoFs to create smoother movements, it makes the motion more predictable for the user. The simulation can generate DoF-mappings either using rendered images for CNN-approaches or as an optional alternative using a slim eight-dimensional status vector.

5 USER STUDY

To evaluate the concept of adaptive DoF control, we ran an initial user study based on the 2D simulation system described above. The aim was to compare the standard control (i.e. a static identity matrix-shaped DoF-set) to our adaptive control.

Following the low-DoF HCIs of assistive systems, control option 2 was used for standard control and option 4 for adaptive control. The user input is therefore limited to four binary keyboard buttons to control two DoFs of the robot and having an automated mode switch after every five seconds without user input. The adaptive DoFs are redefined by the network whenever there is no user input, whereas the standard control is based on the cardinal DoFs.

The users were tasked with completing the scenario twelve times: Use the robot to grasp one box after the other and deliver each of them individually to the goal. After every three attempts, the control method switched between standard and adaptive control. After six attempts, spikes were activated for the boxes. To avoid preferences due to training effects, the initial control method was chosen randomly. Before the experiments, each user was shown an introductory video explaining the interface and control methods. During the experiment, the users were kept informed about the currently selected control method. Finally, each user was asked to anonymously evaluate their experience using a questionnaire.

To evaluate the impact of training, a small subset of users were given additional training of roughly ten minutes after their participation in the above-mentioned experiments. After this training, they repeated the adaptive sections of the experiment and gave their evaluation in a similar questionnaire.

5.1 Training

For the adaptive control we trained CNNs for both the scenario with and without spikes based on individual training sets, where the former dataset had spikes activated during data generation. In order to allow complete freedom of motion, the training data for both sets were generated with control option 3. For each training sequence, the simulation started with a random configuration and the users were tasked with grasping the boxes (on the non-spiked side if applicable) and delivering them to the target.

The dataset used for the scenario without spikes was generated by two people and consists of 392 sequences with a total of 29927 datapoints. The network converged in seven epochs.

The dataset used for the scenario with spikes was generated by three people and consists of 488 sequences with a total of 28075 datapoints. The network converged in eight epochs.

5.2 Results

The group of participants consisted of 23 people with a 8/13/1/1 gender split (female/male/diverse/no answer) with ages from 20 to 34 (25.96 ± 3.30). Of those, 2 male and 2 female, ages from 22 to 26, participated in the extended study after training. Regarding their previous experience with keyboard-based controls, the users responded between 1 and 10 (7.04 ± 3.10) on a scale from 1 (never used before) to 10 (usage on a daily basis).

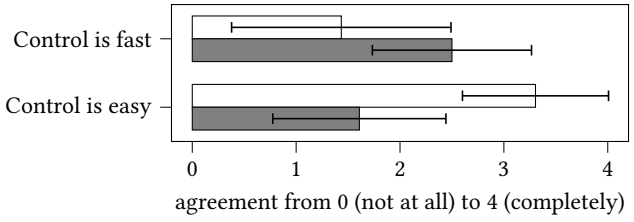

Figure 6: User evaluation of standard (white) and adaptive (grey) control

The users evaluated the speed and ease of both control methods in each scenario (square boxes and boxes with spikes) on a 5 point Likert scale. Figure 6 shows the results in a bar chart with the bar

width representing the mean value and error bars showing the standard deviation.

We evaluated two hypotheses, H_1: *adaptive control is subjectively faster than standard control*, and H_2: *standard control is perceived easier than adaptive control* using dependant two-sampled one-sided t-tests. We were able to reject the null-hypotheses for both H_1 and H_2 and show the differences to be significant (cf. table 1).

On a scale from one to five, the users gave the standard control a rating of 3.17 ± 0.65 and the adaptive control 3.09 ± 1.00. Evaluating the suitability of the presented controls in more complex scenarios on a scale from one to ten, the users gave (4.87 ± 1.79) points for the standard control and 5.83 ± 1.99 for the adaptive control.

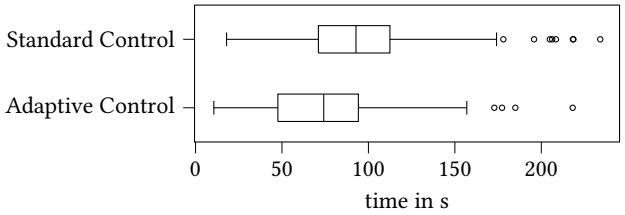

Figure 7: Sequence execution times

Figure 7 shows the distribution of sequence execution times using either standard or adaptive control. While the times vary strongly, it can be observed that the fastest sequences were always performed with the adaptive control, whereas the slowest used standard control. We evaluated hypothesis H_3: *adaptive control is faster than standard control* with a dependant two-sampled one-sided t-test and were able to reject the null-hypothesis and conclude the results to be significant (cf. table 1). This supports the subjective user responses regarding speed and shows that they were able to successfully utilise the subjectively more complex control to achieve lower execution times.

Table 1: T-test results

	M_D	SD_D	t	df	p
H_1	-1.07	1.46	-3.51	44	< 0.001
H_2	1.70	0.86	9.43	44	< 0.001
H_3	21.32	49.96	5.01	274	< 0.001

After additional training, the subset of users performing adaptive control a second time rated the adaptive control faster and easier than before training, while still not rating quite as easy as the standard control. The measured average execution times of the adaptive control sequences after training are lower than before, thus supporting their claim.

5.3 Limitations

The data obtained by this study has been generated entirely online and without any supervision. While this assures real anonymity and avoids personal bias, it cannot be assured that all users completely understood the control methods and the task itself. The participants of the study included a good gender diversity and variety of experience, but only a small age range.

In an optional comment field, some users expressed their desire for a more extensive training and the corresponding expectation that this would greatly benefit the adaptive approach. For the standard control, they also listed the mode switching delay as too long, with some requesting an additional button for switching. Users also complained about not using different subsets of cardinal DoFs (i.e. different definitions of modes). For the adaptive control, there were some complaints about too quick DoF changes, as well as occasional situations where the first and second DoF swapped among each other, therefore missing an opportunity to learn a button-to-action mapping for the user.

In addition to the data presented, five participants generated data, that was deemed flawed and omitted: One person left the simulation idle for several minutes, thus rendering the timings useless; three people seemingly did not follow the instructions by never actually grasping the boxes, and the data of one person was not transmitted completely.

6 CONCLUSION

In this work, we provided proof-of-concept of a novel method for shared control of an assistive robot and evaluated the idea within a 2D simulation environment. For this, we defined a new standardised representation of control modes and introduced a CNN structure to adaptively generate DoF-mappings based on camera data of the current situation and trained it using a specific output layer for conditioned covariance matrices.

The presented application is a simplified proof of concept with a larger scenario as perspective. Even though we expect the largest impact of adaptive DoF-learning in the more complex scenario, the results of our user study show a significant decrease in execution times even in the simple environment. We therefore conclude that adaptive DoF-mapping has the potential to provide a novel interface to assistive robot control and significantly lower task execution times. However, a big challenge for the robot arm application will be communicating the DoFs to the user.

6.1 Future Work

As this work is only a proof of concept in a low-DoF environment, the next steps will be integrating the CNN and concept of control in a more complex 3D environment. It will also be necessary to evaluate the control on more specific tasks of daily living, instead of simple 2D box manipulation.

By addressing more complex environments, an even more flexible interface is necessary. We will therefore evaluate the use of a joystick as an input device for our adaptive control. This will allow users to apply continuous commands, rather than binary button-outputs, to control the robot in the defined modes. This would enable the user to not only control directions of movement, but also control robot velocities.

ACKNOWLEDGMENTS

This work is partially funded by the German BMBF (Bundesministerium für Bildung und Forschung) project AdaMeKoR (FKZ 16SV8534).

REFERENCES

[1] Reem K Al-Halimi and Medhat Moussa. 2016. Performing complex tasks by users with upper-extremity disabilities using a 6-DOF robotic arm: A study. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25, 6 (2016), 686–693.

[2] Brenna D Argall. 2013. Machine Learning for Shared Control with Assistive Machines. In *Proceedings of ICRA Workshop on Autonomous Learning: From Machine Learning to Learning in Real world Autonomous Systems, Karlsruhe, Germany*.

[3] Cheng-Shiu Chung, Hongwu Wang, and Rory A Cooper. 2013. Functional assessment and performance evaluation for assistive robotic manipulators: Literature review. *The journal of spinal cord medicine* 36, 4 (2013), 273–289.

[4] Anca D Dragan and Siddhartha S Srinivasa. 2012. *Formalizing assistive teleoperation*. MIT Press, July.

[5] BJF Driessen, TK Ten Kate, F Liefhebber, AHG Versluis, and JA Van Woerden. 2005. Collaborative control of the manus manipulator. *Universal Access in the Information Society* 4, 2 (2005), 165–173.

[6] D. Gallenberger, T. Bhattacharjee, Y. Kim, and S. S. Srinivasa. 2019. Transfer Depends on Acquisition: Analyzing Manipulation Strategies for Robotic Feeding. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 267–276. https://doi.org/10.1109/HRI.2019.8673309

[7] Aditya Goil, Matthew Derry, and Brenna D Argall. 2013. Using machine learning to blend human and robot controls for assisted wheelchair navigation. In *2013 IEEE 13th International Conference on Rehabilitation Robotics (ICORR)*. IEEE, 1–6.

[8] Felix Ferdinand Goldau, Tejas Kumar Shastha, Maria Kyrarini, and Axel Gräser. 2019. Autonomous Multi-Sensory Robotic Assistant for a Drinking Task. In *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*. IEEE, 210–216.

[9] Deepak Edakkattil Gopinath and Brenna Argall. 2017. Mode Switch Assistance To Maximize Human Intent Disambiguation.. In *Robotics: Science and Systems*.

[10] Axel Graser, Torsten Heyer, Leila Fotoohi, Uwe Lange, Henning Kampe, Bashar Enjarini, Stefan Heyer, Christos Fragkopoulos, and Danijela Ristic-Durrant. 2013. A supportive friend at work: Robotic workplace assistance for the disabled. *IEEE Robotics & Automation Magazine* 20, 4 (2013), 148–159.

[11] Laura V Herlant, Rachel M Holladay, and Siddhartha S Srinivasa. 2016. Assistive teleoperation of robot arms via automatic time-optimal mode switching. In *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*. IEEE Press, 35–42.

[12] Roger A Horn and Charles R Johnson. 2012. *Matrix analysis*. Cambridge university press.

[13] Anja Jackowski, Marion Gebhard, and Roland Thietje. 2017. Head Motion and Head Gesture-Based Robot Control: A Usability Study. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 26, 1 (2017), 161–170.

[14] Siddarth Jain and Brenna Argall. 2019. Probabilistic human intent recognition for shared autonomy in assistive robotics. *ACM Transactions on Human-Robot Interaction (THRI)* 9, 1 (2019), 1–23.

[15] Siddarth Jain, Ali Farshchiansadegh, Alexander Broad, Farnaz Abdollahi, Ferdinando Mussa-Ivaldi, and Brenna Argall. 2015. Assistive robotic manipulation through shared autonomy and a body-machine interface. In *2015 IEEE international conference on rehabilitation robotics (ICORR)*. IEEE, 526–531.

[16] D. Kim, R. Hazlett-Knudsen, H. Culver-Godfrey, G. Rucks, T. Cunningham, D. Portee, J. Bricout, Z. Wang, and A. Behal. 2012. How Autonomy Impacts Performance and Satisfaction: Results From a Study With Spinal Cord Injured Subjects Using an Assistive Robot. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 42, 1 (2012), 2–14. https://doi.org/10.1109/TSMCA.2011.2159589

[17] Kinova Inc. 2019. *KINOVA JACO Assistive robotic arm*. Retrieved 2019-08-19 from https://www.kinovarobotics.com/en/products/assistive-technologies/kinova-jaco-assistive-robotic-arm

[18] Kinova Inc. 2019. *User Guide Kinova Gen2 Ultra lightweight robot*.

[19] Maria Kyrarini, Adrian Leu, Danijela Ristić-Durrant, Axel Gräser, Anja Jackowski, Marion Gebhard, Jochen Nelles, Christina Bröhl, Christopher Brandl, Alexander Mertens, et al. 2016. Human-Robot Synergy for cooperative robots. *Facta Universitatis, Series: Automatic Control and Robotics* 15, 3 (2016), 187–204.

[20] Veronique Maheu, Philippe S Archambault, Julie Frappier, and François Routhier. 2011. Evaluation of the JACO robotic arm: Clinico-economic study for powered wheelchair users with upper-extremity disabilities. In *2011 IEEE International Conference on Rehabilitation Robotics*. IEEE, 1–5.

[21] Bente Martinsen, Ingegerd Harder, and Fin Biering-Sorensen. 2008. The meaning of assisted feeding for people living with spinal cord injury: a phenomenological study. *Journal of advanced nursing* 62, 5 (2008), 533–540.

[22] Jartuwat Rajruangrabin, Isura Ranatunga, and Dan O Popa. 2012. Adaptive interface mapping for intuitive teleoperation of multi-dof robots. In *Conference Towards Autonomous Robotic Systems*. Springer, 49–60.

[23] Jeroen Schäfer and Marion Gebhard. 2019. Feasibility analysis of sensor modalities to control a robot with eye and head movements for assistive tasks. In *Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments*. ACM, 482–488.

[24] Katherine M Tsui, Dae-Jin Kim, Aman Behal, David Kontak, and Holly A Yanco. 2011. "I want that": Human-in-the-loop control of a wheelchair-mounted robotic arm. *Applied Bionics and Biomechanics* 8, 1 (2011), 127–147.

[25] Jörn Vogel, Katharina Hertkorn, Rohit U Menon, and Máximo A Roa. 2016. Flexible, semi-autonomous grasping for assistive robotics. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4872–4879.

[26] Ming Zhong, Yanqiang Zhang, Xi Yang, Yufeng Yao, Junlong Guo, Yaping Wang, and Yaxin Liu. 2019. Assistive Grasping Based on Laser-point Detection with Application to Wheelchair-mounted Robotic Arms. *Sensors* 19, 2 (2019), 303.