Domain Adaption as Auxiliary Task for Sim-to-Real Transfer in Vision-based Neuro-Robotic Control

Connor Gäde, Jan-Gerrit Habekost and Stefan Wermter
Knowledge Technology, Department of Informatics
University of Hamburg
Hamburg, Germany
{connor.gaede, jan-gerrit.habekost, stefan.wermter}@uni-hamburg.de

***Abstract*—Architectures for vision-based robot manipulation often utilize separate domain adaption models to allow sim-to-real transfer and an inverse kinematics solver to allow the actual policy to operate in Cartesian space. We present a novel end-to-end visuomotor architecture that combines domain adaption and inherent inverse kinematics in one model. Using the same latent encoding, it jointly learns to reconstruct canonical simulation images from randomized inputs and to predict the corresponding joint angles that minimize the Cartesian error towards a depicted target object via differentiable forward kinematics.**

We evaluate our model in a sim-to-real grasping experiment with the NICO humanoid robot by comparing different randomization and adaption conditions both directly and with additional real-world finetuning. Our combined method significantly increases the resulting accuracy and allows a finetuned model to reach a success rate of 80.30%, outperforming a real-world model trained with six times as much real data.

***Index Terms*—domain adaption, domain randomization, sim-to-real, kinematics, robot manipulation, humanoid robot**

I. Introduction

Developing vision-based object manipulation abilities is essential for humanoid robots to interact with realistic environments. Generating the large amounts of data required to train such tasks successfully is time-consuming and expensive [1]–[4], requires multiple, identical robot setups to parallelize [2], [3] and wears them out in the process [2].

Simulators provide a more accessible and scalable environment to collect training data. Existing robot learning frameworks such as RLBench [5] provide a variety of different tasks and robots that can be extended with new ones.

However, due to the difficulty of vision-based robot manipulation policies to generalize to new environments [6], a model trained on simulated data with approximated visuals and dynamics is often unable to perform well in a real-world environment [4], [7]. While domain randomization [7] can directly make a policy more transferable by randomizing different aspects of the simulation environment, domain adaption approaches utilizing a separate GAN model [8] to transfer images into the input domain of the policy, with mechanisms to maintain consistent visual features, have proven to be more effective at closing this *reality gap* [9]–[11].

Another challenge in learning robot manipulation tasks is the choice of action representation. An end-to-end model that predicts the target angles for each motor is directly applicable to the robot and ensures that the solution lies within its workspace boundaries, whereas policies using Cartesian end effector actions perform better in tabletop manipulation tasks [12], [13], but require an additional motion planner or inverse kinematics solver to transfer the solution into joint space and apply it to the robot [13], [14]. As such, rather than making the policy itself more robust to these challenges, the addition of models that solve these issues separately allows it to operate under simplified conditions. However, each model adds training and execution time as well as additional sources of error, whereas end-to-end models can be directly applied to the robot and have been shown to benefit from additional image reconstruction losses [15] or integrating multiple action representations with a differentiable kinematic module [12].

Therefore, we examine the benefits of leveraging domain randomization and adaption as well as differentiable forward kinematics as error signals for an end-to-end model that directly predicts joint angles for a given input image. In order to reconstruct the target object in the output image, the latent space of a domain adaption autoencoder has to include an encoding of that object's position, which should make its encoder reusable for grasp pose estimation. Predicting joint angles can be viewed as inherently solving the inverse kinematics problem to reach the object pose. Thus, instead of calculating the error in joint space, using differentiable forward kinematics allows us to compute a loss in Cartesian space between the robot's end effector and the target object.

We propose an end-to-end architecture that combines a randomized-to-canonical adaption network inspired by RCAN [9] with a joint angle predictor trained to directly minimize the error in Cartesian space using differentiable forward kinematics similar to CycleIK [16]. We create a simulated grasping experiment for the NICO humanoid robot [17] to autonomously collect training data with matching randomized and canonical images and replicate our experiment in a real-world setup to transfer and compare to. To train our model, we define a loss function combining the Euclidean and geodesic distance between the position and orientation of the target object and the forward kinematics for the predicted joint angles. First, we demonstrate that our method increases the grasping accuracy of our end-to-end model compared to alternative losses and subsets of our method in simulation.

The authors gratefully acknowledge partial support from the German Research Foundation DFG under the projects CML and LeCAREbot.

We then evaluate each model in our real-world environment before and after finetuning with real-world data. Our method is able to increase the direct transferability of the model and with additional finetuning can outperform a model exclusively trained on six times more real-world data.

II. Related Work

A. Sim-to-Real Transfer

Models trained in simulation often perform poorly when transferred to the real world due to the visual and dynamic differences. Approaches to overcome this issue can be categorized into domain randomization and domain adaption [4].

1) Domain Randomization: In order to make a trained model more robust to changes in its environment, domain randomization methods alter different visual features [7] or dynamics [18] of the scene for the model to interpret the real environment as another variation of the simulation. James et al. [19] utilize the known position information in simulation to train a pick and place task from trajectories generated with inverse kinematics and significantly increase the transferability of the model by altering visual features such as lighting, colors, and textures, the positions and sizes of objects as well as the start configuration of the robot. The approach presented by Gäde et al. [20] trains a NICO robot in simulation to reach an object on a table by first randomly generating valid arm poses within a predefined set of constraints and then placing the target object at the resulting position of the hand. They show that randomizing colors of the scene as well as the head pose of the robot to alter the camera angle increases the real-world performance of the model.

2) Domain Adaption: Rather than improving the robustness of a model towards changing input data, domain adaption [21] transforms input images into the target domain. RL-CycleGAN [10] jointly trains a CycleGAN [22] to transform simulated into real images and a reinforcement learning model for vision-based robotic grasping, ensuring consistency between the original and generated images in both domains by adding an additional loss term which penalizes differences in predicted Q-values. Similarly, RetinaGAN [11] also maps simulated to real images with a CycleGAN, but instead of task-specific Q-values, it ensures perception consistency by computing the bounding box and class prediction loss of an object detector between the images. RCAN [9] combines domain adaption with domain randomization by training an adaption network to transform randomized simulation images into canonical simulation images, enabling the model to transfer real-world images into canonical simulation images as well. They show that their model outperforms regular domain randomization in a grasping task, both when transferred directly and after additional finetuning with real-world data.

B. End-to-End Robot Learning

As the end effector position of the arm results from the combination of its joint angles, close or redundant positions in Euclidean space can require large differences in joint angle space. This makes it more difficult to train a model to directly predict joint angles rather than the target pose for an inverse kinematic solver [13], [15]. Rahmatizadeh et al. [15] combine a VAE-GAN for image reconstruction with an autoregressive Mixture Density Network for joint prediction sharing the same encoder to learn the distribution of each arm joint in sequence from human demonstrations for different robot manipulation tasks. Instead of explicitly learning actions for a given observation, Florence et al. [23] propose the use of implicit models, which learn an energy function for given observation-action pairs by identifying correct actions from random counterexamples. Inference is done via stochastic optimization. They show that these models outperform explicit approaches in multiple robot control tasks. Chi et al. [24] introduced Diffusion Policies, which learn to iteratively retrieve actions from the Gaussian noise input for a given observation.

C. Differentiable Forward Kinematics

A direct way to reflect the relationship between angle space and Cartesian space is to compute the forward kinematics for a given joint configuration to obtain the position of one or more joints. As these calculations are differentiable, they can be used for backpropagation. Pavllo et al. [25] used differentiable forward kinematics with a standardized human skeleton to create a position-based loss function for human pose estimation. Ganapathi et al. [12] extended an implicit model with a differentiable forward kinematic module to use both joint angles and the corresponding Cartesian position as input, showing that this increases performance in difficult robot manipulation tasks. Lastly, CycleIK by Habekost et al. [16] is trained to compute the inverse kinematics for the NICOL robot [13] by calculating differentiable forward kinematics for the predicted joint angles and computing the loss on the input target position in Cartesian space, achieving position errors below $1mm$ and rotation errors under $1°$.

III. Approach

In our approach, we investigate enhancing the performance and transferability of an end-to-end grasping model by jointly learning random-to-canonical domain adaption and predicting joint angles for grasp poses that minimize differentiable forward kinematics, thereby inherently solving inverse kinematics. We evaluate our approach in a grasp pose estimation task trained on a simulated NICO humanoid robot and transferred to a real-world setup. To collect our training data, we develop an autonomous recording method with an evolutionary inverse kinematics solver, which creates matching randomized and canonical simulation images for the same target position. A real-world variant is used to create additional data with the physical robot to finetune and test our model.

We create a neural architecture based on convolutional autoencoders and determine the optimal hyperparameters to solve our task in simulation. With the optimized architecture, we evaluate the performance of different subsets of our model components, first in simulation, then in reality, both before and after finetuning the models with additional real-world data.

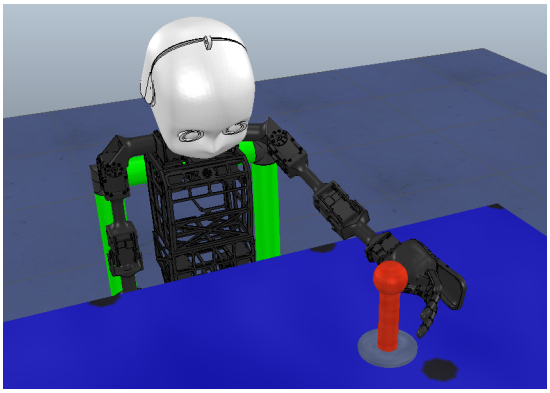


Fig. 1. Simulated environment of a seated NICO robot grasping a target object on a table

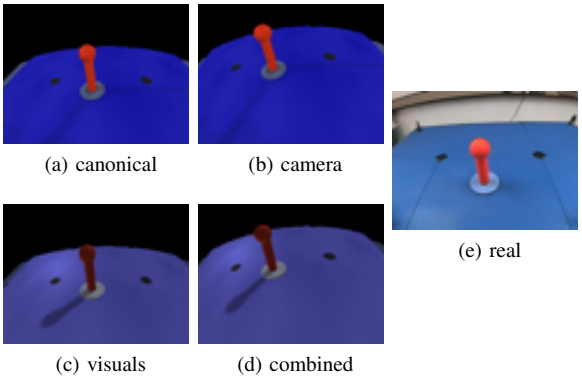


Fig. 2. Matching images for the same sample with different randomization conditions (a-d) as well as a real-world comparison (e)

A. Experimental Setup

Our experimental setup expands upon the grasp pose estimation task which was first proposed in [26] and is further investigated in other experiments with the NICO [20], [27] and NICOL robot [13]. A cylindrical object is placed on a table in front of a NICO humanoid robot, which has to reach the object with its left hand to grasp it from the side.

The arms of the NICO robot are actuated by four Dynamixel MX64 servomotors, three of which rotate the upper arm in any direction, while the fourth serves as the elbow. Additionally, a three-fingered SeedRobotics RH4D hand is mounted on each arm, whose rotation can be adjusted with two wrist motors, with another two actuators opening and closing the fingers. The palm of each hand also contains an infrared distance sensor to detect whether an object is placed within it. Another two MX64 motors are used to pose the head of the NICO robot, which contains two cameras as its eyes.

Using the CoppeliaSim simulator [28], we recreate this setup with a simulated NICO robot (see Fig. 1). Control instructions can be used interchangeably between the simulated and real robot. The simulated controller is based on the PyRep [29] library, which controls the simulation environment through external Python scripts.

B. Dataset Recording

With our simulated setup, we create a dataset of 5184 simulated training samples and 324 test samples containing matching images for four different levels of randomization which are simultaneously collected for each data point, with one canonical image maintaining a consistent visual representation, while the other three have different visual features randomized. Additionally, we collect 2268 real-world samples to test the transferability of our approach and finetune models with additional real-world data points. Each sample contains an image of the scene from the right eye camera of the robot, as well as the joint angles of the 6 left arm motors and the corresponding three-dimensional coordinates and orientation of the left hand to grasp the target at the depicted location.

To collect our data, we expand the strategies proposed by previous works [20], [26], [27] in which the robot autonomously places the object at different positions on the table to generate samples. Rather than relying on human demonstration [26] or random pose generation [20], we create EvoIK, an evolutionary inverse kinematics solver inspired by gaikpy [27], which minimizes the weighted sum of the position and orientation distances defined in Section III-E3 using the gpu-accelerated CMA-ES [30] implementation provided by EvoTorch [31] combined with the forward kinematics of PyTorch Kinematics [32]. With EvoIK, we generate grasping poses from a grid of 18×18 predefined positions on the table with $1cm$ horizontal and vertical spacing between them to systematically cover the robot's workspace. For each row in the grid, the robot repeatedly pushes the object with its left arm from the leftmost position to the right in $1cm$ increments. At each position, the robot tries to enclose the object with its hand to verify the grasping success before the arm is moved back into its initial, pre-grasp position to record images of the object on the table with the right eye camera of the robot.

As the simulation has access to the position of both the target object and the end effector of the robot, the grasp verification is done by ensuring the Euclidean distance between the two is less than $2cm$. On an unsuccessful grasp or at the end of each row, the simulator can immediately reset the arm pose and position the object for the next trial.

In contrast, the real-world robot has no direct knowledge of the position of the target object. Therefore, we use the infrared proximity sensor embedded in its palm to detect if the object is enclosed by the hand. To move the object to the next row, the robot grasps it at its last known position and physically moves it across the table. However, at the beginning of the recording and in case of failure, the robot requires human assistance to set the object to a known position. For this purpose, the robot assumes a grasping pose next to the current target position and asks a human observer to place the object in its open hand.

C. Domain Randomization

In order to increase the visual variety in our simulated data and allow a model to convert input images into a canonical representation to improve transferability to the real world, we create two randomization mechanisms to manipulate the visual

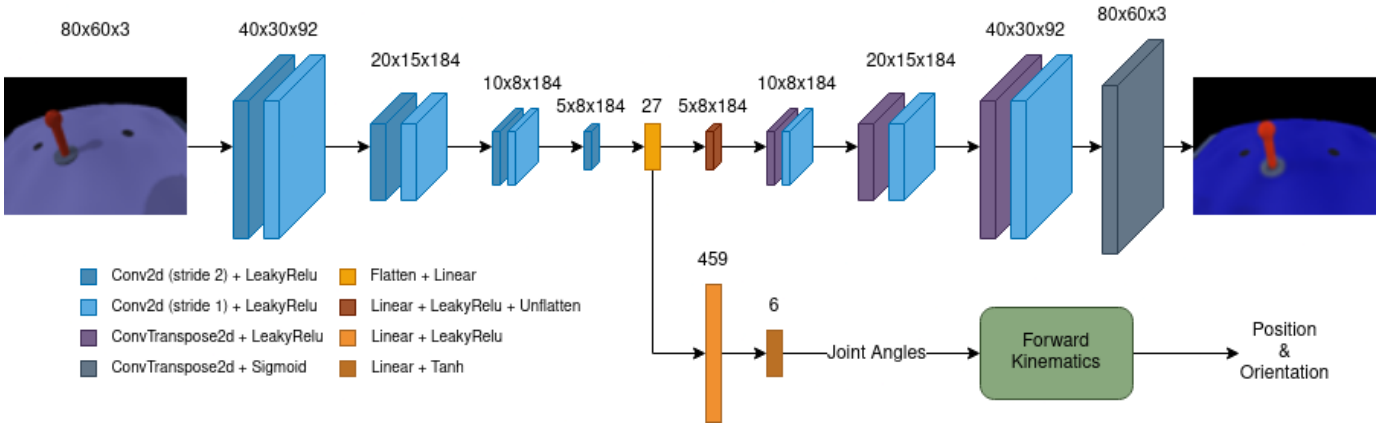


Fig. 3. Optimized network architecture to learn end-to-end visuomotor manipulation from randomized simulation data to improve simulation to real-world transferability. The model consists of a convolutional autoencoder for randomized-to-canonical domain adaption and a fully connected joint control network sharing the same encoder. A differentiable forward kinematics module computes the end effector pose for the predicted joint angles, which allows calculation and backpropagation of the loss in Cartesian space

features of the scene, which we employ both separately and combined to create three additional variants of each image.

As the recorded colors in real-world data are dependent on multiple factors such as lighting conditions, surface texture, and camera settings, we attempt to account for these differences by randomizing the colors, light sources, and textures of the scene. The colors are randomized by sampling the *lightness* channel of the HSL (hue, saturation, lightness) representation of each color from a normal distribution with a standard deviation of 0.2 around its base value. For the light sources, we place each of the four default light sources of the simulator randomly within an area of $0.25m^2$ at a fixed height above the table and assign a random orientation. The textures in our simulated scene are generated with a Perlin noise function [33], which is initialized with a random positive integer seed by default, whereas our canonical representation has a fixed seed of 42.

Due to the ego-movement of the robot, which can cause minor shifts in the robot's position and thus its head angle, and limitations of the simulated cameras, the angle and distortion of the camera differ between the simulated and real-world data. To counteract this, we randomly alter the camera angle of the simulated robot for each sample. In doing so, the resulting model should be able to learn a more generalized representation of the object's position relative to the table as it can no longer rely on the absolute position in the image. The camera angle is determined by the joint angles of the head, which is controlled by two separate motors rotating around the horizontal y-axis (pitch) and the upwards-facing z-axis (yaw). By default, the y-axis motor is positioned at $40°$ and the z-axis at $0°$, which results in the head looking downward in front of its center. Both of these angles are randomized by up to $10°$ in either direction, resulting in camera angles of $30°$ to $50°$ on the y-axis and $-10°$ to $10°$ on the z-axis.

For each collected sample, we first record a canonical image with the default colors and camera angle. Then we change the camera angle for the second image and alter the visual features of the scene for the third image, combining both randomization conditions. Lastly, we return to the default camera angle and record a fourth image with only the random visual features applied. In doing so, we create four consistent variant images for each data point, allowing the reconstruction of the canonical representation from the randomized images.

D. Network Architecture

We construct an end-to-end architecture to predict the joint angle for each of the six arm motors for a given image of the scene. Our model consists of a convolutional autoencoder, which transforms randomized input images into the corresponding canonical representation and a controller network of two fully connected layers to predict the joint angles of the arm required to reach the target object depicted in the image. Similar to Rahmatizadeh et al. [15], the encoder and latent vector are shared between both networks (see Fig. 3).

As our task only involves the area in front of the robot, it does not require the large field of view provided by the fisheye cameras of the NICO robot. Therefore, we pre-process our RGB input images by first cropping out the central region of the original 640×480 image before downsampling it to our input dimensions of 80×60. To account for differences between the distortion of the real and simulated cameras, we use slightly different crop areas so that the resulting, downsampled images depict the object in similar locations of the image to prevent unpredictable out-of-distribution behaviour. For the real-world images, we simply crop the central 320×240 region of the image whereas for the simulation images, we shift the cropped region 35 pixels towards the top of the image and reduce its height to 180 pixels.

Our encoder contains 7 convolutional layers whose strides alternate between 2 and 1, downsampling the input image 4 times without pooling. Likewise, our decoder mirrors this structure by alternating between a total of 4 transposed convolutions of stride 2 to upscale the latent vector back to the

TABLE I
SEARCH SPACES AND RESULTS FOR CONVOLUTIONAL FILTERS (CON), LATENT UNITS (LAT), NUMBER OF LINEAR LAYERS (#) AS WELL AS UNITS (LIN) OF THE MODEL AND INITIAL LEARNING RATE (γ), BETAS (β_1, β_2), EPSILON (ϵ) AND WEIGHT DECAY (l_2) OF ADAM.

	Con	Lat	#	Lin	γ	β_1	β_2	ϵ	l_2
Min	4	16	1	16	1e-5	0.85	0.95	1e-8	1e-7
Max	128	512	4	512	1e-2	1.0	1.0	1e-4	1e-4
Res	92	27	1	459	6e-4	0.86	0.98	1.3e-6	1.1e-6

original dimensions and 3 convolutions of stride 1. Each layer has 3×3 kernels and a zero-padding of 1 to replace the lost edges. All hidden layers in the autoencoder use the leaky ReLU activation. For the output of the decoder, we choose the Sigmoid activation to produce normalized RGB channels.

To predict the joint angles required to reach the target at the depicted position, the latent vector of the encoder is also processed by a fully connected layer of 459 neurons with the leaky ReLU activation, followed by an output layer containing six neurons with the hyperbolic tangent activation corresponding to the individual arm joints with normalized limits within $[-1, 1]$. Each angle is scaled according to the respective motor limits in radians set by the NICO software API, which restricts the robot's action space to reduce the number of unnatural movements and potential self-collisions.

E. Network Training

We conduct a hyperparameter optimization of 128 trials with the Quasi-Monte Carlo sampler provided by Optuna [34] to find our optimizer parameters as well as the number of convolutional filters, linear layers, and units (see Table I). Each model is trained for 400 epochs using 6-fold cross-validation with batch size 54, an Adam optimizer [35] with weight decay, and a learning rate scheduler which halves the learning rate if the validation loss plateaus for 10 epochs. Both the image decoder and the joint angle prediction are trained simultaneously by computing the sum of the image reconstruction loss $\mathcal{L}_{rec}$ and the Cartesian pose loss $\mathcal{L}_{pose}$ as our total loss for backpropagation.

1) Image Reconstruction Loss: Instead of reconstructing the randomized input images, we train our model to reconstruct the corresponding canonical image with default camera angle and visual features, thus performing random-to-canonical domain adaption to invert the domain randomization, similar to RCAN [9]. This way, we condition the model to learn a standardized representation, which should improve its ability to interpret real-world data. Therefore, we compute the image reconstruction loss $\mathcal{L}_{rec}$ by taking the mean squared error between the decoder output and the normalized canonical image for the given randomized input image.

2) Cartesian Pose Loss: While an end-to-end model that directly predicts the target angles for each motor makes the output directly applicable to the robot and ensures the solution lies within its workspace boundaries, this increases the training complexity as the model has to learn to inherently solve the inverse kinematics of the robot due to the end effector pose resulting from the combined motor angles. Therefore, computing the individual joint errors does not accurately represent the distance to the target object and can lead to averaged solutions with poor grasping performance.

Instead, we first determine the forward kinematics for the predicted joint angles to obtain the position and orientation of the end effector and then calculate the loss in Cartesian space between the predicted and the target end effector pose. We compute the forward kinematics using the PyTorch Kinematics library [32], which allows parallel processing of batches on the GPU and computes the necessary gradients for backpropagation. The end effector poses are represented as 7-D vectors consisting of the position as x, y, z coordinates in meters and the orientation as unit quaternion. Rather than computing a single error for the full vector, we compute two separate losses $\mathcal{L}_{pos}$, for the 3-D positional part of the output, and $\mathcal{L}_{quat}$ for the 4-D unit quaternion.

For $\mathcal{L}_{pos}$ we use Smooth L1 loss [36], which squares absolute errors below a threshold β to smooth the gradient towards 0, while computing the L1 loss for greater errors to be less susceptible to outliers. We choose $\beta = 0.01$ to incentivize our model to bring the positional error below $1cm$.

Our rotation loss $\mathcal{L}_{quat}$ is calculated by taking the mean cosine distance for half of the minimal rotation angle between the unit quaternions of our target and the predicted end effector pose, which is a valid rotation metric according to Huynh [37], eliminating the need to compute the inverse cosine:

$$\mathcal{L}_{quat}(q_{pred}, q_{target}) = 1 - |q_{pred} \cdot q_{target}| \tag{1}$$

Additionally, we define a scaling factor α_{quat} for $\mathcal{L}_{quat}$ and sum it with $\mathcal{L}_{pos}$ to obtain the full pose loss $\mathcal{L}_{pose}$:

$$\mathcal{L}_{pose} = \mathcal{L}_{pos} + \alpha_{quat}\mathcal{L}_{quat} \tag{2}$$

We choose $\alpha_{quat} = 0.33$ such that $\mathcal{L}_{quat} \approx \mathcal{L}_{pos}$ for a positional error of $1cm$ and an orientation error of 20°, matching the thresholds of our success metric.

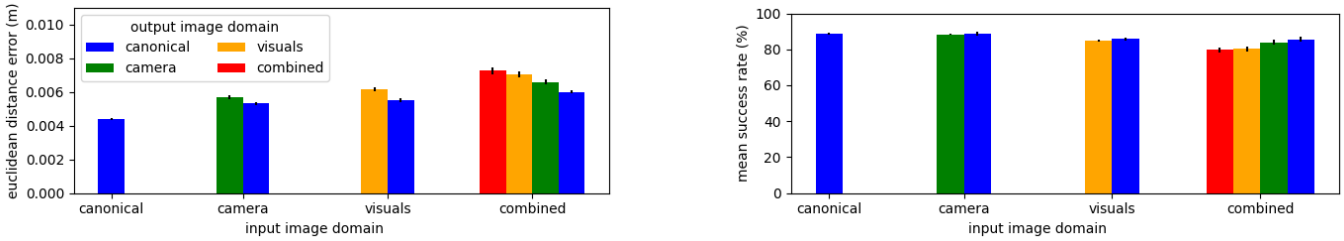
3) Evaluation Metrics: In addition to these losses, we evaluate our model using more interpretable distance metrics for the position and orientation of the predicted end effector pose. The positional error is determined by the three-dimensional Euclidean distance between the target and predicted end effector positions in meters. For the orientation, we calculate the minimum rotation angle θ between both unit quaternions:

$$\theta = 2 \arccos\left(|q_{pred} \cdot q_{target}|\right) \tag{3}$$

Both distance metrics are then used to compute the success rate of the model under the definition of Kerzel et al. [27], which considers a grasp pose as successful if the positional error is below $1cm$ and the orientation error under 20°.

IV. EXPERIMENT RESULTS

We conduct multiple experiments to determine the influence of the individual components of our architecture on its performance. First, we show that using differentiable forward



(a) (b)

Fig. 4. Mean Euclidean distance error (a) and percentage of successful grasps (b) in simulation for each evaluated input and output image domain. The models with a *canonical* output domain (blue) attempt to reconstruct images with canonical camera angle and visual features, whereas the *camera* domain (green) retains the randomized camera angle and *visuals* (yellow) the visual features. The *combined* domain (red) remains fully randomized

TABLE II
LOSS FUNCTION COMPARISON

Loss Function	Precision (cm)	Grasp Success
MSE	1.39	46.96%
Smooth L1	1.03	61.21%
Cartesian	**0.60**	**85.55%**

kinematics to calculate the loss on the end effector pose in Euclidean space leads to a more precise model than computing the loss directly over the individual arm joints. Then, we analyze how different levels of domain randomization and their adaption to more canonical representations affect the performance of the model in simulation and when evaluated with real-world data. Finally, we finetune our model with a small amount of real-world data and demonstrate that we can outperform a model trained exclusively on the real robot.

A. Loss Function Comparison

To evaluate the benefit of our loss function calculated in Cartesian space using differentiable forward kinematics, we optimize our architecture within the same search space and parameters as defined in Table I, altering the loss function to directly minimize the mean squared error of the predicted joint angles as well as Smooth L1 loss with $\beta = 0.01$ for comparison. As we can see in Table II, both of these losses result in a positional error above $1cm$, with mean squared loss resulting in the lowest average precision of $1.39cm$ whereas models trained with Smooth L1 loss can reach $1.03cm$. The generated poses are sufficient to grasp the object in 46.96% and 61.21% of the test cases respectively. In contrast, models trained with our Cartesian loss function utilizing forward kinematics reach an average precision of $6.0mm$ resulting in a success rate of 85.55%, outperforming both other methods. This shows that differentiable forward kinematics can provide our end-to-end model with better context to internally solve the inverse kinematic problem on highly randomized visual information, reaching similar precision as a dedicated neural IK solver such as IKFlow [38] or CycleIK [16].

B. Domain Randomization and Adaption

We retrain our architecture with different input and output image domains to analyze the influence of varying levels of domain randomization and random-to-canonical adaption on the model performance in simulation. For each of our four simulated image domains, we compare the possible encoder output domains, either reconstructing the input image or reproducing a target image with non-randomized visual features, camera angle, or both, if applicable.

As we can see in Fig. 4, the performance decreases for more complex input domains, with the fully canonical model achieving $4.4mm$ precision and 88.84% grasp successes, whereas the autoencoder with fully randomized in- and outputs has an average position error of $7.3mm$ and only 79.68% successful grasps. Altering the camera angle decreased the performance less than randomizing the visual features of the scene.

Consistent with our observations on the input domain, reducing the complexity of the output domain increases the performance of the resulting model, with the least improvement for models trained on randomized camera angles from $5.7mm$ position error and 88.22% successes to $5.3mm$ position error and 88.68% successes whereas the performance of models trained on randomized visuals increases from a precision of $6.16mm$ and success rate of 84.77% to $5.5mm$ and 85.91% respectively. The largest improvement is seen in the full random-to-canonical models with an improvement of $1.3mm$ in average distance and 5.8% successes, resulting in a positional error of $6.0mm$ and a success rate of 85.55%, even outperforming an autoencoder model trained with random visuals but default camera angles. Therefore, we can conclude that using the latent space of a random-to-canonical domain adaption model instead of a regular autoencoder improves the generalization of the model in more visually complex domains.

C. Simulation to Real-world Transfer

1) Direct Transfer: To examine if our previous results hold true when deployed to the physical robot, we evaluate each model with our real-world test data. Additionally, we optimize a baseline model trained only on our full real-world dataset to compare it to the performance of our other models. As we can see in Fig. 5, most of the models are unable to adapt directly to real-world inputs with positional errors above $5cm$. Training a model to predict canonical camera angles without also randomizing the visuals seems to greatly increase the position error to $16.96cm$. Full domain randomization with no domain

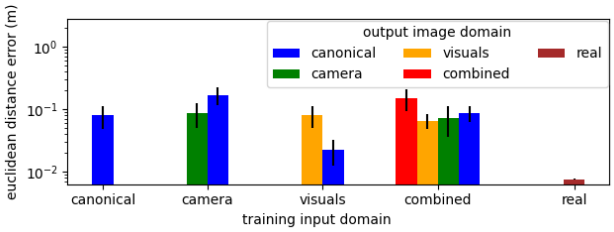
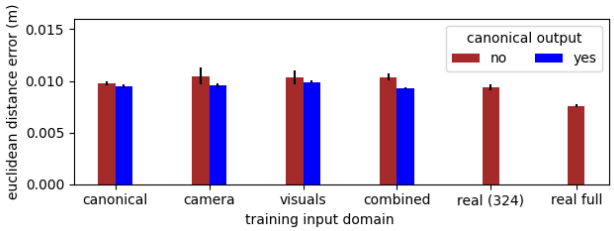
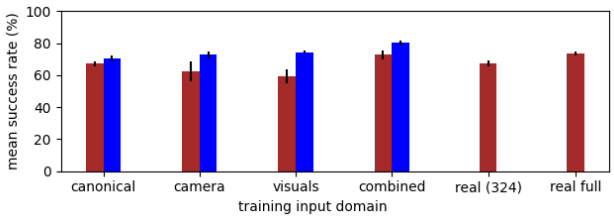


Fig. 5. Mean Euclidean distance error when directly evaluating each model on real-world test data



(a)



(b)

Fig. 6. Mean Euclidean distance error (a) and percentage of successful grasps (b) on real-world test data after finetuning the models trained in simulation with additional real-world training samples

adaption yields the second worst precision at $15.13cm$. While any form of random-to-canonical adaption brings this error below $10cm$, they still do not sufficiently transfer to the real world. However, training a model to predict canonical images from randomized visual features with the default camera angle significantly decreases the real-world position error to $2.27cm$ with a standard deviation of $1cm$, resulting in a success rate of 6.79%. Therefore, while none of the models are able to directly solve the real-world task with high accuracy, we can see that our method significantly increases the ability to interpret real-world inputs over regular domain randomization.

2) Finetuning: Thus, in order to increase the real-world performance of our models trained in simulation, we finetune them for 10 epochs using 6-fold cross-validation with a single batch of 54 real-world samples for each fold. The Adam parameters are optimized in another 128-step Quasi-Monte Carlo search with the search space defined in Table I. We also train a real-world baseline on the same 324 samples.

Notably, while all finetuned models reach between $0.93cm$ and $1.05cm$ positional accuracy (see Table III), the highest errors of over $1cm$ are produced when training models with domain randomization but no random-to-canonical adaption,

TABLE III
MEAN EUCLIDEAN DISTANCE ERROR AND PERCENTAGE OF SUCCESSFUL GRASPS ON REAL-WORLD TEST DATA AFTER FINETUNING

Train	Adaption	Precision (cm)	Grasp Success
canonical	no	0.98 ± 0.02	$67.13\% \pm 1.58$
canonical	yes	0.95 ± 0.02	$70.73\% \pm 1.80$
camera	no	1.05 ± 0.08	$62.71\% \pm 6.19$
camera	yes	0.96 ± 0.02	$72.79\% \pm 2.09$
visuals	no	1.03 ± 0.06	$59.41\% \pm 4.13$
visuals	yes	0.99 ± 0.01	$74.33\% \pm 0.90$
combined	no	1.04 ± 0.03	$72.74\% \pm 2.56$
combined	yes	$\mathbf{0.93 \pm 0.01}$	$\mathbf{80.30\% \pm 1.59}$
real (324)	no	0.94 ± 0.03	$67.54\% \pm 1.86$
real (1944)	no	$\mathbf{0.76 \pm 0.01}$	$73.56\% \pm 1.05$

performing worse than both the model with only canonical training data with $0.98cm$ precision and our real-world baseline with $0.94cm$ (see Fig. 6). Likewise, the models trained on just one of the two randomization conditions generate less successful grasps with 62.71% for random camera angles and 59.41% for visual features, compared to the canonical and real-world baseline models with 67.13% and 67.54% successful grasps respectively. Using the fully randomized data does however increase the success rate to 72.74%, despite the higher distance error, which is close to the real-world model trained on all 1944 samples at 73.56%.

Once we finetune our models with random-to-canonical adaption, the success rates increase to over 70%, outperforming the baseline real-world model trained on just the finetuning data. The models trained solely on random camera angles or visual features reach similar success rates of 72.79% and 74.33% compared to the model trained on the full real-world dataset with 73.56%. Finally, if we finetune our model trained with random-to-canonical adaption on fully randomized data, the resulting model reaches a success rate of 80.30%, significantly outperforming the full real-world model, despite its positional error of $9.3mm$ not matching the $7.6mm$ precision of the real-world model. Therefore, we conclude that our model is able to find a better trade-off between position and orientation error, generating more generalized poses within our success metric at the cost of positional accuracy.

V. CONCLUSION

In this paper, we investigated the benefit of random-to-canonical domain adaption and differentiable forward kinematics for the training and sim-to-real transfer of a vision-based end-to-end model for humanoid robot grasping. To train our architecture, we collected a dataset of 22032 simulated images divided into 4 matching sets with full, partial, and no domain randomization as well as 2268 real-world samples for finetuning and testing. Our approach was able to significantly increase the success rate of the model in simulation and improve its ability to learn from highly randomized training data. We demonstrated that a model trained to perform random-to-canonical adaption as an auxiliary task is more robust to the sim-to-real gap compared to only domain randomization. With additional finetuning, our method is able to outperform a model fully trained on real-world data.

ACKNOWLEDGMENT

The authors thank Erik Strahl and Matthias Kerzel for their valuable feedback and guidance on the experimental setup and development of the architecture, as well as Philipp Allgeuer for his insights on orientation metrics and model optimization.

REFERENCES

[1] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 3406–3413.

[2] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[3] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on Robot Learning*. PMLR, 2018, pp. 651–673.

[4] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A survey on learning-based robotic grasping," *Current Robotics Reports*, pp. 1–11, 2020.

[5] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.

[6] A. Xie, L. Lee, T. Xiao, and C. Finn, "Decomposing the generalization gap in imitation learning for visual robotic manipulation," *arXiv preprint arXiv:2307.03659*, 2023.

[7] 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 *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[9] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 627–12 637.

[10] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari, "Rl-cyclegan: Reinforcement learning aware simulation-to-real," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 157–11 166.

[11] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai, "Retinagan: An object-aware approach to sim-to-real transfer," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 920–10 926.

[12] A. Ganapathi, P. Florence, J. Varley, K. Burns, K. Goldberg, and A. Zeng, "Implicit kinematic policies: Unifying joint and cartesian action spaces in end-to-end robot learning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2656–2662.

[13] M. Kerzel, P. Allgeuer, E. Strahl, N. Frick, J.-G. Habekost, M. Eppe, and S. Wermter, "Nicol: A neuro-inspired collaborative semi-humanoid robot that bridges social interaction and reliable manipulation," *IEEE Access*, vol. 11, pp. 123 531 – 123 542, Nov 2023.

[14] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 785–799.

[15] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3758–3765.

[16] J.-G. Habekost, E. Strahl, P. Allgeuer, M. Kerzel, and S. Wermter, "Cycleik: Neuro-inspired inverse kinematics," in *International Conference on Artificial Neural Networks*. Springer, 2023, pp. 457–470.

[17] M. Kerzel, E. Strahl, S. Magg, N. Navarro-Guerrero, S. Heinrich, and S. Wermter, "Nico—neuro-inspired companion: A developmental humanoid robot platform for multimodal interaction," in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2017, pp. 113–120.

[18] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[19] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 13–15 Nov 2017, pp. 334–343. [Online]. Available: http://proceedings.mlr.press/v78/james17a.html

[20] C. Gäde, M. Kerzel, E. Strahl, and S. Wermter, "Sim-to-real neural learning with domain randomisation for humanoid robot grasping," in *International Conference on Artificial Neural Networks*. Springer, 2022, pp. 342–354.

[21] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4243–4250.

[22] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[23] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conference on Robot Learning*. PMLR, 2022, pp. 158–168.

[24] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *arXiv preprint arXiv:2303.04137*, 2023.

[25] D. Pavllo, D. Grangier, and M. Auli, "Quaternet: A quaternion-based recurrent model for human motion," in *British Machine Vision Conference (BMVC)*, 2018.

[26] M. Kerzel and S. Wermter, "Neural end-to-end self-learning of visuomotor skills by environment interaction," in *International Conference on Artificial Neural Networks*. Springer, 2017, pp. 27–34.

[27] M. Kerzel, J. Spisak, E. Strahl, and S. Wermter, "Neuro-genetic visuomotor architecture for robotic grasping," in *International Conference on Artificial Neural Networks*. Springer, 2020, pp. 533–545.

[28] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013, www.coppeliarobotics.com.

[29] S. James, M. Freese, and A. J. Davison, "Pyrep: Bringing v-rep to deep robot learning," *arXiv preprint arXiv:1906.11176*, 2019.

[30] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.

[31] N. E. Toklu, T. Atkinson, V. Micka, P. Liskowski, and R. K. Srivastava, "Evotorch: Scalable evolutionary computation in python," *arXiv preprint arXiv:2302.12600*, 2023.

[32] S. Zhong, T. Power, A. Gupta, and M. Peter, "PyTorch Kinematics," 3 2023.

[33] K. Perlin, "An image synthesizer," *ACM Siggraph Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.

[34] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[36] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[37] D. Q. Huynh, "Metrics for 3d rotations: Comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, pp. 155–164, 2009.

[38] B. Ames, J. Morgan, and G. Konidaris, "Ikflow: Generating diverse inverse kinematics solutions," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7177–7184, 2022.