Transformer-based Neural Augmentation of Robot Simulation Representations

Agon Serifi[1,2], Espen Knoop[2], Christian Schumacher[2], Naveen Kumar[2], Markus Gross[1,2], Moritz Bächer[2]

Abstract—**Simulation representations of robots have advanced in recent years. Yet, there remain significant sim-to-real gaps because of modeling assumptions and hard-to-model behaviors such as friction.**

In this paper, we propose to augment common simulation representations with a transformer-inspired architecture, by training a network to predict the true state of robot building blocks given their simulation state. Because we augment building blocks, rather than the full simulation state, we make our approach modular which improves generalizability and robustness.

We use our neural network to augment the state of robot actuators, and also of rigid body states. Our actuator augmentation generalizes well across robots, and our rigid body augmentation results in improvements even under high uncertainty in model parameters.

Index Terms—**Deep Learning Methods, Simulation and Animation, Neural Augmentation, Robotics, Dynamics.**

I. Introduction

In robotics, simulation plays a role of ever-increasing importance. Yet, even with state-of-the-art techniques, discrepancies between simulations and reality (sim-to-real gaps) are observed. This is in part due to modeling assumptions (e.g., no deformation of rigid bodies, no backlash, no friction), and in part due to model inaccuracies (e.g., errors in rigid body mass properties, tolerances during assembly). Actuator drives may often also implement control loops where the detailed implementation is undisclosed, making them difficult to model accurately. While additional verification and characterization experiments can reduce the sim-to-real gap, this adds complexity and does not scale well. Here, we instead tackle this problem with a learning-based data-driven approach.

Concretely, we propose a transformer-based architecture that augments simulation representations of individual building blocks robots are made of. For each class of building blocks, we train a separate State Augmentation Transformer (SAT), taking the time-varying state and interaction forces with other entities as input. We minimize the sim-to-real gap with a physics-informed loss that compares augmented simulation states to measurements taken from physical robots. As we demonstrate with several examples, the augmented state consistently improves the prediction quality, both when augmenting *actuator* states and *rigid body* states.

Manuscript received: February 09, 2023; Accepted April 18, 2023.

This paper was recommended for publication by Editor Cecilia Laschi upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by ETH Zurich and Disney Research.

[1]Computer Graphics Lab, Department of Computer Science, ETH Zurich: {aserifi|grossm}@inf.ethz.ch

[2]Disney Research, Zurich/Glendale: first.lastname@disney.com

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

Our actuator augmentation generalizes well across robots that are built with the same actuators. Training a single augmentation for all rigid components of a robot, we show that our augmentation improves simulation prediction even under high uncertainty in model parameters. Note that due to the higher dimensionality of rigid bodies, the rigid body augmentation does not currently generalize across different robots.

Succinctly, our technical contributions are:

- a transformer-based neural augmentation of simulation representations for a large class of robots consisting of rigid components, mechanical joints, and actuators.
- a modular augmentation approach that interfaces with all common simulation representations of dynamical systems and could be extended to other robot building blocks.

Our method enables accurate digital twin representations of robots, with applications including more accurate state estimation, and improved closed- and open-loop control. While we here augment rigid components and actuators, we keep our formal description general and would expect our method and modular approach to also extend to other building blocks.

II. Related Work

Our transformer-based augmentation shares similarities with architectures commonly used in natural language processing and time series forecasting [21], [26]. We first review related architectures, followed by a discussion of neural simulation representations.

1) Neural Architectures for Time Series Forecasting: Recurrent Neural Networks (RNNs) [23] have been applied to time series forecasting tasks, with Long Short-Term Memory (LSTM) cells seeing widespread use [6], [16]. However, their limited short-term memory [5], [22] is insufficient for simulation augmentation as we demonstrate with a series of experiments.

Transformers [27] overcome these limitations with an attention mechanism, providing long-term memory. While transformers lead to state-of-the-art performance on problems ranging from natural language [27] to image [10] and audio [3] processing, we have not seen the use of this architecture in augmentations of simulation representations. We base our architecture on the Temporal Fusion Transformer (TFT) proposed by Lim et al. [20]. We confirm that a combination of short- and long-term memory outperforms an attention-only architecture, and is well-suited for the problem domain we consider here.

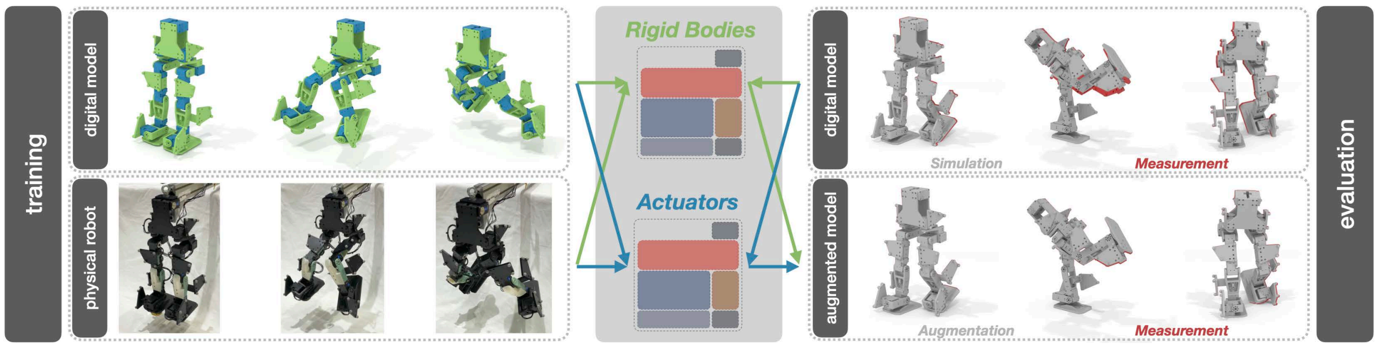

Fig. 1. **Overview.** As input, our processing takes reference motion that is sent to a physical robot, and also a digital model thereof (left). The physical behavior of the robot results in partial measurements of its time-varying state, while simulations (digital model) provide full state estimates. We first sort the different components, with their corresponding simulated and measured state, into categories (rigid bodies in green, actuators in blue). We then train a State Augmentation Transformer (SAT) per category, minimizing a loss that penalizes differences between augmented and measured states (middle). To apply augmentations to simulations of robots (right), we proceed analogously (reference motion, digital model), unifying the individual augmentations after evaluation (augmented model). See also the supporting video.

2) Neural Simulation: Differentiable simulation [7], [9], [12], [13], [17], [25] enables a tight integration of simulation and data-driven techniques. For example, Geilinger et al. [12] integrate a differentiable simulator as a last node of a neural network to train control policies. However, differentiable simulators are not widely available. An advantage of our technique is that it interfaces with a wide range of standard simulators, augmenting states of building blocks robots are made of.

Golemo et al. [14] train an RNN to predict differences between simulated and real-world behavior and integrate it with the simulator to learn more robust patterns. Although differentiable simulators allow gradient-based methods for parameter fitting, they are limited by the underlying approximate model. Hence, it is impossible to fully close the sim-to-real-gap. Recently, data-driven methods were introduced to learn additional nonlinear dynamics, where neural networks are used to enhance the simulators. Kloss et al. [18] integrate a neural network to augment the input of an analytic model into the simulation pipeline. This approach, however, keeps the simulation within the space of the analytical model and is therefore still restricted. On the other hand, Ajay et al. [1] apply a variational RNN [8] as a post-process, allowing the augmentation to go beyond the expressiveness of an analytic model. Heiden et al. [15] present a hybrid simulator combining a physics engine with a neural network to augment simulation variables. Their approach is trained in an end-to-end fashion, specializing in specific tasks. In contrast to previous works, we focus on a decoupled augmentation of simulation states and show that this modular approach can generalize over different robots assembled from the same building blocks. Additionally, our transformer-based model augments the simulation states while consuming a more extended history of previous states, mitigating error accumulations over time.

III. Overview

We consider here the dynamic simulation of robots made of rigid components which are driven by actuators and may also be coupled together with mechanical joints.

1) Training: During the training phase (Fig. 1, left), we send representative reference motions to the physical robot and its digital model, resulting in partial measurements and simulations of the time-varying state of the robot.

We then group the building blocks into categories. For most robots, we define two categories: one for rigid components (Fig. 1, in green), and one for actuators (in blue). We then train a separate SAT (middle) for each category, by minimizing a loss that penalizes differences between augmented simulation states and corresponding measurements.

2) Evaluation: To augment simulations of the same or a different physical instance of the same robot performing a different task or motion (right), we proceed analogously: We first simulate the robot's time-varying behavior, then group the robot's components into the same categories as defined during training (digital model). We then evaluate the SATs for each component separately, resulting in an augmentation of the full state of the robot at every time step (augmented model).

IV. Modular Augmentation of Simulation Representations

Before delving into the specifics of our transformer architecture, we will discuss how we prepare simulation data and measurements for training.

To allow for generalization, we propose to decompose a simulation representation into building blocks, then learn an augmentation for *all* building blocks of a particular type or class. To decompose simulation representations, we utilize that the robot is in an equilibrium at every time step. This method scales well because we can augment *any* robot that is made of the same building blocks. Moreover, the technique is extensible because we can easily add new SATs for new types of building blocks.

A. Decomposing Dynamics Simulations of Robots

A dynamic simulation computes the time-varying state of the robot, along with the forces and torques on each component, such that it is in equilibrium at any point in time. The interactions between components manifest themselves as forces and torques at the actuators and joints. To isolate a component, we, therefore, consider its state along with the computed time-varying forces and torques acting on it (in green in Fig. 2).

We here interface with a maximal-coordinate simulation representation [11], to allow for robots with arbitrary kinematics, including series-parallel structures. However, our approach could also be readily used with a reduced-coordinate simulation formulation.

The mechanical behavior of an individual building block can be represented with position and velocity quantities, $\mathbf{p}(t)$ and $\mathbf{v}(t)$, uniquely describing its state $\mathbf{s}(t)$. For rigid bodies, the position quantities are the pose of the body, and the velocity quantities consist of its linear and angular velocities. For actuators, we are interested in the position and velocity of the output shaft, whose state is fully described with a 2D vector.

Building blocks are coupled with a set of constraints $\mathcal{C}$, implementing the degrees of freedom of mechanical joints and actuators. For rigid components, the constraint forces $\mathbf{f}_{\mathcal{C}}(t) = \mathcal{C}_{\mathbf{p}}^T \boldsymbol{\lambda}$, with Lagrange multipliers $\boldsymbol{\lambda}(t)$, enable the decoupled simulation of the building block.

Succinctly, in this work, we seek to learn an augmentation $\Delta\mathbf{s}(t)$ of the time-varying states $\mathbf{s}(t)$ to account for modeling uncertainties in the equations of motions

$$\begin{array}{rcl} \dot{\mathbf{p}} & = & \mathbf{v} \\ \dot{\mathbf{v}} & = & \mathbf{M}^{-1}(\mathbf{p})(\mathbf{f}(\mathbf{p},\mathbf{v}) + \mathbf{f}_{\mathcal{C}}) \end{array} \quad \text{with state} \quad \mathbf{s} = \begin{bmatrix} \mathbf{p} \\ \mathbf{v} \end{bmatrix}, \tag{1}$$

of mechanical components, with generalized mass matrix $\mathbf{M}$, forces $\mathbf{f}$ that model gravity, damping and other body forces, and constraint forces $\mathbf{f}_{\mathcal{C}}$ that we extract from a simulation.

Rather than training an augmentation for each building block in isolation, we seek to train a network that outputs an augmentation $\Delta\mathbf{s}(t)$ when fed with $\mathbf{s}(t)$ and $\mathbf{f}_{\mathcal{C}}(t)$ of building blocks of a particular class. For example, we seek to train a single SAT that generalizes across *all* rigid bodies. This is possible as long as all building blocks in a particular class have the same number of state variables and constraint force components (e.g., six for rigid bodies). Note that if there are robots that consist of components of vastly different size, it could make sense to train several SATs for a particular class.

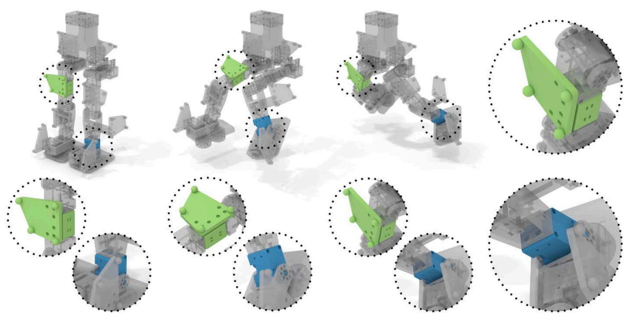

Fig. 2. **Modular Augmentation.** We decouple the motion of building blocks a robot is made of by recording the time-varying forces and torques that neighboring building blocks exert on them. By doing so, we could simulate their motion in isolation.

In our simulation, we model actuators with a PID controller and a standard brushed DC motor model. The state of an actuator is described by its (1D) position, velocity, current, and voltage, and it takes as input reference values $\mathbf{r}(t)$ of position, velocity, and torque. Analogously to rigid bodies, we augment classes of actuators that we can represent with the same number of state variables. To this end, we record the constraint forces, which in this context represent the dynamics of the part of the robot the actuator is driving. In contrast to a mechanical system, however, our neural augmentation takes the reference curves $\mathbf{r}(t)$ as additional inputs.

B. Measurements

Our actuators use built-in sensors to measure their position, velocity, voltage, and current, which we use for training our actuator SATs. To train augmentations of the mechanical components, we add motion markers (four spheres on the green component in Fig. 2, zoom-in, top, right), then track them using a commercial tracking system to obtain the time-varying rigid body trajectory. These data sources result in partial measurements $\bar{\mathbf{s}}(t)$ of the full state.

C. Reference Motions

To collect a representative and sufficiently large dataset, we draw end-effector trajectories from randomly generated B-Spline curves of varying order $(0-4)$. We then use an inverse kinematics formulation [24] to generate the reference curves $\mathbf{r}(t)$ that we send to the simulator and the physical robot. We refer to the accompanying video for an example of a reference motion.

D. Problem Statement

For every class of building blocks, our neural augmentation takes decoupled states $\mathbf{s}(t)$, constraint forces $\mathbf{f}_{\mathcal{C}}(t)$, and partial state measurements $\bar{\mathbf{s}}(t)$ as input, learning an augmentation $\Delta\mathbf{s}(t)$ that minimizes the difference between the augmented $\mathbf{s}(t) + \Delta\mathbf{s}(t)$ and measured state $\bar{\mathbf{s}}(t)$.

To measure differences between states and partial measurements, we introduce a constant matrix $\mathbf{S}$ that selects and transforms full-state quantities so that we can numerically compare them to measurements. In loss functions, we then compare $\mathbf{S}\left(\mathbf{s}(t) + \Delta\mathbf{s}(t)\right)$ to $\bar{\mathbf{s}}$. To avoid unnecessary transformations during training and evaluation, we preprocess the data, subtracting the transformed state, $\mathbf{S}\mathbf{s}(t)$, from the measurements, then comparing

$$\Delta\bar{\mathbf{s}}(t) = \bar{\mathbf{s}} - \mathbf{S}\mathbf{s}(t) \quad \text{to} \quad \mathbf{S}\Delta\mathbf{s}(t) \tag{2}$$

in loss functions.

V. Transformer-based Augmentation

Our SAT is an instance of the *Temporal Fusion Transformer (TFT)* introduced by Lim et al. [20]. The TFT was designed for a variety of forecasting tasks. All base modules required for an implementation are supported by Beitner et al. [4]. In our setting, there is only a causality going forward in time (i.e. the current state is only influenced by previous states). Moreover, unlike most forecasting tasks, our model consumes an initial state estimate and predicts an additive residual value.

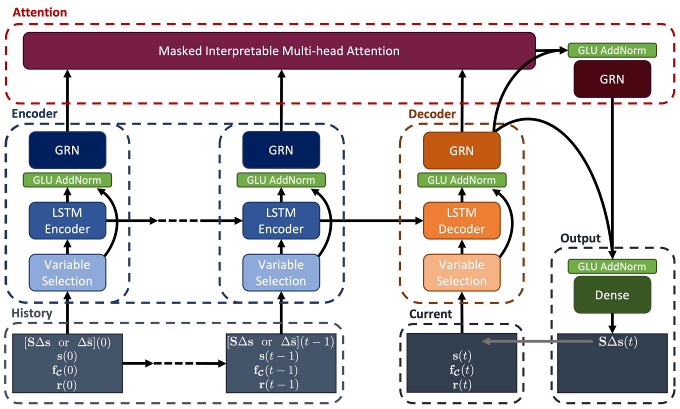
Attention
Masked Interpretable Multi-head Attention
GLU AddNorm
GRN
Encoder
GRN
GLU AddNorm
LSTM Encoder
Variable Selection
GRN
GLU AddNorm
LSTM Encoder
Variable Selection
Decoder
GRN
GLU AddNorm
LSTM Decoder
Variable Selection
Output
GLU AddNorm
Dense
History
$[\mathbf{S}\Delta\mathbf{s}$ or $\Delta\bar{\mathbf{s}}](0)$ $\mathbf{s}(0)$ $\mathbf{f}_{\mathcal{C}}(0)$ $\mathbf{r}(0)$
$[\mathbf{S}\Delta\mathbf{s}$ or $\Delta\bar{\mathbf{s}}](t-1)$ $\mathbf{s}(t-1)$ $\mathbf{f}_{\mathcal{C}}(t-1)$ $\mathbf{r}(t-1)$
Current
$\mathbf{s}(t)$ $\mathbf{f}_{\mathcal{C}}(t)$ $\mathbf{r}(t)$
$\mathbf{S}\Delta\mathbf{s}(t)$

Fig. 3. **State Augmentation Transformer (SAT).** For every class of components, we learn an augmentation $\Delta\mathbf{s}(t)$ of the current state $\mathbf{s}(t)$, taking the simulated constraint forces $\mathbf{f}_{\mathcal{C}}(t)$ and reference curves $\mathbf{r}(t)$ as additional inputs. Curved arrows represent skip connections.

Our custom instance of the more general TFT is shown in Fig. 3.

The encoder of our SAT consumes the previous simulation states, together with the difference between measured and simulated states, $\Delta\bar{\mathbf{s}}$, or the previously predicted error $\mathbf{S}\Delta\mathbf{s}(t)$. This is motivated by the teacher forcing approach proposed by Williams and Zipser [28]. At the initial time step, we assume the error to be zero or known. For electromechanical components, we can feed the network with measurements even during the evaluation phase. For rigid bodies, we feed the network with predictions from previous time steps or set them to zero. The decoder only consumes the current observable state.

Both the encoder and decoder transform the simulation states into a latent representation. This embedding has aggregated local information and the SAT continues with a multi-head attention layer [27]. The attention-weighted combination of the embeddings now encodes global information and is further processed in a final output layer which predicts a residual value for the current simulation state. For the backbone of the encoder/decoder, we experimented with different architectures, including dense fully-connected layers or the LSTM cells that are used in the TFT.

The model makes use of *Variable Selection Networks (VSN)* to assign varying importance to the input dimensions, suppressing irrelevant or disturbing fields in each prediction step.

Many hyperparameters can be chosen in our architecture. We can reliably estimate some of them (e.g., hidden dimensionality or numbers of attention heads) using hyperparameter optimization frameworks [2]. An interesting hyperparameter in our setting is the number of previous states or the length of the history that the model consumes in a single prediction step. In our experiments (Sec. VI), we show the attention profile of the TFT and how it improves prediction performance.

A. State Augmentation

To perform state augmentations, the model must handle fields with different scales and units. We apply standard normalization. For example, we define four global scale factors for state quantities that describe the position and orientation of a body and its linear and angular velocity. We then scale all coordinates of a particular 3-dimensional quantity with the same global scale factor.

For orientations, we rely on a 6-dimensional representation that is continuous and results in smaller errors for regression tasks [30]. This representation is obtained by dropping the last column of a 3×3 rotation matrix. To recover the rotation matrix, we perform a Gram-Schmidt orthogonalization step on the first two columns.

B. Training

A supervised training procedure is used where the SAT minimizes an objective function between the predicted $\mathbf{S}\Delta\mathbf{s}$ and the measured error $\Delta\bar{\mathbf{s}}$ at each time step. To reduce the final sim-to-real gap of the simulation, we hence want to reduce an objective based on the absolute state values as, e.g., the $\mathcal{L}_1$ loss.

As we are augmenting a simulation where physics must hold, we also propose an additional physics-informed term for position and velocity augmentations, where we penalize differences between the time derivative of the position and the velocity. The full physics-inspired loss is given as

$$\mathcal{L}_{PL1} := \mathcal{L}_1(\mathbf{S}\Delta\mathbf{s}, \Delta\bar{\mathbf{s}}) + \mathcal{L}_1\left(\frac{d}{dt}\boldsymbol{\theta}(\mathbf{S}\Delta\mathbf{s}), \dot{\boldsymbol{\theta}}(\mathbf{S}\Delta\mathbf{s})\right), \quad (3)$$

where $\boldsymbol{\theta}(\cdot)$ and $\dot{\boldsymbol{\theta}}(\cdot)$ extract the position and velocity from the state augmentation, respectively, and we use finite differences to approximate the time derivative of the position. We demonstrate later that the additional loss reduces overfitting (Sec. VI-D2).

To compare two orientations $\mathbf{R}_1$ and $\mathbf{R}_2$, we rely on the geodesic distance

$$D(\mathbf{R}_1, \mathbf{R}_2) = \cos^{-1}\left((tr(\mathbf{R}_1\mathbf{R}_2^{-1}) - 1)/2\right). \quad (4)$$

VI. Experimental Results

For the following experiments, we consider three robot configurations as shown in Fig. 4. KickBot is a small custom humanoid robot (height 438mm, mass 3.08kg, 12 DoFs), which we either attach to mechanical ground at the pelvis (KickBotA) or on one foot (KickBotB). These two configurations lead to significantly different forces and torques, and also show that our method can handle changes in contact configurations as would be seen for legged robots. DanceBot is a small biped (height 325mm, mass 2.58kg, 12 DoFs), with series-parallel kinematics, where most of the actuators are placed in the body, and we fix its feet to mechanical ground. The robots are driven with *Dynamixel XM430-W350-R* actuators and are 3D printed. The KickBot has motion capture markers attached to seven rigid bodies.

For the training data, we sampled 600 trajectories of 30s each, sampled at 250Hz with the method described in Sec. IV-C. The test data was collected from artist-created animations. The animations allow us to show generalizability over motions not sampled from the same underlying model used for the training data.

During the training of the SAT models, we never showed any data from KickBotB or DanceBot; the only training data comes from KickBotA. This makes our training-test split strong, and allows us to show generalizability over new, unseen robot configurations. The robots are built using the same actuators.

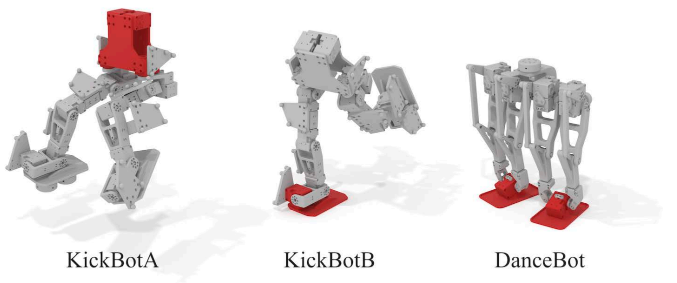

Fig. 4. **The robots.** KickBotA (attached at the pelvis), KickBotB (attached at the right foot), and DanceBot (attached at both feets). Fixed components are shown in red.

A. Actuator Modeling

This section presents the augmentation capabilities of the SAT model on actuator states, which include position, velocity, and electric current. For this test, the SAT is solely trained with randomly sampled animations simulated and measured on KickBotA, and evaluated on unseen artist-created animations on KickBotB and DanceBot. Thus, we demonstrate that the model generalizes over different robots and over unseen animations. We compare our modular approach against a non-modular version of the SAT, where the augmentation takes the state of all actuators as input and learns to augment all states simultaneously.

In the first experiment, an instance of the modular and non-modular SAT learns to augment the actuators' position and velocity states simultaneously, with the physics-informed loss.

Fig. 5 shows bar plots summarizing the augmentation performance on the training robot (KickBotA) and the unseen robot (KickBotB) for position and velocity, respectively. We evaluate the models on three metrics: the Concordance Correlation Coefficient (CCC) [19]; the Mean Absolute Error (MAE); and the max error. A higher CCC indicates that the state follows the up and down trend of the measurements more accurately. For the max error and MAE, we evaluate the deviation between the measurements and the simulated/augmented states per trajectory, and look at the error distribution over all test trajectories.

The simulation shows a max error of $>1^o$ on average, which the SAT state augmentation reduces to 0.6^o. The non-modular approach performs slightly better on the training robot (KickBotA), especially since the variance is smaller. However, it performs worse and has almost no improvement on the KickBotB. This suggests that the non-modular approach overfits the error patterns of the specific actuators that do not generalize to new acting forces and torques. Our proposed modular approach, however, achieves similar performance on the new configuration.

Fig. 7 shows an example of the state of an actuator over a window of 4s. The augmented simulation matches the

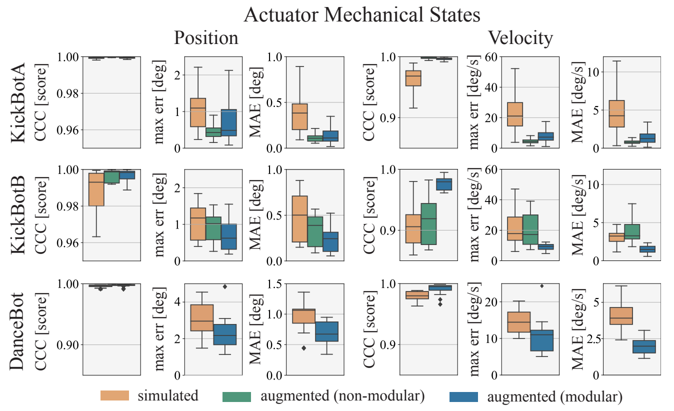

Fig. 5. **Actuator Augmentation Evaluation.** Position and velocity, for KickBotA and KickBotB non-modular vs. modular approach. And modular approach on unseen DanceBot.

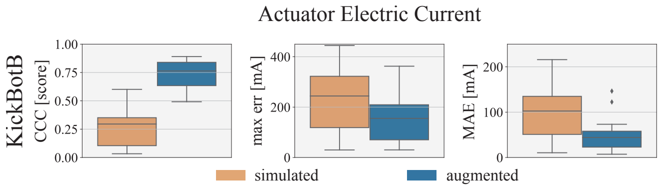

Fig. 6. **Actuator Augmentation Evaluation.** Electric current, for KickBotB.

measured state for position and velocity more closely than the initial simulation. Note that the measurements are quantized, and we avoid pre-filtering in our training pipeline. Due to the physics-based loss, we see that the velocity augmentation follows the expected smooth profile although supervised on the quantized measurements.

We further show the augmentation capabilities on an unseen robot with a different topology (DanceBot, Fig. 5). The difference between the two unseen robots is the higher torque and complexity of the DanceBot, resulting in a more significant simulation error. Note that the non-modular approach is inherently topology-dependent by design, and can not be applied to a different robot. Our modular method can augment the actuator states and reduces the max simulation error from $\sim 3^o$ to $\sim 1^o$. In general, the state augmentation shows that the mean and variance of the simulation error can be reduced significantly for both unseen robots.

Besides the mechanical, the model can also learn to augment the electric states. We show the results of a modular SAT instance trained on predicting the error of the electric current of actuators in Fig. 6. The SAT increases the CCC from 0.25 to 0.75, showing that the model can capture the trends of the electric current more accurately.

B. Rigid Body Modeling

We repeated similar experiments for the rigid bodies. We tracked multiple rigid bodies of the KickBot with a motion capture setup while performing the randomly generated trajectories. We train an instance of the SAT to predict augmentations for the 3D positions of the rigid bodies, and a second SAT on augmenting the 6D representation of the orientations. We show results for the KickBot in Fig. 8. For position, we measure the Euclidean distance, and for orientation, we evaluate the geodesic distance (Eq. 4).

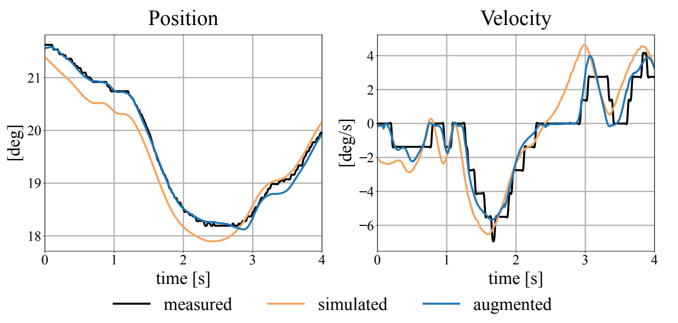

Fig. 7. **Actuator Augmentation Trajectories.** Illustrative example of time-varying trajectories. Measured (black), simulated (orange), augmented (blue).

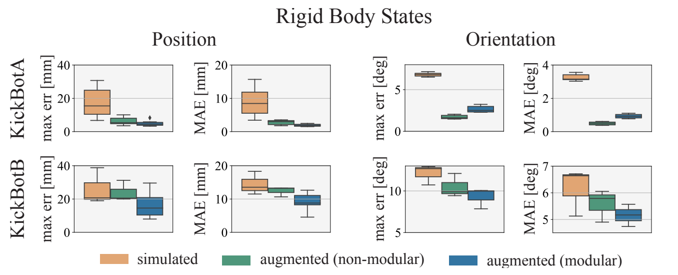

Fig. 8. **Rigid Body Augmentation Evaluation.** Position and orientation for KickBotA and KickBotB non-modular vs. modular approach.

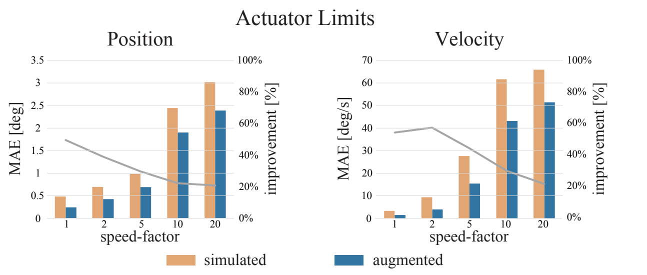

Fig. 9. **Augmentation Limits** Position and velocity MAE of KickBotB on artist-designed motion for different speed-factors under and over the actuator limits. Gray lines show percentage improvement.

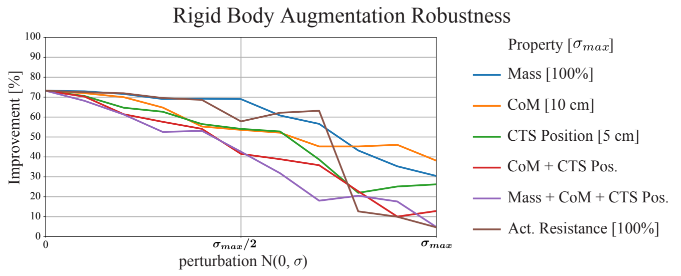

Fig. 10. **Augmentation Robustness** Augmentation around sampled rigid bodies is robust for large perturbations in Mass properties, shifting center of mass (CoM), moving constraint positions (CTS Pos.) and combinations. We also see robust behavior when changing actuator parameters like the resistance.

The simulation has an average max position error of ∼17.5mm on KickBotA. Both the non-modular and modular approach can reduce the error by a factor ∼3.5, to ∼5mm. A similar effect is seen in the MAE.

The rigid body orientation error sees a similar reduction, with max error reduced from 7^o to $\sim 2.5^o$ and MAE reduced from 3.5^o to $< 1^o$. The non-modular approach performs similarly or better on the training robot but performs worse on a new robot configuration.

In general, the augmentation method performs better on actuators. The reason for this is the higher dimensional space of the rigid bodies that are more difficult to sample densely.

C. Augmentation Limits

We further investigate the limitations of our augmentation. In a first experiment, we show augmentation limits when approaching actuator limits (Fig. 9). For this, we speed up the artist's animation by factors of $1-20$x. We see that the augmentation performance declines as we reach the actuator limits. At 10x speed-up, we are simulating motions that exceed the actuator limits; while the method can not achieve the same performance as before, it still improves the simulation states.

Our rigid body augmentation is robust to model variations, i.e. increasing sim-to-real gaps, as we demonstrate with the following experiments where we perturb simulation parameters with Gaussian noise of increasing magnitude. Concretely, we perturb each rigid body mass by $m_\epsilon = m\epsilon$, $\epsilon \sim N(1,\sigma)$, shift each rigid body center of mass by $COM_\epsilon = COM + \epsilon$, $\epsilon \sim N(0,\sigma)$, and constraint positions by $CTS_\epsilon = CTS + \epsilon$, $\epsilon \sim N(0,\sigma)$ for increasing values of σ up to $\sigma_{\max}$ as indicated in Fig. 10. For multiplicative noise, we ensure values remain non-negative by clipping to a small positive value. Note that we set $\sigma_{\max}$ to a large value (cf. Fig. 10), well beyond what could be expected as a sim-to-real gap, in order to stress-test our method.

As seen in Fig. 10, the rigid body augmentation is robust for a large perturbation and still achieves 50% improvement against the simulation for $\sigma_{max}/2$ and over 25% improvement for σ_{max}-perturbation. We also show the perturbation of the actuator's resistance parameter ($r_\epsilon = r\epsilon$, $\epsilon \sim N(1,\sigma)$), which shows a similar trend. Similar effects are observed when perturbing other actuator parameters. This shows that the augmentation is robust in a large neighborhood under uncertainty in simulation parameters and is evidence of generalization across different robot instances.

D. Ablation Studies

In the following, we evaluate the importance of the components of the SAT model as well as alternatives, give intuition into the attention mechanism, investigate different loss functions, and show how a physics-motivated term helps the model converge and stabilizes the training.

1) Architecture: The final architecture combines recurrent neural layers with an attention-based module. To evaluate the importance of each component, we investigated the modules' performance in isolation and with alternative architectures as the encoder/decoder backbone. Each instance was trained to augment the position state of the actuators. All models saw the same training data, and were evaluated on unseen animations. The results are summarized in Table I, where we report the MAE and Max error of the base simulation, and the augmentation performance of the different architectures. The LSTM model without attention results in an increased error, while the attention models can consistently improve the base simulation and reduce the MAE and Max error. A more complex encoder/decoder backbone has slightly better

TABLE I
ARCHITECTURE EVALUATION ON KICKBOTA. LOWEST VALUES IN BOLD. ACT.: ACTUATORS, RBS: RIGID BODIES, ATTN.DENSE: ATTENTION + DENSE.

	Simulation		**LSTM**		**Attention**		**Attn.Dense**		**SAT**	
	MAE	Max	MAE	Max	MAE	Max	MAE	Max	MAE	Max
Act. [deg]	0.48	1.10	1.50	2.80	0.26	0.82	0.26	0.76	**0.24**	**0.71**
Rbs. [mm]	8.72	17.52	2.78	7.48	**1.87**	5.57	2.43	6.12	1.90	**5.03**

performance. To verify the results, we also train instances on augmenting position states for the rigid bodies of KickBotA, see Table I. We see similar results here, where the attention models beat a pure LSTM model and reduce the mean and max error of the rigid body position.

In the following, we discuss the results for each architecture in more detail.

a) RNN: We removed the attention part of the model and investigated the output of the Variable Selection Network (VSN) [20] followed by the LSTM layer. The VSN acts as a pre-filtering of the input fields, avoiding the disturbance of the limited LSTM cells caused by less important or irrelevant fields. This pure LSTM approach stagnates on long animation sequences leading to an increased max- and mean-error of the simulation, on average by a factor of 2. We observe that the stagnation of the model is slower on rigid bodies, and the augmentation beats the base simulation on the 30s-long animations. Besides a lower error accumulation, this also suggests that there are some simpler error patterns for rigid bodies where an RNN can already improve. The MAE can be reduced by 70% from 8.7mm to 2.8mm.

b) Attention: We investigate whether attention can improve the results by removing the LSTM and using the multi-head attention layer with its final fully connected output module. In this configuration, the model applies attention to the output of the VSN module, and has an unlimited perceptive field without information losses. The transformer aggregates the past and current simulation states together, and a final dense output layer predicts the residual of the position state. This very simple model significantly reduces the sim-to-real gap. The average improvement on the unseen robot is 45% for the mean error and 25% for the max error. The attention model also further improves the results on the rigid bodies by reducing the MAE by 80%. The aggregation with the attention layer is powerful enough for a simple dense output layer to outperform the RNN model consistently and significantly over a larger prediction period. This shows the attention layer to be crucial in the architecture and sufficient for achieving a good augmentation. However, we further investigate the composition of both architecture types as proposed in the initial TFT model and alternatives.

c) Encoder/Decoder: Starting from the simple transformer model above, we investigated different extensions of the encoder/decoder. The first version *Attention Only* is the previously described model with no additional layers attending directly to the output of the VSN. *Attention + Dense* has two additional Dense ReLU layers after the VSN. Finally, *Attention + LSTM* describes the encoder/decoder architecture proposed in the TFT with an LSTM layer that is recurrently applied over the history sequence. The additional components in the

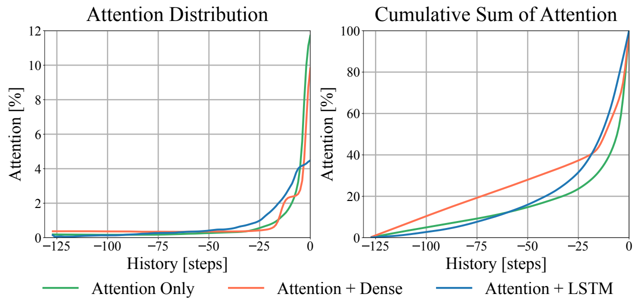

Fig. 11. **Attention over 128 history states.** The attention profile for the three encoder/decoder variations. Left: Normalized average attention per history step. Right: Cumulative sum of the attention distribution.

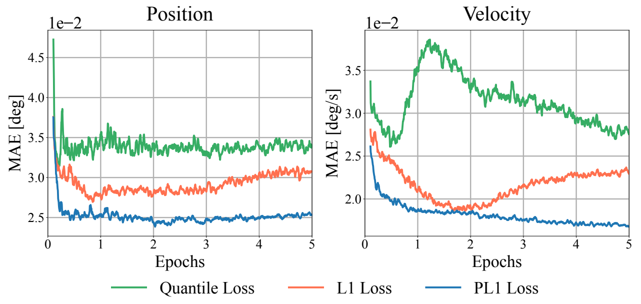

Fig. 12. **Validation Loss.** We trained the SAT model to simultaneously augment the position and velocity state. Here we show the Mean Absolute Error (MAE) on an unseen motion while training with three different objectives.

encoder/decoder lead to slightly better results. Notably, the Max error can be reduced by a further 5% by the dense layers and 10% by the LSTM layer on average for both robots. We, therefore, further investigate their effects on the attention layer.

Fig. 11 shows the normalized average attention of each architecture over the input states in percentage. We see that the *Attention Only* and *Attention + Dense* models assign over 10% of their attention to the single previous history state. This is significantly more then the *Attention + LSTM* model with only 4.5%. The *Attention + Dense* instance assigns similar attention over the last 10 states. In the cumulative sum of the attention plot, shown in Fig. 11 (right), we observe that using no additional layers in the encoder assigns only 27% of the attention to states that are 20 or more steps in the past. Using the dense layers increases this attention to 40%, but assigns it almost equally along the states. The model with LSTM-cells shows a more gradual decrease of attention for past states.

Thus, we refer to this final attention model with an LSTM encoder/decoder as the State Augmentation Transformer (SAT). In contrast to the TFT, the SAT consumes no future or static covariates and acts as a residual network predicting the correction of the current simulation state.

2) Physics-Informed Loss: We evaluated the convergence of the SAT when trained on different loss functions. We train an SAT model to augment the actuators' position and velocity state simultaneously. Fig. 12 evaluates the MAE of an unseen animation during the training process. We use the Quantile, $\mathcal{L}_1$, and the Physical-Informed $\mathcal{L}_1$ loss ($\mathcal{L}_{PL1}$). In contrast to usual forecasting tasks, we observe that the Quantile loss does not fit our problem. Its statistical nature does not concentrate on reducing the absolute error between the simulation and the measurements. We observe that the MAE of the position augmentation saturates quickly and increases temporarily for the velocity. With the $\mathcal{L}_1$ loss, we see a significant decrease in the MAE. The model's predictive power increases, and we learn more general applicable patterns from the training data. Over the training time, however, we observe an overfitting effect that leads to increasing position and velocity errors, which is a known issue with neural networks [29]. The $\mathcal{L}_{PL1}$ loss shows a lower MAE. This loss acts as a filtering process that reduces the effect of noise and mitigates overfitting.

VII. Conclusion

We have devised a transformer-based augmentation of several robot building blocks, and demonstrated that we can learn actuator augmentations that generalize well to other robots, and rigid body augmentations that are robust under uncertainty in modeling parameters. We have presented augmentations for positions, velocities, orientations, and electric currents of components. Moreover, we have evaluated the attention mechanism and argued for the fused LSTM-Transformer architecture. Additionally, we have presented an idea of introducing physical information into the loss function and shown this regularization to positively affect model convergence.

One limitation of the augmentation is that the method has no mechanism to keep constraint violations low. While this allows the augmentation to move out of the constrained simulation space and more accurately match the measured data, it could also lead to unwanted constraint violations for scenarios with insufficient training data. While we only observed relatively small constraint violations in our examples, additional loss terms that minimize such a coupled error metric could be an interesting direction to explore.

While we have demonstrated generalization of actuator augmentations across different robots, our rigid body augmentation does not generalize to new robots. Because we observe good generalization in a neighborhood of a moving rigid body, we believe that a generalization across robots made of different rigid components could be possible with an extensive sampling of rigid body properties and behaviors, which we leave as future work.

Soft components are typically simulated using finite elements. As the number of elements vary per component, our modular approach is not directly applicable. The augmentation of general soft bodies with a single transformer is a future direction which we plan to investigate further.

References

[1] Anurag Ajay, Jiajun Wu, Nima Fazeli, Maria Bauza, Leslie P Kaelbling, Joshua B Tenenbaum, and Alberto Rodriguez. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In *IEEE/RSJ IROS*. IEEE, 2018.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[3] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *NeurIPS*, 33, 2020.

[4] Jan Beitner and colleagues. Pytorch forecasting. https://github.com/jdb78/pytorch-forecasting, 2020.

[5] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2), 1994.

[6] Filippo Maria Bianchi, Enrico Maiorino, Michael C Kampffmeyer, Antonello Rizzi, and Robert Jenssen. An overview and comparative analysis of recurrent neural networks for short term load forecasting. *arXiv preprint arXiv:1705.04378*, 2017.

[7] Justin Carpentier and Nicolas Mansard. Analytical derivatives of rigid body dynamics algorithms. In *Robotics: Science and Systems*, 2018.

[8] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *NeurIPS*, 28, 2015.

[9] Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J Zico Kolter. End-to-end differentiable physics for learning and control. *NeurIPS*, 31, 2018.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[11] Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *IEEE ICRA*, 2015.

[12] Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. Add: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)*, 39(6), 2020.

[13] Markus Giftthaler, Michael Neunert, Markus Stäuble, Marco Frigerio, Claudio Semini, and Jonas Buchli. Automatic differentiation of rigid body dynamics for optimal control and estimation. *Advanced Robotics*, 31(22), 2017.

[14] Florian Golemo, Adrien Ali Taiga, Aaron Courville, and Pierre-Yves Oudeyer. Sim-to-real transfer with neural-augmented robot simulation. In *Conference on Robot Learning*. PMLR, 2018.

[15] Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. In *IEEE ICRA*, 2021.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[17] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *ICLR*, 2020.

[18] Alina Kloss, Stefan Schaal, and Jeannette Bohg. Combining learned and analytical models for predicting action effects from sensory data. *The International Journal of Robotics Research*, 2020.

[19] I Lawrence and Kuei Lin. A concordance correlation coefficient to evaluate reproducibility. *Biometrics*, 1989.

[20] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4), 2021.

[21] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philos. Transactions of the Royal Society A*, 2021.

[22] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*. PMLR, 2013.

[23] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088), 1986.

[24] Christian Schumacher, Espen Knoop, and Moritz Bächer. A versatile inverse kinematics formulation for retargeting motions onto robots with kinematic loops. *IEEE Robotics and Automation Letters*, 6(2), 2021.

[25] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012.

[26] José F Torres, Dalil Hadjout, Abderrazak Sebaa, Francisco Martínez-Álvarez, and Alicia Troncoso. Deep learning for time series forecasting: a survey. *Big Data*, 9(1), 2021.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017.

[28] Ronald Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1989.

[29] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3), 2021.

[30] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF CVPR*, 2019.