Simulation-Aided Handover Prediction From Video Using Recurrent Image-to-Motion Networks

Matija Mavsar, Barry Ridge, Rok Pahič, Jun Morimoto, *Member, IEEE*, and Aleš Ude, *Member, IEEE*

***Abstract*— Recent advances in deep neural networks have opened up new possibilities for visuomotor robot learning. In the context of human–robot or robot–robot collaboration, such networks can be trained to predict future poses and this information can be used to improve the dynamics of cooperative tasks. This is important, both in terms of realizing various cooperative behaviors, and for ensuring safety. In this article, we propose a recurrent neural architecture, capable of transforming variable-length input motion videos into a set of parameters describing a robot trajectory, where predictions can be made after receiving only a few frames. A simulation environment is utilized to expand the training database and to improve generalization capability of the network. The resulting architecture demonstrates good accuracy when predicting handover trajectories, with models trained on synthetic and real data showing better performance than when trained on real or simulated data only. The computed trajectories enable the execution of handover tasks with uncalibrated robots, which was verified in an experiment with two real robots.**

***Index Terms*— Dynamic movement primitives (DMPs), handover, machine vision, recurrent neural networks (RNNs), robot learning, simulation.**

Manuscript received 2 July 2021; revised 29 December 2021; accepted 9 May 2022. Date of publication 30 May 2022; date of current version 5 January 2024. This work was supported in part by the Slovenian Research Agency through the Program Group Automation, Robotics, and Biocybernetics under Grant P2-0076 and Young Researcher Grant PR-09781; in part by the European Union's Horizon 2020 Grant CoLLaboratE under Grant 820767; in part by the Project under Grant JPNP20006; in part by the New Energy and Industrial Technology Development Organization (NEDO); in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP16H06565; in part by the Japan Science and Technology Agency (JST) Mirai Program under Grant JPMJMI18B8 and Grant JPMJMI21B1; and in part by the Tateishi Science and Technology Foundation. *(Corresponding author: Matija Mavsar.)*

Matija Mavsar is with the Humanoid and Cognitive Robotics Laboratory, Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, 1000 Ljubljana, Slovenia, and also with the Faculty of Electrical Engineering, University of Ljubljana, 1000 Ljubljana, Slovenia (e-mail: matija.mavsar@ijs.si).

Barry Ridge is with the Humanoid and Cognitive Robotics Laboratory, Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, 1000 Ljubljana, Slovenia, and also with ATR Computational Neuroscience Laboratories, Department of Brain-Robot Interface, Advanced Telecommunications Research Institute International, Kyoto 619-0237, Japan (e-mail: barry@barr.ai).

Rok Pahič is with the Humanoid and Cognitive Robotics Laboratory, Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, 1000 Ljubljana, Slovenia (e-mail: rok.pahic@ijs.si).

Jun Morimoto is with ATR Computational Neuroscience Laboratories, Department of Brain-Robot Interface, Advanced Telecommunications Research Institute International, Kyoto 619-0237, Japan, and also with the Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan(e-mail: xmorimo@atr.jp).

Aleš Ude is with the Humanoid and Cognitive Robotics Laboratory, Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, 1000 Ljubljana, Slovenia, also with ATR Computational Neuroscience Laboratories, Department of Brain-Robot Interface, Advanced Telecommunications Research Institute International, Kyoto 619-0237, Japan, and also with the Faculty of Electrical Engineering, University of Ljubljana, 1000 Ljubljana, Slovenia (e-mail: ales.ude@ijs.si).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2022.3175720.

Digital Object Identifier 10.1109/TNNLS.2022.3175720

I. Introduction

In THE past decade, robot workspaces have been moving from closed areas with known and predictable conditions into increasingly complex environments with limited amounts of *a priori* information. Many recent works focus on human–robot collaboration, where the aim is to enable robots and humans to work together in close proximity [1]. For humans and robots to effectively work together in unknown environments, a system for robust human motion prediction is required, which guarantees safety and predictability. Machine vision can provide the necessary information to respond to changes by extracting the most important data while being invariant to less important and random phenomena.

Object handover is a collaborative action where an agent, the giver, gives an object to another agent, the receiver [2]. The handover task typically consists of a pre-handover phase and a physical object exchange phase. Aspects such as communication, grasping, motion planning and control must be considered to achieve optimal performance. In the pre-handover phase, the receiver observes the motion of the giver and computes the appropriate receiving motion trajectory. Accurate prediction of motion trajectories from the partially observed giver motion (captured as an red-green-blue and depth (RGB-D) video) is the focus of this article. Thus, the article focuses on the pre-handover phase.

Obtaining meaningful information from a stream of digital images can be facilitated by neural networks due to their efficiency in modeling different nonlinear processes [3]–[5]. Because of the dynamic attributes of human–robot workspaces, neural structures that incorporate time dependency are especially well suited for the task of predicting the movements of a person or robot, with recurrent neural networks (RNNs) being a good choice. One of the most widely used types of RNNs is the long short-term memory (LSTM) network [6]. LSTM networks can efficiently process temporal sequences [7], such as RGB-D videos, while exploiting prior observations to make future predictions. Since gathering large amounts of training data can be expensive and time-consuming, especially in robotics [8], simulation is being increasingly used for data generation. The challenge lies in transferring models trained with simulated data to the real world [9], [10].

In this work, we focus on predicting motion trajectories from RGB-D videos containing the giver motion. Current state-of-the-art methods either rely on the body postures of the giver [11]–[13], or can only predict the final pose or an action description label [12], [14]. Conversely, we propose a Recurrent Image-to-Motion Encoder–Decoder Neural Network (RIMEDNet) capable of translating variable-length RGB-D videos of a giver (human or robot) into a predicted trajectory during a handover task (see Fig. 1). RIMEDNet

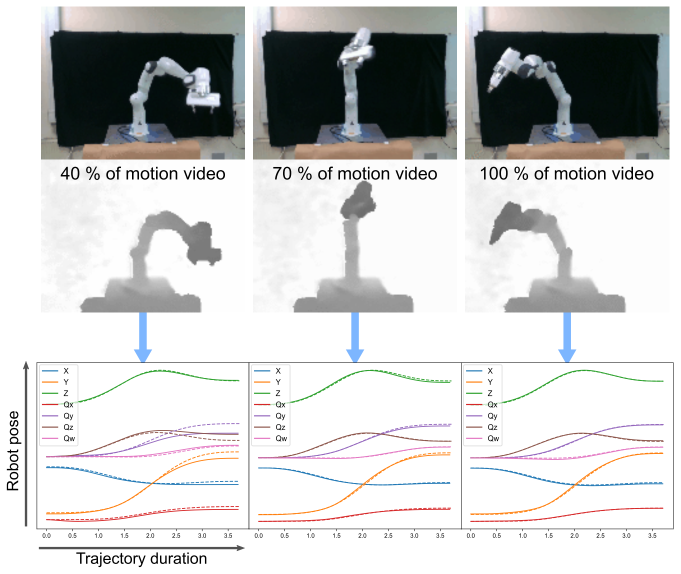

Fig. 1. Proposed RNN can predict joint or Cartesian robot trajectories (bottom row) after processing only a fraction of the complete RGB-D video of robot motion (upper two rows). Predictions are updated online with each received frame. Note that the prediction accuracy improves as more data become available.

trained with the appropriate data can predict either the motion trajectory of the giver or the motion trajectory of the receiver. It consists of convolutional layers to extract spatial representations, LSTM layers to analyze temporal dependencies, and fully connected layers to compute the motion parameters. Initial predictions of the entire output trajectory can be made immediately during the early stages of giver–receiver interaction based on only a fraction of the entire RGB-D input video. The accuracy of the predicted trajectory increases as more images are processed. This way, the receiver can anticipate the handover action and respond accordingly, e.g., by moving toward the predicted pose of the handover while avoiding the giver's path. To reduce the amount of real data required for training, we make use of simulation and domain randomization to acquire an extensive and highly randomized synthetic dataset. The proposed approach is compared to state-of-the-art architectures and evaluated in robot-to-robot handover experiments, while it may also be extended for use in human-to-robot handovers. None of the methods described in a recent survey of object handover tasks [2] utilizes such an approach.

The main contributions of the article are: 1) a novel recurrent encoder–decoder neural network architecture for online prediction of entire robot trajectories encoded as third-order dynamic movement primitives (DMPs) based on variable-length RGB-D input videos; 2) early prediction based on partial input videos of motion trajectories (DMPs) while ensuring smooth (up to the second order) transition from one predicted DMP to another as the DMP parameters are refined once a larger portion of the RGB-D video has been processed; 3) a real dataset of video-trajectory pairs supplemented with a large number of highly randomized synthetic samples; and 4) an experimental evaluation showing that the proposed approach exhibits good prediction accuracy of robot trajectories and achieves higher accuracy when making use of large datasets that include both synthetic and real data compared to when smaller sets consisting only of real or simulated data are used.

A. Related Work

For the representation of handover trajectories, DMP [15] representation has proven as an attractive choice, since it can provide both a mechanism to plan a feed-forward trajectory as well as the ability to modulate the desired trajectory in a feedback loop. DMPs have shown to be useful as a representation of likely giver trajectories to estimate the handover location and timing based on human hand position measurements and an extended Kalman filter [16]. To encode several handover trajectories in a unified representation, Amor *et al.* [17] developed interaction primitives, which build on DMPs by maintaining a distribution over the DMP parameters. Bahl *et al.* [18] proposed neural dynamic policies (NDPs) that make predictions in trajectory distribution space for reinforcement learning. Probabilistic movement primitives (ProMPs) similarly maintain a distribution of trajectories and can be applied to predict the observed motion trajectories [19]. As an alternative to compact representations such as DMPs and ProMPs, Yamane *et al.* [20], [21] explored the application of motion graphs to enable motion generation from a large database of example trajectories and demonstrated that augmented motion graphs can be used to generate object handover behaviors. However, the above works do not address the issue of discontinuities in receiver motion that arise once the neural network changes the prediction of the giver motion, e.g., based on information obtained as longer videos of the giver motion become available, which is an important feature of our approach based on third-order DMPs.

In recent years, deep neural networks have become a method of choice when processing raw input images for object detection [22]. In the context of interaction, Park and Kim [23] utilized a novel convolutional neural network (CNN) architecture for human identification from images for an improved human–robot collaboration. CNNs were also employed to generate robot trajectories in the form of DMPs from input images [24], [25] and to detect the pick up location of the shaft for a robot-to-robot handover [26], however, without prediction capability. Pahič *et al.* [27], [28] derived differential equations for an effective gradient calculation during backpropagation when using DMP representation. However, these approaches are not suitable for prediction from variable-length (partial) videos as they arise in object handover tasks.

RNNs are more suitable for the processing of a stream of incoming data in dynamic settings such as object handover. LSTM-based RNNs [6] have proven to be especially useful in such settings. They have been applied for action recognition from optical flow [29], for choosing optimal actions based on future predictions of input images [30] and in route planning of ground-based mobile robots for seed delivery [31]. In the context of human motion prediction with a potential use in object handover settings, RNNs have been successful in anticipating and classifying human actions [14], [32], [33] and whole body motions [34], while several methods for forecasting body poses were developed [11]–[13]. The above-mentioned methods, however, either rely on motion capture systems for extraction of the observed person's skeleton, or can only predict labels and final poses rather than smooth motion trajectories from RBG-D videos as our proposed approach.

Simulation-to-Real Transfer: has been used to reduce the amount of real-world data needed for neural network training. One of the methods for optimizing transfer of knowledge gained through simulation is *domain randomization*, where several parameters are randomized for each training sample. It aims to train a neural network for as many environments as possible, including the real world. Domain randomization has been successfully used in robotics to train hierarchical manipulation policies [35], to determine object grasping locations from red-green-blue (RGB) images [9], to perform grasp planning on unseen objects by training on randomized shapes in simulation [36], and for object detection [37]. Some methods attempt to improve the performance by focusing on the most troublesome environmental parameters [38]. Others use generative adversarial networks for domain adaptation [39], to translate randomized simulation images into canonical versions and then use the trained structure on real-world images [10], or to map a simulation image to a realistic one while using a policy trained in simulation [40]. An alternative approach for efficient database gathering has been presented in [41], where a small number of task executions is performed with a robot and statistical generalization is applied to generate more data. We incorporated synthetic data and domain randomization to further improve motion prediction in the proposed system.

As explained above and different from our approach, the current methods for motion observation and prediction during object handover either provide the final handover pose or classify the observed motion from RGB-D images or require expensive motion capture systems to obtain appropriate input data, such as pose measurements. Our method, on the other hand, predicts the entire observed trajectories directly from (partial) RGB-D videos and therefore provides richer information about the handover process without requiring the intermediate step of extracting the giver postures.

II. Generation of Object Handover Trajectories

Given the object handover scenario, the aim of motion representations described in this section is to enable the generation of smooth receiver motion from the partially observed motion of the giver, where the motion of the giver is observed by an RGB-D camera. Thus the input data consist of a sequence of RGB-D images, which are used either to predict the motion of the giver or to directly compute the corresponding motion of the receiver. The idea is that an RNN can start predicting the handover motion based on the incomplete motion of the giver, i.e., based on partial RGB-D videos, and that this prediction is improved once more complete videos are processed. We selected third-order DMPs to encode the predicted motion. The motivation for this is that with third-order DMPs, we can ensure smooth receiver motion transitions (up to the second derivatives) when the RNN computes the next predicted DMP, which provides a better estimate for the desired motion. This cannot be guaranteed by many other motion representations [15], [17], [19] that were utilized to implement interaction tasks in the past.

A. Third-Order DMPs

DMPs are especially well-suited to represent handover trajectories because they can be used to smoothly pull the robot toward the desired motion even if its current position is not on the desired trajectory. Let us assume that the trained network maps the incoming stream of RGB-D images to the receiver motion represented by a DMP. The receiver agent would typically start its movement before the giver finishes its motion. This is possible in our approach because RNNs compute the best estimate for the receiver motion after every processed frame. This estimate is improved as more data become available (see Fig. 1). Third-order DMPs provide a mechanism to smoothly switch from one movement to another.

Let us denote the robot control parameters (internal joint angles or Cartesian space pose) by $\mathbf{y} \in \mathbb{R}^d$, where d is the number of robot degrees of freedom. In the DMP formalism, the control variable $\mathbf{y}$ and its derivatives are computed by integrating a nonlinear dynamic system. To ensure continuity up to the second order derivatives when switching from one DMP to another and motivated by the DMP systems developed in [42] and [43], we propose the following third-order dynamic system to specify the required handover motion

$$\tau\dot{\mathbf{v}} = \mathbf{K}(\mathbf{r}-\mathbf{y}) - \mathbf{D}\mathbf{v} - x\mathbf{K}(\mathbf{r}-\mathbf{y}_0) + \mathbf{K}\mathbf{f}(x) \tag{1}$$

$$\tau\dot{\mathbf{y}} = \mathbf{v} \tag{2}$$

$$\tau\dot{\mathbf{r}} = \mathbf{H}(\mathbf{g}-\mathbf{r}) \tag{3}$$

where $\mathbf{r}$, $\mathbf{v} \in \mathbb{R}^d$ are auxiliary variables, $\mathbf{y}_0$, $\mathbf{g} \in \mathbb{R}^d$ are the start and endpoint of the movement, respectively, $\mathbf{K}$, $\mathbf{H} \in \mathbb{R}^{d\times d}$ are spring matrices, $\mathbf{D} \in \mathbb{R}^{d\times d}$ is a damping matrix, and $\tau > 0$ is a temporal scaling factor, which is usually set to the duration of motion. In our experiments we set $\mathbf{K} = K\mathbf{I}$, $\mathbf{D} = D\mathbf{I}$, $\mathbf{H} = H\mathbf{I}$, $D = 2\sqrt{K}$, $H = \sqrt{K}$, $K > 0$, which provides for the critical damping of the dynamic system. The dynamic system (1)–(3) is driven by the phase variable x, which evolves according to the following equation:

$$\tau\dot{x} = -\alpha_x x \tag{4}$$

where $\alpha_x > 0$ is a positive constant.

The nonlinear forcing term $\mathbf{f}$ from (1) is defined as a combination of radial basis functions (RBFs)

$$\mathbf{f}(x) = \frac{\sum_{k=1}^{N} \boldsymbol{w}_k \Psi_k(x)}{\sum_{k=1}^{N} \Psi_k(x)} x \tag{5}$$

$$\Psi_k(x) = \exp(-h_k(x-c_k)^2). \tag{6}$$

The parameters c_k, $h_k \in \mathbb{R}$ and $\boldsymbol{w}_k \in \mathbb{R}^d$ define the centers, widths, and weights of individual RBFs, respectively, and N is the number of RBFs. The weights $\boldsymbol{w}_k$ are determined in such a way that by integrating equation system (1)–(4) we obtain the desired trajectory $\mathbf{y}$ starting at the initial configuration $\mathbf{y}_0$ and ending at the goal configuration $\mathbf{g}$. The integration process is initialized by setting $\mathbf{y} = \mathbf{y}_0$, $\mathbf{v} = 0$, $\mathbf{r} = \mathbf{g}$, and $x = 1$. Note that $\mathbf{r}$ remains constant unless $\mathbf{g}$ changes. If the goal position $\mathbf{g}$ changes abruptly, $\mathbf{r}$ and consequently $\mathbf{y}$ converge to the new goal position without causing any discontinuities in the acceleration of $\mathbf{y}$. This is important because $\mathbf{y}$ is used to specify the desired robot trajectory.

B. Smooth Switching Between Receiver DMPs

The RNN described in Section III computes a complete new DMP after every processed video frame, not just the final end position $\mathbf{g}$. Thus, to ensure that the integrated motion remains smooth up to the second order when switching to the next DMP, we have to make sure that the initial integration parameters are all set correctly for the next DMP integration. Third-order DMPs have a sufficient number of free parameters to guarantee such smoothness. Let us denote the current DMP integration state as $\mathbf{y}_p, \mathbf{v}_p, \mathbf{r}_p$ and the terms defined by the previous and next DMP (temporal scaling factor, forcing term, end, and initial configuration) as $\tau_p, \mathbf{f}_p, \mathbf{g}_p, \mathbf{y}_{0,p}$ and $\tau_s, \mathbf{f}_s, \mathbf{g}_s, \mathbf{y}_{0,s}$, respectively. We should initialize the next DMP integration state $\mathbf{y}_s, \mathbf{v}_s$, and $\mathbf{r}_s$ so that the position, velocity, and acceleration of the robot motion remain smooth, i.e., $\mathbf{y}_p = \mathbf{y}_s$, $\dot{\mathbf{y}}_\mathrm{p} = \dot{\mathbf{y}}_\mathrm{s}$, $\ddot{\mathbf{y}}_\mathrm{p} = \ddot{\mathbf{y}}_\mathrm{s}$. Using (1) and (2), we compute the following initialization values for the integration of the next DMP, starting at the current phase x:

$$\mathbf{y}_s = \mathbf{y}_p \tag{7}$$

$$\mathbf{v}_s = \frac{\tau_s}{\tau_p}\mathbf{v}_p \tag{8}$$

$$\begin{aligned}\mathbf{r}_s = {} & \frac{\tau_s^2}{\tau_p^2}\mathbf{r}_p + \frac{1-\tau_s^2/\tau_p^2}{1-x}\mathbf{y}_p + \frac{\tau_p\tau_s - \tau_s^2}{\tau_p^2(1-x)}\mathbf{K}^{-1}\mathbf{D}\mathbf{v}_p \\ & + \frac{1}{1-x}\left(\frac{\tau_s^2}{\tau_p^2}(x\mathbf{y}_{0,p} + \mathbf{f}_p(x)) - x\mathbf{y}_{0,s} - \mathbf{f}_s(x)\right).\end{aligned} \tag{9}$$

By initializing the integration of the newly computed DMP with (7)–(9) and by continuing the integration process at the current phase x, we ensure that the generated robot motion is twice continuously differentiable.

The computed initialization values $\mathbf{y}_s, \mathbf{v}_s$, and $\mathbf{r}_s$ do not necessarily lie on the desired trajectory defined by the newly computed DMP. However, since a DMP defines a control policy, the integration of the DMP can also be started away from the desired trajectory. The DMP integration is nevertheless guaranteed to converge to the desired final configuration where the object exchange should take place.

C. Smooth Switching Using Quintic Polynomials

If the neural network predicts the giver trajectory, we can generate the receiver motion by utilizing the predicted final configuration $\mathbf{g}$ of the DMP. If the giver motion is predicted in joint space, we first convert the predicted goal configuration into Cartesian space coordinates using forward kinematics. To enable collision-free object handover, we rotate the goal orientation by 180° so that the robot reaches the desired final pose from the opposite side of the object. The resulting pose is then converted into joint coordinates of the receiver robot using its inverse kinematics, thus providing the desired final configuration of the receiver robot. Finally, we generate a smooth motion for the receiver robot from its current configuration to the computed goal configuration using quintic polynomials [44].

Let us write down the quintic polynomial describing the motion of the receiver robot from the current robot configuration $\mathbf{y}_0$ to the desired goal configuration $\mathbf{g}$

$$\mathbf{y}(t) = \sum_{j=0}^{5}\mathbf{q}_j t^j \tag{10}$$

where $\mathbf{q}_j \in \mathbb{R}^d$ are the coefficients of the polynomial, $t \in [0, T]$, and T represents the time at which the goal position $\mathbf{g}$ should be reached with zero velocity and acceleration from the current position $\mathbf{y}_0$. To prevent excessively fast motions, the duration T is calculated as follows:

$$T = 0.5\max_i\left\{\frac{|g_i - y_{0,i}|}{v_{i,\max}}\right\} \tag{11}$$

where $v_{i,\max}$ is the maximum velocity for the ith robot degree of freedom.

To ensure smooth robot motion, the position, velocity, and acceleration must be continuous when transitioning from one predicted goal configuration to the next one. Denoting the current robot position, velocity and acceleration with $\mathbf{y}_0$, $\dot{\mathbf{y}}_0$, and $\ddot{\mathbf{y}}_0$, respectively, and resetting time to zero, we obtain

$$\mathbf{q}_0 = \mathbf{y}_0,\ \ \mathbf{q}_1 = \dot{\mathbf{y}}_0,\ \ \mathbf{q}_2 = \ddot{\mathbf{y}}_0/2. \tag{12}$$

The following formulas provide the rest of the coefficients:

$$\mathbf{q}_3 = \frac{20(\mathbf{g}-\mathbf{y}_0) - 12\dot{\mathbf{y}}_0T - 3\ddot{\mathbf{y}}_0T^2}{2T^3} \tag{13}$$

$$\mathbf{q}_4 = \frac{30(\mathbf{y}_0-\mathbf{g}) + 16\dot{\mathbf{y}}_0T + 3\ddot{\mathbf{y}}_0T^2}{2T^4} \tag{14}$$

$$\mathbf{q}_5 = \frac{12(\mathbf{g}-\mathbf{y}_0) - 6\dot{\mathbf{y}}_0T - \ddot{\mathbf{y}}_0T^2}{2T^5}. \tag{15}$$

Every time a new goal configuration is predicted by the neural network, a new receiver trajectory is computed using the above procedure.

III. LSTM-Based Neural Network for Object Handover

Due to their ability to process temporal sequences of data, RNNs are well-suited for the task of motion prediction from a sequence of input images. Their structure, consisting of memory units, allows them to store information, dependent on inputs from previous states. However, during the training of classic RNNs, the gradients of the optimized loss function can explode or vanish [45]. This was the motivation behind the development of a special kind of RNN–the LSTM network [6].

Unlike classic RNNs, LSTM units are composed of a cell and several gates, regulating the flow of information in and out of the cell. Fig. 2 shows the structure of the LSTM unit, where each new cell state c_t and new hidden state h_t are fed back to the LSTM unit when processing the next input. Thus, each new network output is affected by the results from the previous time steps.

The trainable parameters in an LSTM unit are the weights for the inputs x_t, the weights for the combined hidden state h_t, and the biases for each of the four gates. For an LSTM layer with m units and an input size of n, the number of trainable parameters is therefore equal to $4m(n + m + 1)$.

A. Network Architecture

Our proposed architecture is called RIMEDNet and consists of convolutional, fully connected, and LSTM layers. It takes a sequence of images of the observed motion as input and predicts either the motion trajectory of the giver performing

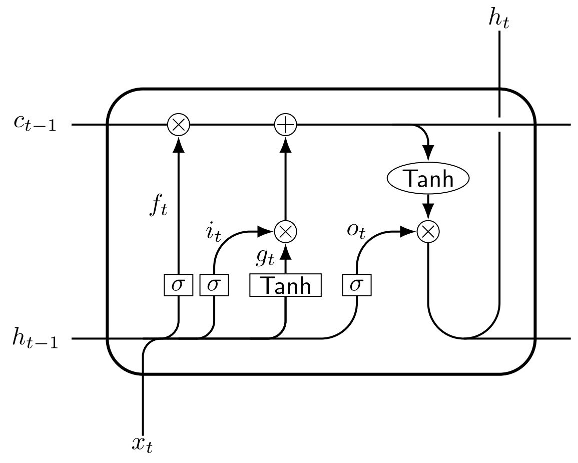

Fig. 2. Structure of an LSTM unit. The inputs to the unit are the hidden state h_{t-1} and the cell state c_{t-1} from the previous time step and the current input data x_t. The new cell state c_t and the new hidden state h_t are defined by the inputs and the trainable parameters of the LSTM unit, with i_t, f_t, g_t, and o_t representing the outputs of different gates, i.e., the input gate, forget gate, cell gate, and output gate.

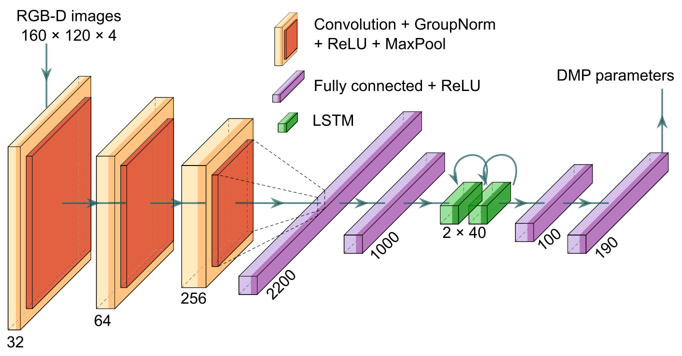

Fig. 3. Example recurrent RIMEDNet architecture with the custom convolutional part, fully connected, and stacked LSTM layers. RIMEDNet takes RGB-D images as input to predict a motion trajectory in the form of DMP parameters. Convolution layers have a kernel size of 4 × 4 and a stride of 1, while all max pooling layers have a kernel size of 3×3 and a stride of 3, except the first one, which has a stride of 2. In this example, the input image size was 160 × 120 × 4, the robot had 7 degrees of freedom, there were 25 RBFs in the forcing term, seven parameters for the initial and end configuration, and the temporal scaling factor, thus altogether 190 output parameters. The total number of trainable parameters in this network is 22 416 346.

the handover task or the corresponding motion of the receiver, where the resulting trajectory is represented by a DMP. The basic network structure is shown in Fig. 3, where the input data are in the form of RGB-D image sequences and the output data are the DMP parameters.

Input images are passed into the convolutional part of the network, which can either be custom-defined CNN layers (see Fig. 3) or the layers from one of the popular pretrained CNN architectures, commonly used for image classification (e.g., ResNet [46], AlexNet [3], GoogLeNet [47], etc.). In all cases, the convolutional part is followed by two fully connected layers with ReLU functions, two LSTM layers, and another two fully connected layers that finally produce the output of the network. At the heart of the network architecture are two stacked LSTM layers. For each element in the input sequence, i.e., an RGB-D image, the LSTM network returns an output based on all the previous input images in the sequence and the previously computed states of the LSTM units.

The number of input features of the two LSTM layers has been set to 1000 and 40, respectively, and the number of units in each LSTM layer is 40, thus resulting in 40 outputs for each layer and 179 520 trainable parameters. The proposed encoder–decoder structure forces the network to create low-dimensional representations of input images and thus extract the essential features of the data, which may improve the network's ability to generalize to previously unseen data.

The network takes sequences of $W \times H \times 4$ pixel RGB-D images as input, where W and H are the width and height of input images. Let us denote by d the number of robot degrees of freedom (Cartesian or internal). The network outputs a parameter vector consisting of DMP parameters that include d values for start and goal configuration $\mathbf{y}_0$ and $\mathbf{g}$, respectively, $d \times N$ values describing the weights (5) of the forcing term, and one value for the temporal scaling factor τ, thus altogether $(N + 2)d + 1$ parameters. The robot degrees of freedom are given either as Cartesian space positions $\mathbf{p}(t) \in \mathbb{R}^3$ and orientations $\mathbf{q}(t) \in \mathbb{R}^4$, $\mathbf{y}(t) = [\mathbf{p}(t)^{\mathrm{T}}, \mathbf{q}(t)^{\mathrm{T}}]^{\mathrm{T}}$, or as robot joint angles $\mathbf{y}(t) = [\theta_1(t), \ldots, \theta_d(t)]^{\mathrm{T}}$.

B. Training Method

The proposed RIMEDNet architecture is trained on data pairs, defined as

$$\mathbf{D} = \left\{\{\mathbf{X}_{ij}\}_{i=1}^{L_j},\ \mathbf{d}_j\right\}_{j=1}^{M} \tag{16}$$

where M is the number of example sequences in the training dataset, $\mathbf{X}_{ij} \in \mathbb{R}^{W\times H\times 4}$ is the ith frame of the jth input video consisting of L_j frames, and $\mathbf{d}_j \in \mathbb{R}^{(N+2)d+1}$ is the corresponding ground-truth trajectory (either the trajectory of the observed giver motion or the corresponding trajectory of the receiver), encoded with DMP parameters

$$\mathbf{d}_j = \left\{\{\mathbf{w}_{k,j}\}_{k=1}^{N},\ \mathbf{g}_j,\ \mathbf{y}_{0,j}, \tau_j\right\} \tag{17}$$

where N is the number of weights in the forcing term (5). See Section IV for details about how these data are gathered.

To train the proposed network, we need to define a suitable loss function. We use the standard mean squared error (mse) as the basis for the definition of the loss function. However, since the input data are image sequences instead of single images, we employ a weighted variant of MSE. The loss of the complete jth training sample is computed as

$$E(j) = \frac{1}{L_j}\sum_{i=1}^{L_j}\gamma_i\Bigg(\sum_{k=1}^{N}||\mathbf{w}_{k,j} - \mathbf{w}_{k,i}^{o}||^2 + \alpha_g\ ||\mathbf{g}_j - \mathbf{g}_i^{o}||^2 + \alpha_y||\mathbf{y}_{0,j} - \mathbf{y}_{0,i}^{o}||^2 + \alpha_\tau||\tau_j - \tau_i^{o}||^2\Bigg). \tag{18}$$

The parameters $\mathbf{w}_{k,j}$, $\mathbf{g}_j$, $\mathbf{y}_{0,j}$, and τ_j are the ground-truth DMP parameters from the training set (17), while $\mathbf{w}_{k,i}^{o}$, $\mathbf{g}_i^{o}$, $\mathbf{y}_{0,i}^{o}$ and τ_i^{o} are the output DMP parameters computed by the neural network for the ith input image. Additional weights

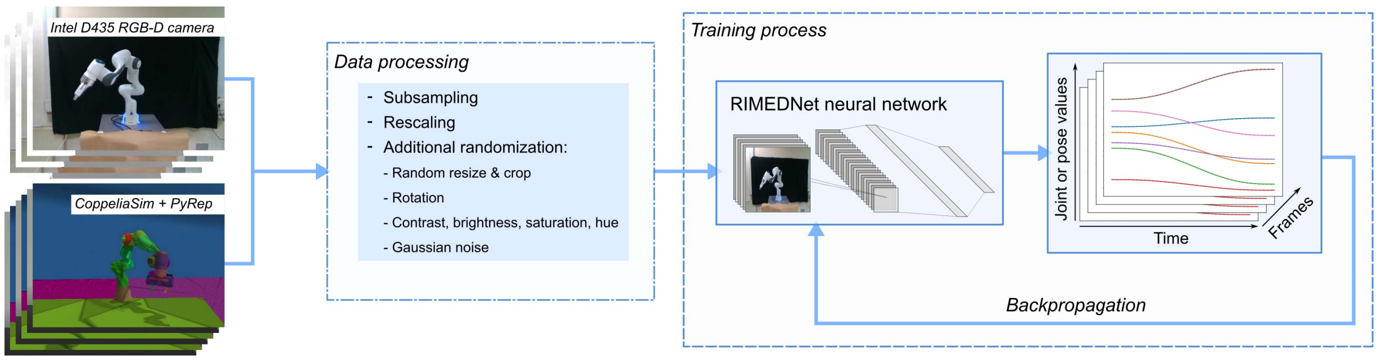

Fig. 4. Training process flow. Real and synthetic videos are transformed using subsampling, rescaling, and additional randomization techniques. The processed data are then used to train the RIMEDNet architecture using backpropagation and a temporally weighted mse loss.

$\alpha_g, \alpha_y, \alpha_\tau > 0$ are used to balance the importance and scaling of parameters with different units. A logistic function with scaling factor α_t is used to compute the weights γ_i

$$\gamma_i = \frac{1}{1 + e^{-\alpha_t \frac{i-1}{L_j - 1} + 0.5}}. \tag{19}$$

This way we decrease the significance of early video frames and increase the significance of later frames while ensuring that the values of weights γ_i are in the range from 0 to 1.

The proposed neural network model and its training were implemented using PyTorch [48] and an NVIDIA GeForce GTX 1080 graphics processing unit. The training process was carried out using the Adam optimizer [49] and a batch size of 30. To prevent overfitting of the network to training data, the error on validation dataset was computed after each epoch and the training was stopped after 60 consecutive epochs of no validation error decrease. The learning rate was changed in a cyclical manner using the approach from [50], which increases training speed and reduces the need for parameter tuning.

To enable batch training, which significantly reduces the training time, the input videos were padded using the final frame in each video so that all the videos in the batch were of the same length. Besides speeding up the training process, by padding the videos we also increase the robustness of motion prediction when the images of the robot standing still are fed into the network after the end of the motion. This enables more accurate trajectory prediction even after the robot motion has been finished.

IV. Experiments

The goal of our experiments was to determine the accuracy of the RIMEDNet architecture when predicting the giver trajectories or the corresponding receiver trajectories during robot-to-robot handover tasks and whether the accuracy can be improved by utilizing simulation data. Table I shows different experimental scenarios, where the combinations of giver, receiver, Cartesian, and joint trajectories were used for training, and either only real or mixed (real and synthetic) data were used. To compare different scenarios we used the custom CNN structure, depicted in Fig. 3, due to its simple structure and faster training. Additional architectures are discussed in Section IV-D. The process flow is depicted in Fig. 4.

TABLE I
RIMEDNet Training Scenarios

Name	Robot	Trajectory type	Data source
R-GC	Giver	Cartesian	Real
R-GJ	Giver	Joint	Real
R-RC	Receiver	Cartesian	Real
R-RJ	Receiver	Joint	Real
SR-GC	Giver	Cartesian	Sim + Real
SR-GJ	Giver	Joint	Sim + Real
SR-RC	Receiver	Cartesian	Sim + Real
SR-RJ	Receiver	Joint	Sim + Real

A. Data Collection

1) Synthetic Data: The generation of synthetic data (31 195 samples) was implemented using the robot simulator CoppeliaSim [51] in combination with the PyRep toolkit [52] for robot learning research. To simulate a robot-to-robot handover task, a giver robot performed minimum jerk motions [44] with randomly selected initial position, fixed initial orientation, and randomly selected end pose, while being recorded with a simulated camera. The initial and final positions as well as final orientations were confined to a fixed range of values. During each simulation episode, a stream of RGB-D image data were recorded with a simulated camera, and joint and Cartesian trajectories were generated both for giver and receiver robots, the duration ranging from 1.5 to 5 s. The corresponding motion of the receiver robot was generated as a minimum jerk trajectory, starting from the fixed initial pose to the end position of the giver, while the orientation was rotated by 180° to enable collision-free object handover. Note that the receiver motion is only generated for data collection and does not need to be executed.

A customized fork of PyRep[1] was developed featuring domain randomization functionality that was employed in the simulation environment in an attempt to minimize the discrepancy between the source domain (simulation) and the target domain (real world). The purpose of randomization is to capture the variability of real images in simulated images. To achieve appropriate variability, the colors and texture

[1]https://github.com/abr-ijs/PyRep/tree/feature/domain_randomization

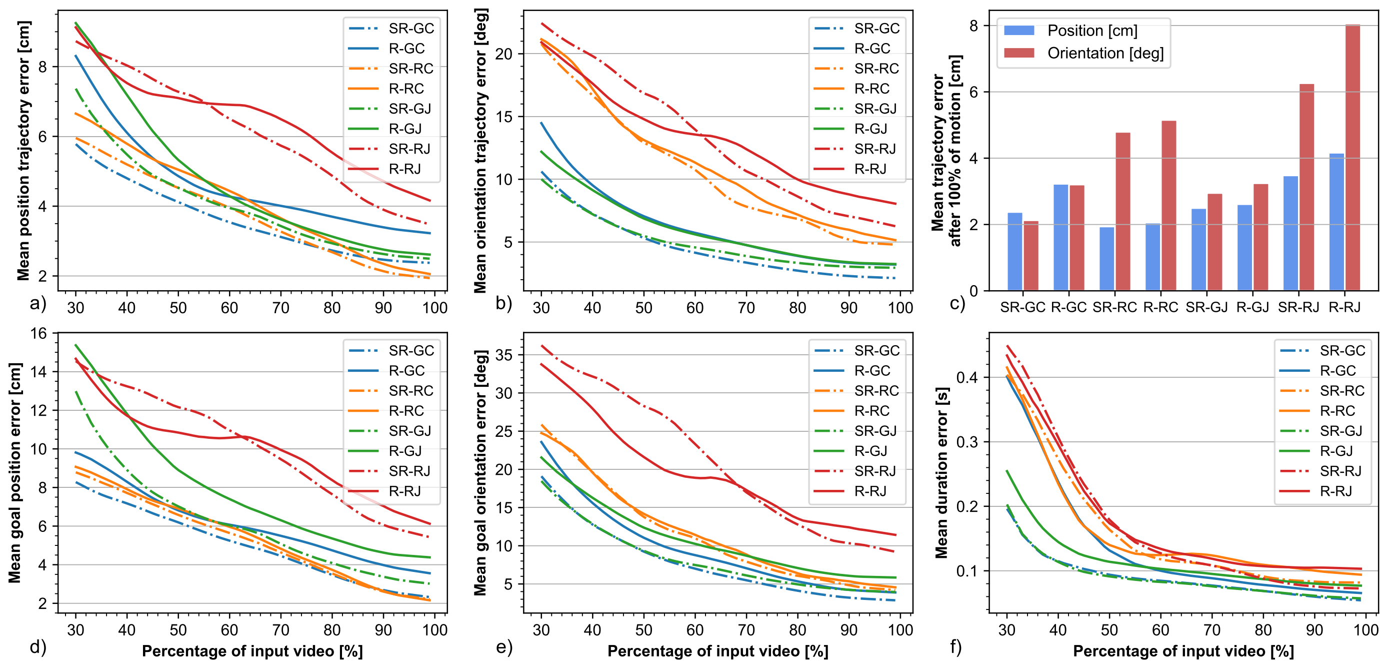

Fig. 5. Comparison of test errors for RIMEDNet models with the custom CNN structure, trained on different trajectory and data types, where an increase in accuracy can be observed with more processed frames for all models. Additionally, in nearly all cases accuracy is better when mixed data are used instead of only real data. Errors in (a) and (b) are defined as mses between the actual and predicted robot trajectories, while only goal errors are shown in (d) and (e). In (c), the mean trajectory errors after processing 100% of input video are shown, mean duration errors in regard to percentage of processed input videos are displayed in (f).

patterns of all surfaces (including those of the robot) were randomized, along with the locations of the table and the back wall as well as camera and light source poses. In addition, brightness, contrast, and saturation were randomly adjusted and Gaussian noise was added to the acquired frames.

2) Real Data: The real environment closely resembled the simulated setting. The acquisition of real data was carried out by performing a portion (7 198) of the generated simulation trajectories with a real giver robot and recording its movements with an Intel RealSense D435 depth camera. The Robot Operating System (ROS) was used to synchronize robot commands and the camera recording program by sending the appropriate start and stop signals through a ROS topic. We applied small random rotations to the captured RGB-D images to account for possible small changes of the camera view directions. Just like in simulation, brightness, contrast, and saturation were randomly adjusted and Gaussian noise was added to the acquired frames. However, the standard deviation of changes and noise was smaller in this case.

B. Datasets

All RGB-D frames (simulated and real) were resized to 160×120 pixels and the video streams were sampled at 5 Hz. Values of input videos and output DMP parameters were normalized to be in the range from 0 to 1. The datasets used in the experiments were as follows.

1) *SimVTTrain*: This set consists of 31 195 simulated video-trajectory pairs and was used for network training.
2) *RealVTTrain*: This set consists of 5879 video-trajectory pairs, obtained with a real robot, and was also used for training.
3) *RealVTVal*: For validation of the training performance, we used 654 video-trajectory pairs, obtained with a real robot and with the same processing as the RealVTTrain.
4) *RealVTTest*: For evaluation of the final neural network models, we used 665 video-trajectory pairs, obtained with a real robot and with no additional randomization.

All videos were sampled at 5 Hz. The frame rate of 5 Hz is sufficient for accurate motion prediction because we only need to recover specific robot receiver or giver motions for which the networks were trained, not any arbitrary motion.

C. Performance of RIMEDNet in Different Scenarios

The RIMEDNet architecture with the custom CNN structure was trained using different scenarios from Table I. The datasets described in Section IV-B were employed for this purpose. The resulting training, validation, and test errors are shown in Table II, where the mse between ground-truth and predicted normalized DMP parameters from (17) are depicted. The training and test errors for each network are relatively similar, with the training errors being lower by a factor of 1.31–2.42. This indicates negligible overfitting of the networks to training data, which is due to the fact that the analysis of the validation dataset results stopped the training process before significant overfitting could occur.

The trained networks were evaluated using the RealVTTest dataset, which was excluded from the training and validation process. The test videos were passed through the models in order to obtain trajectory predictions for each frame, as shown in Fig. 1. The prediction accuracy of the obtained trajectories was analyzed in Cartesian space, both in terms of the spatial course of motion and the duration of the trajectories.

TABLE II
RIMEDNET TRAINING/VALIDATION/TEST DMP ERRORS

	Training error	Validation error	Test error
R-GC	0.293	0.402	0.384
R-GJ	0.175	0.418	0.423
R-RC	0.780	1.384	1.19
R-RJ	0.765	1.233	1.028
SR-GC	0.144	0.274	0.269
SR-GJ	0.154	0.331	0.335
SR-RC	0.829	1.378	1.121
SR-RJ	0.645	1.230	0.997

Note: The values in the table should be multiplied by 10^{-2} to obtain the correct errors.

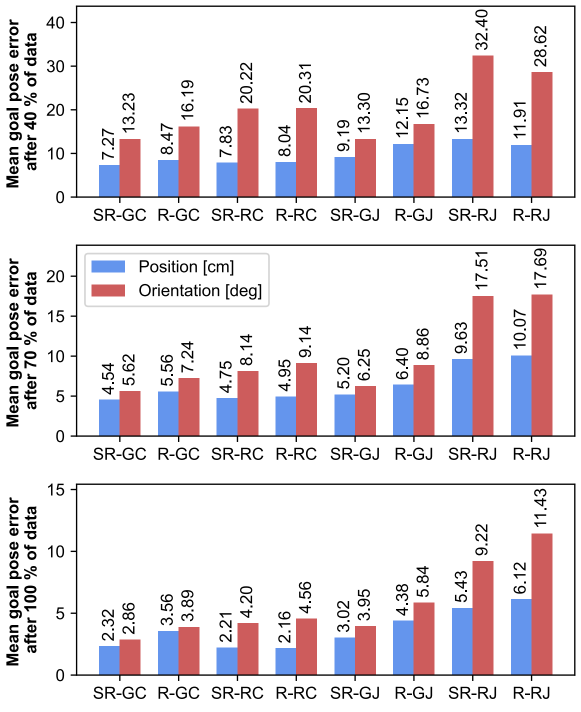

Fig. 6. Mean goal pose errors after processing 40%/70%/100% of the motion video. The position and orientation errors are decreasing as more of the motion video is processed, reaching as low as 2.16 cm and 2.86°.

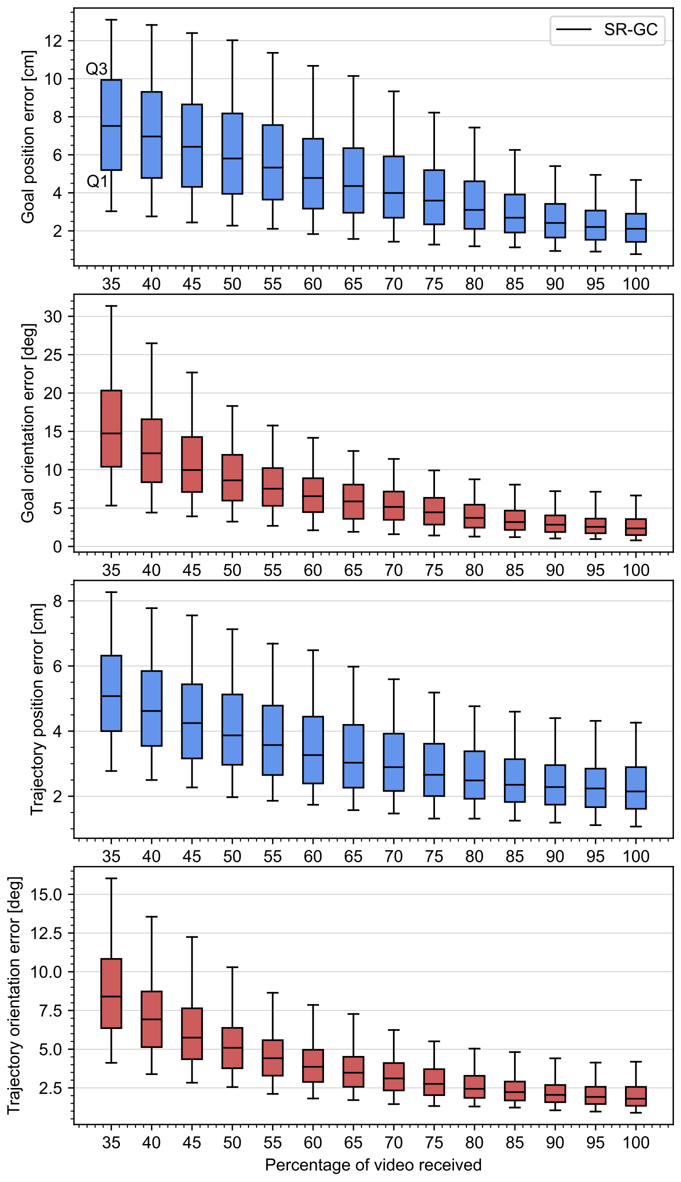

Fig. 7. Box-and-whiskers plot of position and orientation errors for the SR-GC model. The black middle lines are the medians. The boxes show the range of data between the first quartile $Q1$ (25%) and the third quartile $Q3$ (75%). The whiskers extend from the fifth to the 95th percentile.

The comparison of trajectory test errors for RIMEDNet architectures is shown in Fig. 5. The position, orientation, and duration prediction errors for each model are represented in relation to the percentage of the motion video that has been processed by the network, starting at 30%, where the errors are shown for the entire trajectories as well as for only the goal poses. The mean trajectory errors after processing the entire input video are also compared, while the average goal pose errors after processing 40%, 70%, and 100% of motion video are shown in Fig. 6.

The results show gradual decrease of the prediction errors when more frames of the recorded motion are processed. The models trained on both synthetic and real data exhibit higher prediction accuracy than when trained only on real data. Models trained in simulation and real - giver Cartesian (SR-GC), simulation and real - giver joint (SR-GJ), simulation and real - receiver Cartesian (SR-RC), and real - receiver Cartesian (R-RC) scenarios all performed comparably well. While the models computed in the SR-RC and R-RC scenarios achieved especially low mean position trajectory errors toward the end of motion, their goal pose errors were similar or higher. The low average trajectory errors arising in the SR-RC and R-RC scenarios are probably due to the fact that the receiver motions all begin from the same starting position and are thus less variable, whereas the giver trajectories can start from different initial positions. The higher errors of the predicted joint trajectories, on the other hand, probably arise from the conversion of joint space trajectories into Cartesian space, which may increase the scale of error due to the nonlinear mapping. The trajectory duration error for all models is below 0.2 s after approximately 50% of motion.

A detailed box-and-whiskers graph of goal and trajectory errors for SR-GC scenario is displayed in Fig. 7, showing that the median errors decrease with the number of frames. The

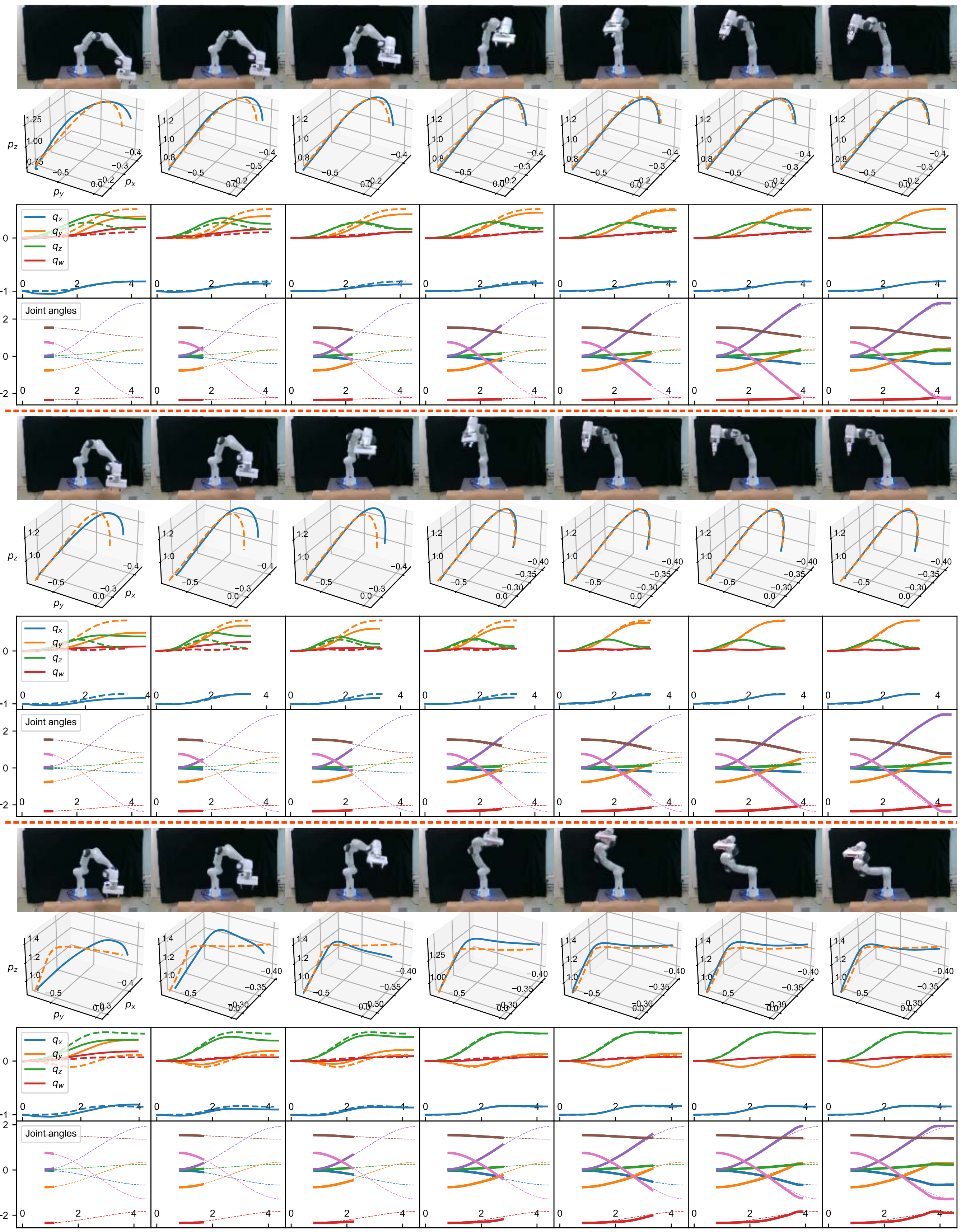

Fig. 8. Examples of trajectory predictions for the motion of the giver. Each group of four rows belongs to the same motion example, where rows 2 and 3 display position predictions as paths in 3-D space and orientation predictions as quaternion values through time (obtained using the SR-GC scenario). Row 4 shows the motion trajectory of the receiver robot, generated by switching between the predicted receiver trajectories as described in Section II-C. Here the dashed lines show the ground truth receiver trajectories, while the actual robot receiver joint trajectories are shown with thick lines.

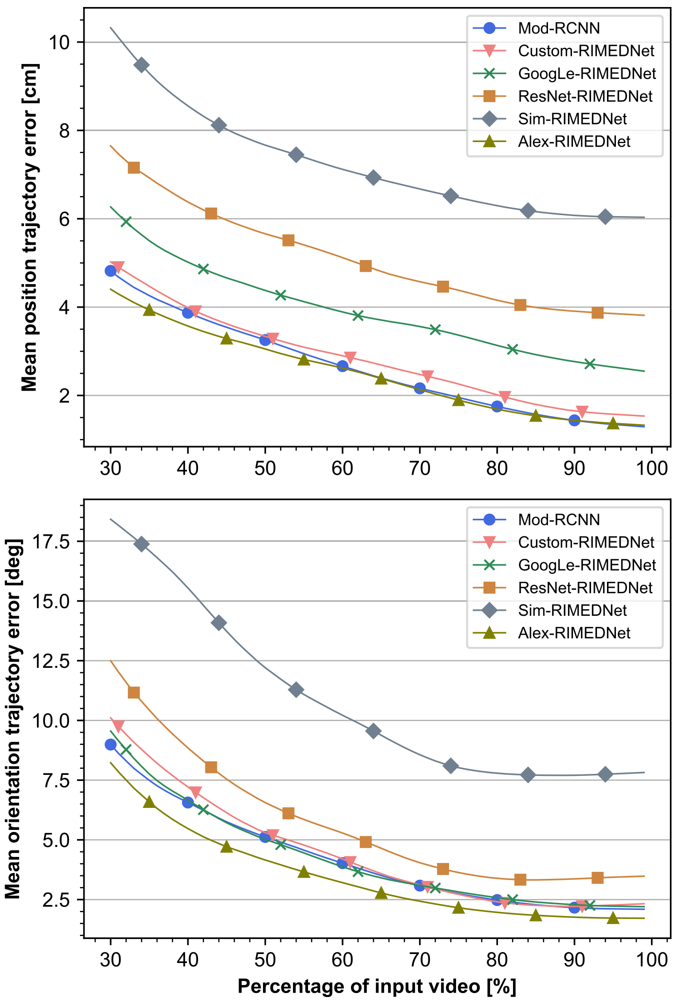

Fig. 9. Mean trajectory position and orientation errors for RIMEDNet variants with custom-designed convolutional layers (Custom-RIMEDNet), convolutional layers from GoogLeNet (GoogLe-RIMEDNet), convolutional layers from ResNet-34 (ResNet-RIMEDNet), and convolutional layers from AlexNet (Alex-RIMEDNet). The modified RNN architecture from [14] (Mod-RCNN) was also evaluated. Finally, the performance of RIMEDNet with custom-designed convolutional layers trained on simulation data only (Sim-RIMEDNet) is shown.

goal errors reach 2.11 cm and 2.35° after processing an entire motion video. Fig. 8 similarly demonstrates the improvement of accuracy as more frames are processed and also shows examples of giver trajectory predictions and the associated robot receiver trajectories generated by the approach described in Section II-C.

The accuracy of the generated robot trajectories, especially in the SR-GC and SR-RC scenarios, is acceptable for object handover in many real environments, particularly when using compliant robots, where the error of a few centimeters does not prevent a successful handover.

D. Comparative Study of Variants of RIMEDNet

To the best of our knowledge, there exists no previous recurrent deep neural network for direct prediction of DMP-encoded trajectories from RGB-D videos. However, the RNN architecture from [14] similarly uses RGB-D videos to classify the observed human actions. We modified this network by adapting the output layer to allow for DMP parameter prediction and enable comparison to our proposed model. In addition to this network and the network with custom-designed CNN layers shown in Fig. 3, we also tested the variants of RIMEDNet where the convolutional layers were provided by pretrained state-of-the-art CNN architectures: ResNet-34 [46], GoogLeNet [47] and AlexNet [3]. The aim was to identify the best convolutional layers for RIMEDNet. In addition, we also evaluated the RIMEDNet with custom CNN layers trained with synthetic data only.

Since the network in [14] has an input size of 227×227 and common state-of-the-art CNN networks usually have an input size of above 224×224, we set the input size of all compared architectures to 227×227. The frames of RGB-D videos were appropriately transformed to match the input size. Layers of our custom CNN structure from Fig. 3 were slightly modified to preserve the roughly same number of parameters, i.e., the kernel sizes of the three convolution layers were changed to 6–4, while the stride of the first max pooling layer was changed to 3. An additional 1894 training video-trajectory sample pairs, randomized in the same way as RealVTTrain dataset, were obtained using a real robot to improve the accuracy for use in a real handover experiment (see Section IV-E). All networks were trained using the SR-GC scenario due to the high accuracy shown in Section IV-C, i.e., using both real and synthetic data and predicting giver Cartesian trajectories. For the network trained with simulated data only, we used the SimVTTrain dataset. All the networks were validated using the RealVTVal dataset and tested on the RealVTTest dataset. The batch size was reduced to 10 due to the larger resolution of input images.

The comparison of mean trajectory position and orientation errors in relation to the percentage of processed motion video is shown in Fig. 9. RIMEDNet with AlexNet convolutional layers has shown the highest pose prediction accuracy, with mean position and orientation errors reaching 1.33 cm and 1.72° at the end of motion, respectively. The RIMEDNet architecture with custom CNN layers performed better than RIMEDNet with GoogLeNet and ResNet-34 convolutional layers, especially when predicting position trajectories. The modified network from [14] also performed well, but not as well as RIMEDNet with convolutional layers from AlexNet. The RIMEDNet with custom convolutional layers trained on synthetic data only performed considerably worse than the same architecture trained on the mixed simulated and real data. This shows that adding real data to simulated data greatly improves the overall accuracy.

RIMEDNet with convolutional layers from AlexNet has thus proven to be the best choice and significantly better than RIMEDNet with convolutional layers taken from ResNet-34 and GoogLeNet. The additional improvement of the accuracy compared to the results from Section IV-C was presumably achieved with the increase of image resolution to 227×227.

E. Robot-to-Robot Object Handover Experiment

The proposed approach was used to implement the handover task with two Franka Emika Panda robots (see Fig. 10), where

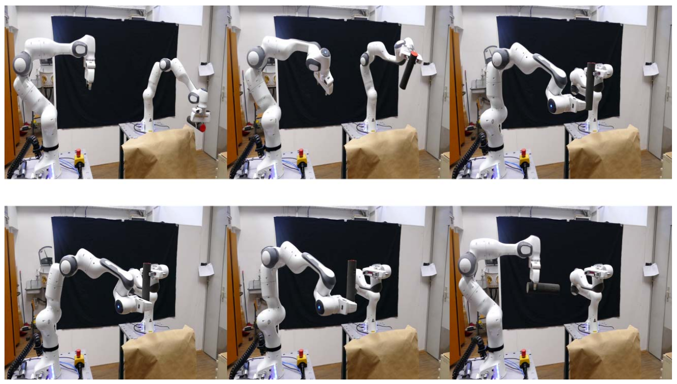

Fig. 10. Application of RIMEDNet to implement the object handover task. The giver robot (right) passes an object to the receiver robot (left). Trajectory predictions are computed online and used to control the receiver robot. With each processed camera frame, a new receiver motion trajectory is generated.

TABLE III
HANDOVER SUCCESS RATE

	Object 1	Object 2
Dimensions ($W \times H \times D$) [cm]	$20 \times 8.3 \times 2.5$	$10 \times 8 \times 3$
Success rate (168 handovers)	74.4%	66.3%

Note: W – width, H – height, D – depth of the object.

the best performing RIMEDNet architecture with convolutional layers from AlexNet was used. The designed handover control system was implemented in ROS. Intel RealSense camera was used to record the motion of the giver robot, with the recording starting at the beginning of motion. Frames were sent in real-time with a rate of 5 Hz to a computer running the RNN model, while the giver robot was following a trajectory from the test dataset at 120 Hz. The duration of test trajectories varied from 1.5 to 5 s, where the average end-effector speed of the giver robot was 0.32 m/s. The system predicted the DMP parameters of the giver robot in Cartesian space, i.e., the SR-GC scenario from Section IV was used. The predicted goal pose of the giver robot was thus used to generate the receiver trajectory toward the object using the approach described in Section II-C. The receiver started its motion 1 s after the giver and its speed was determined by (11).

A subset of 168 randomly selected test motions from the RealVTTest dataset were carried out by the giver robot for two different objects. The pre-handover phase trajectories were designed to approach the predicted goal pose at the distance of 10 cm. In the object exchange phase, the receiver robot moved along a straight line toward the goal pose and attempted to grasp the object.

Table III shows the success rate of the handover process for both objects. Two examples showing one successful and one unsuccessful object handover for Object 1 are shown in Fig. 11. The handover success rate was 74.4% for the narrower Object 1 ($D = 2.5$ cm) and 66.3% for the wider Object 2 ($D = 3$ cm). The main reason for the unsuccessful handovers becomes evident if we analyze the accuracy of the applied RIMEDNet with convolutional layers from AlexNet (1.35 cm with standard deviation of 0.73 cm for position and 2.59°

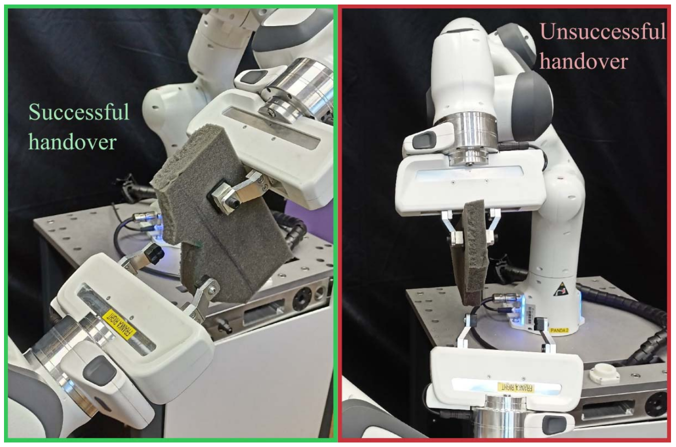

Fig. 11. Example of a successful and an unsuccessful attempt of a robot-to-robot object handover, implemented using RIMEDNet with convolutional layers from AlexNet. The images were taken before the final approach of the receiver robot to grasp the object. The left example shows an accurate prediction of the goal pose, while the grasp attempt on the right image would fail after the approach of the receiver.

with standard deviation of 1.87° for orientation), the distance between the fingers of the fully opened gripper (6.8 cm), and the width of the two objects (2.5 and 3 cm). Thus there is only a small clearance between the object and the gripper fingers, which can be exceeded when the error reaches its mean plus standard deviation. A more reliable object handover could be achieved by utilizing a gripper with wider distance between the fingers or by applying a visual servoing system to fine-tune the final position of the receiver motion. This is similar to human-to-robot object handover, where the human can appropriately adapt the object pose so that the robot can successfully grasp it. In all cases, the proposed system is effective at generating the pre-handover phase motion for the receiver robot.

V. CONCLUSION

In this article, we propose a new approach for motion prediction and generation from RBG-D videos in the context of robot-to-robot object handover tasks. At the core of the approach is the new RNN architecture RIMEDNet, which has been designed for motion prediction from incomplete videos. It consists of stacked convolutional, LSTM, and fully connected layers. RIMEDNet can start making predictions based on the partially observed motion of the giver agent and can process variable-length RGB-D videos. To exploit RIMEDNet's predictions for the generation of robot receiver trajectories, we designed an appropriate motion generation system based on third-order DMPs and quintic polynomials. This system enables smooth switching (up to the second-order derivatives) between the predicted trajectories. Additionally, we have shown that the amount of real-world training examples can be significantly reduced by supplementing the training data with synthetic data, which are usually much easier to gather. The developed system allows for the implementation of an efficient and dynamic object handover procedure, where the receiver robot can start moving toward the object handover location before the giver robot completes its motion.

We plan to extend our approach to a human-to-robot handover task, where the ground-truth data can be obtained using motion trackers. One of the challenges that remain for future research is the generalization power of the proposed network; ideally, the network should generalize to entirely new environments that were not included in the training dataset, especially in the real world. One option is to record the sample videos in front of a green screen and digitally change the background with real images to obtain environments as diverse as possible. This way we can avoid performing too many movements with a real robot while still gathering a large amount of examples based on real robot motion and real images. Another promising method is the domain adaptation approach, where the network can utilize the target domain data without ground truth data, making it easier to transfer knowledge from one domain to another. This approach is especially suitable for the case of human-to-robot handover, where gathering ground-truth data is difficult.

Another possible extension is to include a general-purpose pose tracker network for motion estimation from RGB-D videos. Such a network can provide a separate estimation of the current robot or human pose. The results can be used to improve the accuracy of the predicted trajectories and for segmentation, i.e., to determine the start and end of the giver motion. This way all parts of the proposed system could be fully automated.

REFERENCES

[1] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, Nov. 2018.

[2] V. Ortenzi, A. Cosgun, T. Pardi, W. P. Chan, E. Croft, and D. Kulić, "Object handovers: A review for robotics," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1855–1873, Dec. 2021.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[4] D. Zissis, E. K. Xidias, and D. Lekkas, "A cloud based architecture capable of perceiving and predicting multiple vessel behaviour," *Appl. Soft Comput.*, vol. 35, pp. 652–661, Oct. 2015.

[5] N. Sengupta, M. Sahidullah, and G. Saha, "Lung sound classification using cepstral-based statistical features," *Comput. Biol. Med.*, vol. 75, pp. 118–129, Aug. 2016.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] K. Greff, R. K. Srivastava, J. Koutnìk, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space Odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[8] D. Kalashnikov *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proc. Conf. Robot Learn. (CoRL)*, 2018, pp. 651–673.

[9] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 23–30.

[10] S. James *et al.*, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12627–12637.

[11] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, "Recurrent network models for human dynamics," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4346–4354.

[12] J. Zhang, H. Liu, Q. Chang, L. Wang, and R. X. Gao, "Recurrent neural network for motion trajectory prediction in human–robot collaborative assembly," *CIRP Ann.*, vol. 69, no. 1, pp. 9–12, 2020.

[13] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-RNN: Deep learning on spatio-temporal graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5308–5317.

[14] R. Wang, S. M. Pizer, and J.-M. Frahm, "Recurrent neural network for (un-)supervised learning of monocular video visual odometry and depth," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5555–5564.

[15] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, Feb. 2013.

[16] D. Widmann and Y. Karayiannidis, "Human motion prediction in human–robot handovers based on dynamic movement primitives," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2018, pp. 2781–2787.

[17] H. Ben Amor, G. Neumann, S. Kamthe, O. Kroemer, and J. Peters, "Interaction primitives for human–robot cooperation tasks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 2831–2837.

[18] S. Bahl, M. Mukadam, A. Gupta, and D. Pathak, "Neural dynamic policies for end-to-end sensorimotor learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 5058–5069.

[19] G. J. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, O. Kroemer, and J. Peters, "Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks," *Auto. Robots*, vol. 41, no. 3, pp. 593–612, Mar. 2017.

[20] K. Yamane, Y. Yamaguchi, and Y. Nakamura, "Human motion database with a binary tree and node transition graphs," *Auto. Robots*, vol. 30, no. 1, pp. 87–98, Jan. 2011.

[21] K. Yamane, M. Revfi, and T. Asfour, "Synthesizing object receiving motions of humanoid robots with human motion database," in *Proc. IEEE Int. Conf. Robot. Autom.*, Karlsruhe, Germany, May 2013, pp. 1621–1628.

[22] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019.

[23] J.-Y. Park and J.-H. Kim, "Online incremental classification resonance network and its application to human–robot interaction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 5, pp. 1426–1436, May 2020.

[24] A. Pervez, Y. Mao, and D. Lee, "Learning deep movement primitives using convolutional neural networks," in *Proc. IEEE-RAS 17th Int. Conf. Hum. Robot. (Humanoids)*, Nov. 2017, pp. 191–197.

[25] M. Theofanidis, A. Bozcuoglu, and M. Kyrarini, "Learning visuomotor policies with deep movement primitives," in *Proc. 14th Pervasive Technol. Rel. Assistive Environ. Conf.*, Jun. 2021, pp. 140–146.

[26] M. Sileo, M. Nigro, D. D. Bloisi, and F. Pierri, "Vision based robot-to-robot object handover," in *Proc. 20th Int. Conf. Adv. Robot. (ICAR)*, Dec. 2021, pp. 664–669.

[27] R. Pahič, A. Gams, A. Ude, and J. Morimoto, "Deep encoder-decoder networks for mapping raw images to dynamic movement primitives," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 5863–5868.

[28] R. Pahič, B. Ridge, A. Gams, J. Morimoto, and A. Ude, "Training of deep neural networks for the generation of dynamic movement primitives," *Neural Netw.*, vol. 127, pp. 121–131, Jul. 2020.

[29] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao, "A multi-stream bi-directional recurrent neural network for fine-grained action detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1961–1970.

[30] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 2786–2793.

[31] S. Das, S. M. Welch, and D. Flippo, "Recurrent neural network based multi-robot route planning for steep-land harvesting systems," in *Proc. 8th Int. Congr. Adv. Appl. Informat. (IIAI-AAI)*, Jul. 2019, pp. 542–545.

[32] P. Schydlo, M. Rakovic, L. Jamone, and J. Santos-Victor, "Anticipation in human–robot cooperation: A recurrent neural network approach for multiple action sequences prediction," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 5909–5914.

[33] F. Formica, S. Vaghi, N. Lucci, and A. M. Zanchettin, "Neural networks based human intent prediction for collaborative robotics applications," in *Proc. 20th Int. Conf. Adv. Robot. (ICAR)*, Dec. 2021, pp. 1018–1023.

[34] J. Martinez, M. J. Black, and J. Romero, "On human motion prediction using recurrent neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4674–4683.

[35] O. Nachum, M. Ahn, H. Ponte, S. S. Gu, and V. Kumar, "Multi-agent manipulation via locomotion using hierarchical sim2real," in *Proc. Conf. Robot Learn. (CoRL)*, 2020, pp. 110–121.

[36] J. Tobin *et al.*, "Domain randomization and generative models for robotic grasping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 3482–3489.

[37] J. Tremblay *et al.*, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 969–977.

[38] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active domain randomization," in *Proc. Conf. Robot Learn. (CoRL)*, 2020, pp. 1162–1176.

[39] K. Bousmalis *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 4243–4250.

[40] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari, "RL-CycleGAN: Reinforcement learning aware simulation-to-real," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11157–11166.

[41] Z. Lončarević, R. Pahič, A. Ude, and A. Gams, "Generalization-based acquisition of training data for motor primitive learning by neural networks," *Appl. Sci.*, vol. 11, no. 3, p. 1013, Jan. 2021.

[42] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 2587–2592.

[43] B. Nemec and A. Ude, "Action sequencing using dynamic movement primitives," *Robotica*, vol. 30, no. 5, pp. 837–846, Sep. 2012.

[44] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. New York, NY, USA: Wiley, 2006.

[45] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 6, no. 2, pp. 107–116, 1998.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[47] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[48] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 8024–8035.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.

[50] L. N. Smith, "Cyclical learning rates for training neural networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 464–472.

[51] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Tokyo, Japan, Nov. 2013, pp. 1321–1326.

[52] S. James, M. Freese, and A. J. Davison, "PyRep: Bringing V-REP to deep robot learning," 2019, *arXiv:1906.11176*.

Matija Mavsar received the M.Sc. degree in electrical engineering from the Faculty of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia, in 2018, where he is currently pursuing the Ph.D. degree.

He is employed at the Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia. His research focuses mainly on the use of deep learning and motion prediction in robotics.

Barry Ridge received the B.Sc. degree in computer applications from Dublin City University, Dublin, Ireland, in 2002, the M.Phil. degree in pure mathematics from the University of St Andrews, St Andrews, Scotland, in 2006, and the Ph.D. degree from the University of Ljubljana, Ljubljana, Slovenia, in 2014, with a focus on robotic learning of object affordances.

He is currently a Robotics Technologist with the NASA Jet Propulsion Laboratory (JPL), California Institute of Technology, Pasadena, CA, USA. He has previously held post-doctoral positions at the Advanced Telecommunications Research Institute International, Kyoto, Japan, and the Jožef Stefan Institute, Ljubljana, Slovenia. Since joining JPL in 2020, he has been working on robot vision and simulation capabilities for a variety of missions, including Mars Sample Return, the DARPA Subterranean Challenge, Europa Lander, and others. His research interests include cognitive robotics, computer vision, and machine learning.

Dr. Ridge was awarded the Marie Curie Fellowship.

Rok Pahič received the M.Sc. degree in mechanical engineering from the University of Maribor, Maribor, Slovenia, in 2016, and the Ph.D. degree from the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia, in 2021.

His research focuses mainly on modularization of robotic software and deep learning in robotics.

Jun Morimoto (Member, IEEE) received the Ph.D. degree in information science from the Nara Institute of Science and Technology, Nara, Japan, in 2001.

From 2001 to 2002, he was a Post-Doctoral Fellow with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. He joined the Advanced Telecommunications Research Institute International (ATR), Kyoto, Japan, in 2002. He was also a Researcher and a Group Leader of Japan Science and Technology Agency - The International Cooperative Research Project (JST-ICORP) Computational Brain Project from 2004 to 2009. He is currently a Professor with the Graduate School of Informatics, Kyoto University, Kyoto. He is also the Head of the Department of Brain Robot Interface (BRI), ATR Computational Neuroscience Laboratories, and a Senior Visiting Scientist of the Man-Machine Collaboration Research Team, Guardian Robot Project, RIKEN, Kyoto.

Aleš Ude (Member, IEEE) received the Diploma degree in applied mathematics from the University of Ljubljana, Ljubljana, Slovenia, in 1990, and the Dr.Eng. degree in sciences from the University of Karlsruhe, Karlsruhe, Germany, in 1995.

He is currently the Head of the Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana. He is also a Visiting Researcher with the ATR Computational Neuroscience Laboratory, Kyoto, Japan. He has been a coordinator and/or a principal investigator of numerous national and international projects in these areas. His research interests include robot learning, imitation learning, reconfigurable robotic systems, and humanoid robotics.