In: *Proceedings of the 31st International Conference on Artificial Neural Networks (ICANN 2022)*, LNCS, Vol 13529, pp. 342-354, Bristol, UK, Sep 6–9, 2022

Sim-to-Real Neural Learning with Domain Randomisation for Humanoid Robot Grasping*

Connor Gäde ✉, Matthias Kerzel, Erik Strahl, and Stefan Wermter

Knowledge Technology, Department of Informatics, University of Hamburg,
Vogt-Koelln-Str. 30, 22527 Hamburg, Germany
{connor.gaede, matthias.kerzel, erik.strahl,
stefan.wermter}@uni-hamburg.de
http://www.knowledge-technology.info

Abstract. Collecting large amounts of training data with a real robot to learn visuomotor abilities is time-consuming and limited by expensive robotic hardware. Simulators provide a safe, distributable way to collect data, but due to discrepancies between simulation and reality, learned strategies often do not transfer to the real world. This paper examines whether domain randomisation can increase the real-world performance of a model trained entirely in simulation without additional fine-tuning. We replicate a reach-to-grasp experiment with the NICO humanoid robot in simulation and develop a method to autonomously create training data for a supervised learning approach with an end-to-end convolutional neural architecture. We compare model performance and real-world transferability for different amounts of data and randomisation conditions. Our results show that domain randomisation improves the transferability of a model and can mitigate negative effects of overfitting.

Keywords: Sim-to-real transfer · Domain randomisation · Humanoid robot grasping · Deep learning.

1 Introduction

A humanoid robot assisting humans in a complex environment needs a set of visuomotor abilities, such as grasping, to properly manipulate and interact with its surroundings. Learning these skills requires the collection of large amounts of training data. Collecting data with a real robot is time-consuming and can wear out or damage the expensive hardware [10,13]. Parallelising this process to speed it up requires multiple copies of the same robot [10]. Therefore, it is often beneficial to use a simulator, which allows for easy distribution and avoids safety concerns. However, simulated environments are not a fully accurate representation of the real robot and its environment, often making additional fine-tuning necessary to transfer a policy to the real world [1]. A way to reduce this *reality gap* is domain randomisation [14]. By altering aspects of the simulation such as

* The authors gratefully acknowledge support from the German Research Foundation DFG for the projects CML TRR169, LeCAREbot and IDEAS.

dynamics and visual features, learned strategies become more robust to changes in the environment, allowing an easier transfer to the real world.

We examine how domain randomisation affects the transferability of a grasping approach from simulation to a real-world setup. We develop a simulated approach to autonomously generate data for end-to-end training of a deep convolutional architecture, replicating a real-world reach-to-grasp experiment [7] with the NICO (Neuro-Inspired COmpanion) humanoid robot [6], and create three datasets with varying degrees of randomisation which we compare to a real-world dataset. First, we test whether our simulated data is suitable to solve the task. We show that models trained with our simulated data can reach a target object with similar accuracy in simulation as a model trained with real data could in the real world. We then evaluate whether a model trained on simulated data can solve the task in the real world and if domain randomisation affects it. For this, we compare the performance of models trained on canonical simulation data to ones trained with either randomised colours or camera angles when evaluated with real-world data. Our randomised samples reach a better real-world performance than the unaltered ones. We discover that an increasing amount of canonical training data leads to overfitting, which hurts the transferability of the model. Domain randomisation can mitigate this effect.

2 Related Work

2.1 Robot Grasping

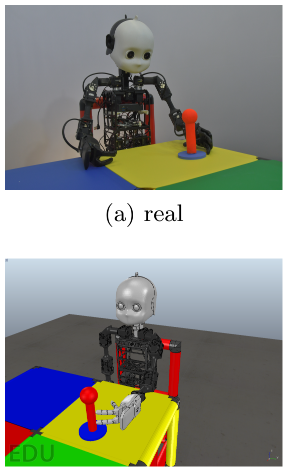
Developing robotic grasping and manipulation agents is an important challenge in research [9]. Many approaches are trying to solve this task. Pinto and Gupta [13] developed a multistage learning algorithm to train a Convolutional Neural Network (CNN) to predict the probability of a grasp at a given position and angle. Each resulting model was used to collect additional data for subsequent training iterations. Levine et al. [10] used a CNN architecture to predict the likelihood of a given image and motion to produce a successful grasp. The robot could then choose the path with the highest predicted probability. In the method proposed by Kerzel and Wermter [7], a CNN is used to predict the motor positions to reach an object in a given image. A NICO robot produced training data autonomously by placing an object on a table and recording the motor values of the arm as well as an image of the object. Our work expands this approach by Kerzel and Wermter by replicating their experiment in a simulated environment and evaluating its transferability to the real world.

2.2 Domain Randomisation for Physical Manipulation

Domain randomisation has been shown to improve the transferability of physical manipulation tasks. James et al. [3] produced training data in simulation by calculating a set of trajectories with inverse kinematics to have a robotic arm pick up a cube and put it into a basket. Images of the scene and motor velocities were recorded to train a neural model to predict velocities from images. They showed that by randomising colours, textures, light sources, and object sizes as well as introducing additional clutter, they could significantly improve the performance of the transferred model in the real environment. Similarly, Matas et al. [11] utilised domain randomisation to train a set of cloth manipulation tasks entirely on a simulated robotic arm and transfer it to a real-world setup. They altered textures, lighting, and camera orientation as well as the location and size of objects and starting position of the arm. These results show, that domain randomisation can greatly increase the transferability of a model.

3 Approach

To analyse the real-world applicability of a strategy learned in simulation, we recreate a reach-to-grasp experiment with the NICO humanoid robot in a simulated environment. We develop an approach to autonomously collect training data with a simulated NICO robot placing the target object at random positions on a table, which we use to create three datasets. One of them serves as a canonical baseline, without domain randomisation, while the others have either the colours or the camera angle altered for each image. As a real-world comparison and test of transferability, we use a fourth dataset, collected with a physical NICO robot. We implement a deep convolutional architecture and optimise its hyperparameters for each dataset under the same conditions to ensure comparability. The resulting models are first evaluated within their domain to see how the simulated and real data compare to each other. We then evaluate the sim-to-real transfer through their performance on our real-world data.



(a) real

(b) simulation

Fig. 1. Real and simulated NICO robot seated at a table with the target object on top.

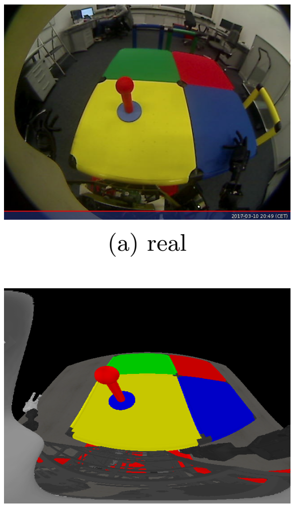
3.1 Experimental Setup

We examine the same reach-to-grasp task as presented by Kerzel and Wermter [7]. A NICO humanoid robot has to grab a cylindrical object that is randomly placed on the table in front of it by performing a side grasp with its left arm. We also use a NICO humanoid robot in our experiments. Both of its arms offer four degrees of freedom to control the shoulder and elbow angles. Each arm also has a three-fingered SeedRobotics RH4D hand attached with two wrist motors to adjust the hand's rotation as well as another two actuators to open and close the fingers. The head of the robot can be positioned with another two motors and contains two cameras within its eye sockets.

We replicate the experimental setup within the CoppeliaSim simulator[1] with a simulated NICO robot (see Fig. 1). Both the environment and the robot are controlled using the PyRep [4] library, which allows manipulation of the simulation with the Python programming language. The free, open-source software API[2] of the NICO robot provides an integrated PyRep mode, which allows for the same controls as the real robot.

3.2 Dataset Recording

With our simulated setup, we generate three datasets of 2000 samples each. One set maintains a canonical visual representation for all trials, whereas the others are randomised for each sample. The canonical images serve as a baseline to analyse the benefits of domain randomisation. Additionally, we use a fourth dataset consisting of 1100 samples collected with the real NICO robot. The purpose of this set is to test the transferability of our approach. Each sample consists of an image of the target object from the robot's perspective (see Fig. 2), as well as the associated angles of each of the six motors of the left arm.


(a) real

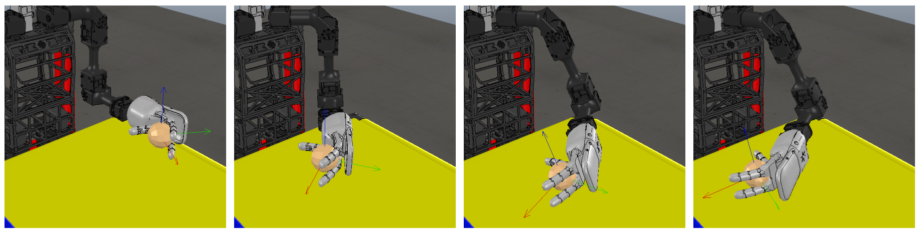
(b) simulation

Fig. 2. Sample image taken with the right eye camera of a real and a simulated NICO robot.

The real-world dataset was collected with the same method as proposed by Kerzel and Wermter [7]. Grasping can be viewed as the inversion of putting an object onto the table. Accordingly, training samples are produced by the robot placing the target object at a random position on the table and memorizing the motor angles of the arm which were used to reach that position. After removing the arm, a picture of the target object on the table is taken with the robot's eye cameras. The object is then moved to a new location to repeat the process. An experimenter initially demonstrates reachable positions to the robot by manually moving its arm over the table. The robot memorises the motor angles, which it then randomly reproduces to generate training data autonomously.

In our simulated approach, rather than demonstrating valid poses to the robot, we define ranges for each actuator to randomly generate motor configurations, such that $-75.38° \leq l\_shoulder\_z \leq -9.45°$, $-39.69° \leq l\_shoulder\_y \leq 20.35°$, $-3.91° \leq l\_arm\_x \leq 36.79°$, $35.30° \leq l\_elbow\_y \leq 102.90°$, $0.0° \leq$

[1] https://coppeliarobotics.com/
[2] https://github.com/knowledgetechnologyuhh/NICO-software



(a) $\epsilon = 0$ (b) $\epsilon = 15^\circ$ (c) $\epsilon = 30^\circ$ (d) $\epsilon > 45^\circ$

Fig. 3. Increasing hand orientation difference ϵ between the baseline (a) and the respective pose. Poses with $\epsilon > 45^\circ$ are rejected as the hand is rotated too much (d).

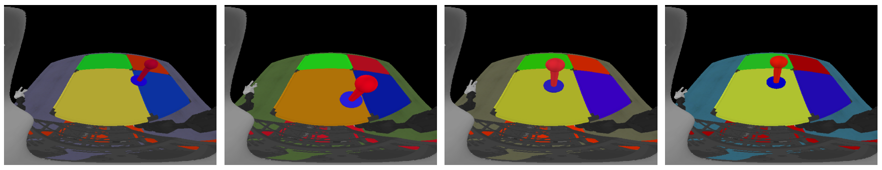
$l\_wrist\_z \leq 22.44^\circ$ and $-50.0^\circ \leq l\_wrist\_x \leq -8.84^\circ$. These limits were obtained from the minimum and maximum values recorded on the real robot.

Additionally, we ensure with a set of constraints that the generated motor values result in a valid side grasping pose. Before executing a pose, we calculate its forward kinematics with the kinematic model provided by *gaikpy* [5] to obtain the position and orientation of the hand. First, we define positional boundaries around the size of an A4 sheet of paper, such that $0.2037 \leq x \leq 0.4231$, $-0.2060 \leq y \leq 0.1454$ and $0.5773 \leq z \leq 0.6674$, to ensure that the hand is within the target area on the table at the correct height to grasp the object. After that, we confirm whether the orientation approximates a valid side grasp. For this, we define a baseline pose in which the arm is forming an L-shape and the palm faces right towards the centre of the table (see Fig. 3a). Only poses with a total angle difference of less than 45° between their end-effector orientation and our baseline are accepted. This angle is calculated between the rotation axes of the quaternion representation of both orientations. Thus, it measures the combined rotation along all three dimensions. As depicted in Fig. 3, the constraint is chosen such that it still accepts side grasps towards the right side of the robot, while poses with a strongly rotated hand are rejected.

Once these constraints are met, the simulated robot executes the pose. Rather than physically moving the object across the table, it is directly placed at the final location of the hand. The angles of the arm motor are then saved, and the robot returns to its default pose, leaving the object in the robot's view to take an image with its simulated eye cameras.

3.3 Domain Randomisation

To increase robustness and allow transferability to the real world, we randomise visual features of the scene for two of our simulated datasets. Due to changing lighting conditions in a real environment, the colours in the simulation do not match perfectly. This difference could provide difficulties for a trained model. Randomising the colours of the scene should make the resulting model more robust to changes in colour and lighting. Therefore, in our first randomised dataset,


(a) randomised colours

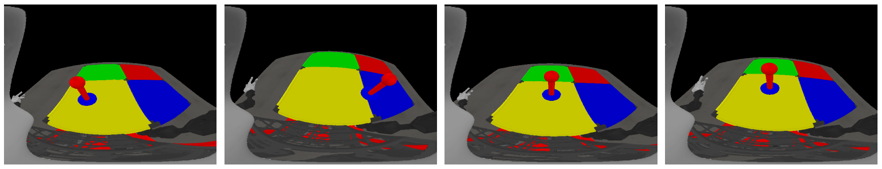

(b) randomised camera angles

Fig. 4. Sample images from our randomised colour and camera angle datasets

we alter the colours of the chair and table elements, the target object, and the floor for each recorded sample (see Fig. 4a). Colours within CoppeliaSim are defined as an $[r, g, b]$ triple, with each value having a range of $[0, 1]$. We randomise these colours by sampling each channel from a normal distribution with a mean corresponding to the canonical colour values and a standard deviation of 0.1.

As the robot is not completely fixed to the table and the head angle is influenced by its motion, the position of the camera and thus the distortion of the image does not align perfectly between simulation and real setup. By changing the angle of the camera for each sample, the model predictions should be less dependent on the accurate camera perspective and more on the position of the object relative to the table. Previous works have shown that randomisation of the camera position was crucial to successfully transfer a task into the real world [3,11]. Therefore, in the second randomised dataset, we alter the angles of head motors and thus the camera angle for each recorded sample (see Fig. 4b). By default, our data collection positions the *head_y* motor, which rotates around the y-axis (pitch), at $55°$ and leaves the *head_z* motor, which rotates around the z-axis (yaw), at $0°$ before taking an image. Our randomisation instead chooses a pitch between $40°$ and $60°$ and a yaw of up to $10°$ in either direction.

3.4 Network Architecture

We use a Convolutional Neural Network to predict motor positions for a given image (see Fig. 5), similar to the one proposed by Kerzel and Wermter [7]. Our models are implemented using the Pytorch library [12]. As input, the model is given an 80×60 RBG image, which was downsampled from a 320×240 crop of the original sample image. Our output layer consists of 6 units to predict the motor configuration for a given image. The hidden layers are comprised of two convolutional layers with rectified linear activation and a 3×3 kernel, followed by two dense layers with hyperbolic tangent activation. To determine the number of

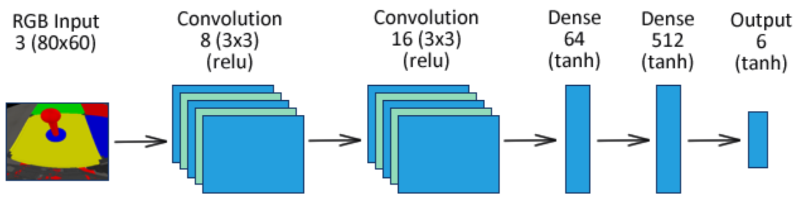


Fig. 5. Optimised model architecture for the canonical dataset

convolutional channels and linear units, as well as training epochs, we conduct a hyperparameter optimisation for each dataset using Hyperopt [2] (see Sect. 4.2). Each model is trained using mean squared error as our loss function and the Adam optimiser [8] with a default learning rate of 0.001 as well as an additional L2 penalty of 0.001 to prevent overfitting.

4 Experiments and Results

First, we optimise the hyperparameters of our deep convolutional architecture for each of our datasets and evaluate them separately. The canonical set serves as a baseline to observe if either of our randomisations affects the performance. Additionally, we analyse the influence of the dataset size on the performance. To evaluate the transferability of our models, we train each architecture with its respective dataset while using the real-world data as a test set.

4.1 Metrics

To determine the capabilities of our trained models, we define some additional metrics. We measure the distance between the prediction and the target object, by computing the forward kinematics of both the predicted and target motor configuration and calculating the Euclidean distance between the respective end-effector positions. Additionally, we verify whether our models generate valid side grasping poses by utilising the same constraints we defined for our data generation (see Sect. 3.2). We calculate the percentage of test cases where the predicted pose is within the height boundaries and deviates less than a total of $45°$ in orientation from our baseline pose.

4.2 Hyperparameter Optimisation

We conduct a hyperparameter optimisation of 100 trials for each model, choosing epochs in $\{10 \cdot n \in \mathbb{N} \mid n \leq 12\}$, convolutional filters in $\{2^n, n \in \mathbb{N} \mid 3 \leq n \leq 7\}$ for the first and $n \leq 6$ for the second layer, as well as $\{2^n, n \in \mathbb{N} \mid 6 \leq n \leq 9\}$ units for both dense layers. The models are evaluated on 1000 samples of the respective dataset with 5-fold cross-validation. The best model configurations and their respective test loss are listed in Table 1.

Table 1. Final model parameters and test performance for each examined dataset

Data	Conv		Dense		Epochs	Loss ($\mu \pm \sigma$)
canonical	8	16	64	512	110	$7.3940 \times 10^{-4} \pm 6.0177 \times 10^{-5}$
randomised colour	64	32	64	256	110	$7.1711 \times 10^{-4} \pm 5.2067 \times 10^{-5}$
randomised angles	8	8	64	512	100	$8.8976 \times 10^{-4} \pm 5.7630 \times 10^{-5}$
real	128	32	128	128	90	$5.2640 \times 10^{-4} \pm 7.8630 \times 10^{-5}$

Notably, the architectures for the canonical and randomised-angle dataset feature a minimal number of convolutional filters, with 8 on the first as well as 16 and 8 on the second layer, whereas the randomised-colour and real dataset feature 64 and 128 filters on the first and 32 on the second layer. This could reflect the higher visual complexity of the latter two sets. For the dense units, the simulated datasets result in similar architectures with 64 units on the first and 512 on the second layer for the canonical and randomised-angle datasets or 256 for the colour dataset. The real-world dataset differs from this with 128 units on both layers. The number of epochs is also similar, with 110 for the canonical and colour datasets, 100 for randomised angles, and 90 for the real data.

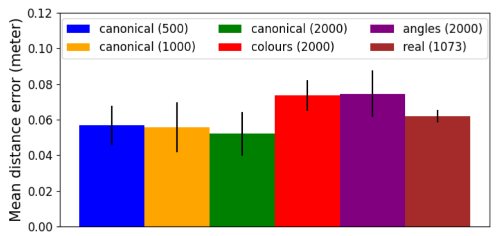
4.3 Dataset Performance

Before analysing the transferability of our datasets, we establish how they perform on their own. We train each optimised architecture with the full 2000 samples of the respective dataset for the number of epochs determined by our hyperparameter optimisation and evaluate them with 5-fold cross-validation. We analyse the influence of the dataset size by training reduced versions of the canonical dataset with 500 and 1000 samples in the same manner. To maintain consistency throughout all of our datasets, we apply our orientational constraint of 45° (see Sect. 3.2) to the real-world data, reducing it to 1073 samples.

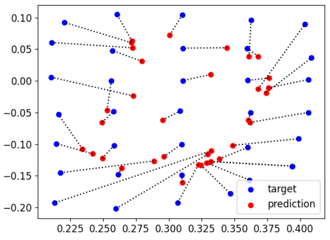
Table 2. Test loss and percentage of valid generated poses for each examined dataset

Dataset	# Samples	Loss ($\mu \pm \sigma$)	Valid Poses ($\mu \pm \sigma$)
canonical	500	$7.7718 \times 10^{-4} \pm 9.3450 \times 10^{-5}$	100.0% ± 0.00
	1000	$8.2458 \times 10^{-4} \pm 1.7685 \times 10^{-4}$	96.4% ± 7.20
	2000	$7.6861 \times 10^{-4} \pm 1.0146 \times 10^{-4}$	100.0% ± 0.00
colours	2000	$1.2258 \times 10^{-3} \pm 2.7900 \times 10^{-4}$	100.0% ± 0.00
angles	2000	$1.0962 \times 10^{-3} \pm 2.5002 \times 10^{-4}$	93.8% ± 8.23
real	1073	$5.5526 \times 10^{-4} \pm 4.3983 \times 10^{-5}$	100.0% ± 0.00

Table 2 shows the training results for all examined datasets. The model trained on real-world samples reaches the best overall performance, with an average loss of 5.5526×10^{-4}. The results of our simulated datasets span from 7.6861×10^{-4} loss for the full canonical dataset, to 1.2258×10^{-3} for the randomised-colour set. Our full canonical dataset, as well as its subset of 500 samples, both



(a)



(b)

Fig. 6. Mean distance error after evaluating each dataset with 5-fold cross-validation (a) and a selection of actual and predicted positions for the best-performing model (b)

perform slightly better than their 1000-sample counterpart, although still within each other's standard deviations, with the full dataset achieving the better score but also a higher standard deviation than the smaller subset. The randomised datasets achieve the worst performances. However, as indicated by the higher standard deviations, their best models are closer to the canonical ones. This difference could be due to the higher difficulty of learning more randomised data. Yet, during hyperparameter optimisation, the randomised-colour data performed better than the other simulated datasets.

All models generate valid poses as defined in Sect. 4.1 for more than 90% of their test cases. Most architectures achieve 100%, barring the ones trained on 1000 canonical samples and randomised camera angles (see Table 2).

By looking at the average distance between the target object and end-effector position, as depicted in Fig. 6a, we can see that the loss does not fully represent the ability to reach the target. As the reached position depends on the combination of all angles, the errors of different motors seem to be able to compensate for each other. All models trained with canonical data reach the object at a closer average distance than the real robot. The smallest subset reaches the target pose around $5mm$ closer at $5.68cm$ than the real data with $6.19cm$. An increase of samples improves this to $5.59cm$ for the 1000 sample subset and $5.21cm$ for the full set, though they are also more inconsistent between the different runs at $11.2mm$, $14.1mm$, and $12.3mm$ standard deviation respectively. The randomised sets perform about $1cm$ worse than the real-world samples with an average distance of $7.35cm$ for randomised colours and $7.45cm$ for angles, but also have a higher standard deviation of $13.1mm$ and $8.5mm$ respectively.

The distance error is not evenly distributed across the samples. As seen in Fig. 6b, all predictions gravitate towards the centre of the workspace. For targets closer to the edge of the reachable space, the inaccuracy of the prediction increases. One explanation for this could be that the edges of the reachable space contain fewer samples than the centre. This would coincide with the findings by

Kerzel et al. [5] who showed that predictions on the fringes of a workspace converge towards its centre unless the data is sampled from a larger area.

While neither of the models are accurate enough to reliably grasp the target object throughout the entire workspace, we can see that our simulated data can reach similar or better results than the real-world data.

4.4 Sim-to-Real Transfer

After confirming that our simulated datasets can produce models with comparable performance to the real-world data, we analyse how well their success transfers into the real-world application. We train each optimised architecture on the full 2000 samples of the respective simulated dataset for the optimised number of epochs while using our 1073 real-world samples as test set. Each model is evaluated five times to account for deviations between individual runs. As in Sect. 4.3, we also examine reduced subsets at 500 and 1000 samples of our datasets to analyse the influence of the amount of data on the result.

Table 3. Test loss and percentage of valid generated side grasping poses for each examined dataset when tested on real-world data.

Dataset	**# Samples**	**Loss ($\mu \pm \sigma$)**	**Valid Poses ($\mu \pm \sigma$)**
canonical	500	$\mathbf{2.6830 \times 10^{-3} \pm 2.7898 \times 10^{-4}}$	**91.50% ± 10.16**
	1000	$3.6398 \times 10^{-3} \pm 1.1927 \times 10^{-3}$	50.93% ± 32.72
	2000	$8.6718 \times 10^{-3} \pm 9.7359 \times 10^{-3}$	59.81% ± 37.95
colours	500	$\mathbf{2.1284 \times 10^{-3} \pm 2.3500 \times 10^{-4}}$	**100.00% ± 0.00**
	1000	$2.6896 \times 10^{-3} \pm 2.9349 \times 10^{-4}$	94.63% ± 10.74
	2000	$3.0234 \times 10^{-3} \pm 1.6071 \times 10^{-3}$	87.42% ± 25.14
angles	500	$\mathbf{2.2035 \times 10^{-3} \pm 2.1958 \times 10^{-4}}$	**100.00% ± 0.00**
	1000	$2.6528 \times 10^{-3} \pm 1.8045 \times 10^{-4}$	92.59% ± 10.89
	2000	$2.5690 \times 10^{-3} \pm 2.7915 \times 10^{-4}$	100.00% ± 0.00
real	1073	$\mathbf{5.5526 \times 10^{-4} \pm 4.3983 \times 10^{-5}}$	**100.00% ± 0.00**

Table 3 shows that the test losses of our models increase by an order of magnitude for most datasets when evaluated on real-world data. Our smallest subset of canonical data reaches an average loss of 2.6830×10^{-3}. Increasing the amount of data seems to negatively affect the transferability of the model, as the average loss increases with larger datasets up to 8.6718×10^{-3} for the full canonical set. The standard deviation also becomes incrementally worse. This could be caused by the model overfitting on aspects specific to the simulation which it does not discover when trained with fewer, less redundant samples. Domain randomisation seems to mitigate this effect. Our randomised datasets achieve similar or better results as the small canonical dataset between 2.1284×10^{-3} and 2.6896×10^{-3}, barring the full colour set, which performs slightly worse due to an outlier, as indicated by the higher standard deviation.

The rate of valid generated poses seems to support our overfitting hypothesis. While the smallest subset of 500 canonical samples reaches 91.50%, the bigger datasets only generate an average of less than 60% valid grasp poses for the real-world test samples. The full randomised-colour dataset performs slightly worse than the smallest canonical subset with 87.42% valid poses, whereas the other randomised sets outperform it, with the three best ones reaching 100%.

A similar pattern emerges in the resulting distances between the end-effector of generated poses and the target position. Models trained on the smallest subset of canonical data reach a mean distance of 12.54cm, further than 6cm away from the target than within its own domain or the model trained on the real dataset at 6.19cm. The distance increases to 17.34cm $\pm$ 4.96cm with 1000 samples and 21.91cm $\pm$ 13.24cm for 2000 samples. Models trained on our full randomised-colour dataset achieve a similar result to the small canonical subset, approaching the target at an average 12.60cm with a higher standard deviation of 5.5cm, whereas the other randomised sets reach better performances, coming as close as 9.84cm average distance for the smallest colour subset.

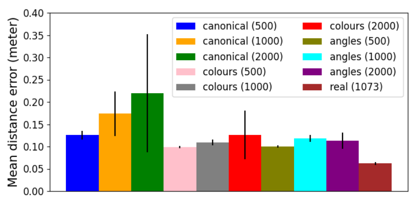


Fig. 7. Mean distance between the real-world target and the predicted hand position for each model. Each dataset was evaluated five times.

Overall, our randomised datasets show a higher transferability than the canonical data. Larger amounts of canonical data result in lower performance due to overfitting, whereas domain randomisation mitigates this effect.

5 Conclusion

We explored the effects of domain randomisation on the transferability of a robotic reach-to-grasp task trained entirely in simulation without additional fine-tuning. Our approach recreated a real-world experiment conducted by Kerzel and Wermter [7] with the NICO robot in a simulated environment. We developed a data collection method similar to the real-world approach, which utilised the advantages of a simulation to create three datasets of 2000 samples each. A fourth dataset of 1100 samples collected in the same way as the original experiment served as a real-world comparison. Each sample consisted of an image of the target object on the table from the robot's perspective and the angles of the six motors of the left arm required to reach it.

Our simulated data reached a similar or better performance within simulation than the real-world dataset. However, when evaluating the models trained on simulated data with real-world samples, the sim-to-real gap becomes apparent through lower accuracy. Contrary to the general expectation, simply adding more training data decreased the transferability of the resulting models. We demonstrated, however, that randomising individual visual features of the simulation mitigates this loss of transferable behaviour for larger amounts of simulated training data and improves the real-world performance of a transferred model.

In future work, we could explore randomising multiple features at once and improve the accuracy of the model to deploy it on a physical robot.

References

1. van Baar, J., Sullivan, A., Cordorel, R., Jha, D., Romeres, D., Nikovski, D.: Sim-to-real transfer learning using robustified controllers in robotic tasks involving complex dynamics. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 6001–6007. IEEE (2019)
2. Bergstra, J., Yamins, D., Cox, D.D., et al.: Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In: Proceedings of the 12th Python in science conference. vol. 13, p. 20. Citeseer (2013)
3. James, S., Davison, A.J., Johns, E.: Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. In: Conference on Robot Learning. pp. 334–343. PMLR (2017)
4. James, S., Freese, M., Davison, A.J.: Pyrep: Bringing v-rep to deep robot learning. arXiv preprint arXiv:1906.11176 (2019)
5. Kerzel, M., Spisak, J., Strahl, E., Wermter, S.: Neuro-genetic visuomotor architecture for robotic grasping. In: International Conference on Artificial Neural Networks. pp. 533–545. Springer (2020)
6. Kerzel, M., Strahl, E., Magg, S., Navarro-Guerrero, N., Heinrich, S., Wermter, S.: 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). pp. 113–120. IEEE (2017)
7. Kerzel, M., Wermter, S.: Neural end-to-end self-learning of visuomotor skills by environment interaction. In: International Conference on Artificial Neural Networks. pp. 27–34. Springer (2017)
8. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
9. Kleeberger, K., Bormann, R., Kraus, W., Huber, M.F.: A survey on learning-based robotic grasping. Current Robotics Reports pp. 1–11 (2020)
10. Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., Quillen, D.: Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. The International Journal of Robotics Research **37**(4-5), 421–436 (2018)
11. Matas, J., James, S., Davison, A.J.: Sim-to-real reinforcement learning for deformable object manipulation. In: Conference on Robot Learning. pp. 734–743. PMLR (2018)
12. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
13. Pinto, L., Gupta, A.: Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In: 2016 IEEE international conference on robotics and automation (ICRA). pp. 3406–3413. IEEE (2016)
14. Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P.: 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). pp. 23–30 (2017). https://doi.org/10.1109/IROS.2017.8202133