Neuro-Genetic Visuomotor Architecture for Robotic Grasping*

Matthias Kerzel[1], Josua Spisak[1], Erik Strahl[1], and Stefan Wermter[1]

Knowledge Technology, Department of Informatics, University of Hamburg, Germany
kerzel / 6spisak / strahl / wermter @informatik.uni-hamburg.de
http://www.knowledge-technology.info

Abstract. We present a novel, hybrid neuro-genetic visuomotor architecture for object grasping on a humanoid robot. The approach combines the state-of-the-art object detector RetinaNet, a neural network-based coordinate transformation and a genetic-algorithm-based inverse kinematics solver. We claim that a hybrid neural architecture can utilise the advantages of neural and genetic approaches: while the neural components accurately locate objects in the robot's three-dimensional reference frame, the genetic algorithm allows reliable motor control for the humanoid, despite its complex kinematics. The modular design enables independent training and evaluation of the components. We show that the additive error of the coordinate transformation and inverse kinematics solver is appropriate for a robotic grasping task. We additionally contribute a novel spatial-oversampling approach for training the neural coordinate transformation that overcomes the known issue of neural networks with extrapolation beyond training data and the extension of the genetic inverse kinematics solver with numerical fine-tuning. The grasping approach was realised and evaluated on the humanoid robot platform NICO in a simulation environment.

Keywords: Bio-inspired visuomotor learning · neuro-robotic models · genetic algorithms · hybrid neural networks.

1 INTRODUCTION

We present a novel neuro-genetic architecture for robotic grasping with a humanoid platform. The architecture, depicted in Figure 1, leverages the strength of two different bio-inspired approaches (neural networks and genetic algorithms) in a modular architecture that allows developing, training and evaluating each module independently. The architecture addresses the challenge that neural end-to-end approaches for learning a direct mapping from visual input to motor commands become challenging to analyse with increasing task complexity. In case of a failed grasp, there are competing error hypothesis: 1) the object was not recognised or located incorrectly in the visual input; 2) the object's location in

* The authors gratefully acknowledge partial support from the German Research Foundation DFG under project CML (TRR 169)

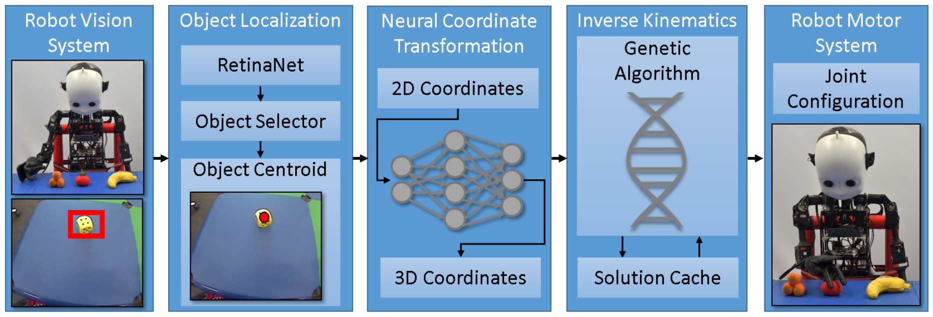

Fig. 1. Hybrid neuro-genetic visuomotor architecture.

the two-dimensional visual input is not correctly transformed into the robot's three-dimensional reference frame and 3) the inverse kinematics computation failed to compute the suitable joint configuration to reach for the object. Our proposed neuro-genetic architecture complements previous work on end-to-end visuomotor learning [10, 8] by allowing to analyse and monitor each of these steps separately and thus affording explainability and transparency. The proposed architecture consists of three modules: two neural modules localise a graspable object in the robot's field of view and transform the position of the object in the robot's two-dimensional camera image into three-dimensional coordinates. A third module based on a genetic algorithm computes a joint configuration for grasping the object.

The *object localisation* uses the neural object detection network RetinaNet [14] with a ResNet50 backbone pre-trained on ImageNet. The architecture is a fast and reliable single-stage image detector that achieves state-of-the-art results on the COCO dataset. RetinaNet outputs bounding boxes of objects. In a scene with multiple objects, an object selector submodule selects the object to be grasped based on the object classifications.

The neural *coordinate transformation* module transforms the position of the centroid of the target object in the robot's camera image to a 3d coordinate in the robot's reference frame. The architecture is based on a multi-layer perceptron. We introduce a novel spatial-oversampling method to improve transformation accuracy by compensating known issues of neural networks with extrapolation beyond known data points [20].

The *inverse kinematics* (IK) solver transforms the 3d-coordinates into a joint configuration using a combination of a genetic algorithm and a numerical approach. Especially for humanoid robots, genetic IK solvers have advantages over classical approaches. Humanoid robots, do not have the classical 6-DOF-design of industrial robots but mimic the joint limits of humans. Genetik IK solvers can handle these constraints better than classical approaches. The constraints lead to unreachable positions, where a genetic IK can compute a best-possible solution [19] which we further optimise with gradient-based Sequential Least SQuares Programming. The novel architecture offers separate and well-controllable training and evaluation options: The object detector is trained on a large, existing image dataset while the coordinate transformation is trained with data from a virtual environment. We implemented and evaluated the approach on the humanoid robot NICO [9] in the V-REP[1] simulation environment, see Figure 4.

Our main contributions are 1) A neuro-genetic visuomotor architecture that combines the advantages of neural and genetic approaches. 2) A novel spatial-oversampling method that overcomes inaccuracies in neural networks for spatial visuomotor tasks caused by extrapolation beyond known data points.

2 BACKGROUND AND RELATED WORK

2.1 Visuomotor Frameworks for Object Grasping

Visuomotor frameworks for object grasping solve three subtasks: they locate an object in the sensory input of the robot, transform this location to the reference frame of the robot and solve the inverse kinematic calculation to enable the robot to reach for the object. Classical frameworks consist of independent modules, often applying analytical solutions to coordinate transformations, which require accurate localisation of the robot and the object in a common reference frame, e.g., [12]. Neural approaches utilise different learning strategies ranging from supervised end-to-end learning [13, 10] to deep reinforcement learning, see [16] for an overview. Many Approaches take inspiration from biology and human development, e.g., [7]. However, these systems are difficult to analyse [8]. In contrast to approaches that employ an evolutionary algorithm to optimise a visuomotor neural network [18], our system uses a neural network and an evolutionary algorithm in separate modules, allowing easier analysis of each module.

2.2 Neural Object Detectors

Object detectors locate and classify objects in an image. Two-stage architectures [5] use one network to generate region proposals and a second network to classify these proposals. To reduce redundant computations, single-stage models realise proposal generation and classification with a shared network [17], called backbone. Retinanet [14] uses a deep residual network (Resnet [6]) and a Feature Pyramid Network (FPN) for this purpose. The Resnet's feature maps are connected to regression and classification subnetworks that output object bounding boxes and class labels.

2.3 Inverse Kinematics Solvers

Once an object is located in the robot's reference frame, the visuomotor framework calculates an inverse kinematics solution, a joint configuration that moves the end effector of a robotic arm into a grasping pose. While analytical solutions are feasible for a low number of degrees of freedom (DoFs), numerical methods

[1] https://www.coppeliarobotics.com/

are often used to solve the inverse kinematics for industrial arms and robots. Most commonly, iterative models, like the Jacobian approach, are applied [1]. Humanoid robots with constrained joints can not reach all possible poses. In this case, iterative models tend to follow the gradient into local minima, resulting in suboptimal outcomes. Purely neural approaches can overcome this issue, but require a large amount of training data and can suffer from relatively high errors. Daya et al. [3] attributes this to the non-linearity of the robotic kinematic systems. Köker et al. [11] combine neural and genetic approaches to overcome this issue. Genetic and particle-swarm algorithms use a pool of IK solutions that are iteratively improved by random changes (mutation) and exchange of partial solutions (crossing over). These algorithms are well-suited to find the best possible solution in a given time for a humanoid robot [19].

3 METHODOLOGY

The neuro-genetic visuomotor architecture consists of three main modules, as shown in Figure 1. 1) the *Object Localisation* is realised with the neural object detector RetinaNet [14]; in case of multiple detected objects, the *Object Selector* arbitrates which object to grasp. 2) The *Neural Coordinate Transformation* generates the robot-centric 3d-coordinates based on the position of the detected object in the visual input. 3) Finally, the *Inverse Kinematics* generates a joint configuration to move the robot's hand to the 3d-coordinates and into a grasping position that is then executed by the robot's motor system.

3.1 Object Localisation

RetinaNet [14] with a ResNet-50 backbone pretrained on the Imagenet dataset is used for object detection. The network takes RGB images as input and outputs bounding boxes and classifications for found objects. A Keras implementation[2] is used. In the case of multiple objects in the robot's visual input, an *Object Selector* arbitrates between objects based on the objects' class labels. From the bounding box of the selected object, the centroid is computed.

3.2 Neural Coordinate Transformation and Spatial Oversampling

The *Coordinate Transformation* is a multi-layer perceptron (MLP) that regresses from the 2d-coordinates of the object's centroid in the robot's visual input to 3d-coordinates in the robot's reference frame. Both the two input and the three output coordinates are scaled to the range of [-1, 1]. The hyperparameters of the architecture result from automated hyperparameter optimisation. The input layer is followed by three dense layers with 30 units. The ReLU activation function is used for the hidden layers and the sigmoid function for the output layer. We developed a novel spatial oversampling strategy for training the *Coordinate Transformation*. Neural networks excel at interpolating between known

[2] https://github.com/fizyr/keras-retinanet

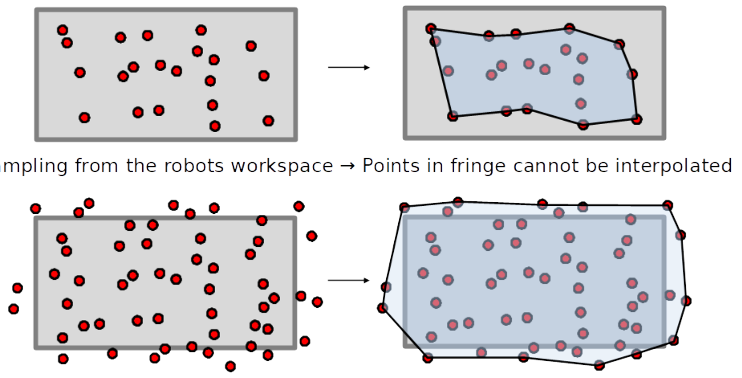
Sampling from the robots workspace → Points in fringe cannot be interpolated
Spatial oversampling from area larger than workspace resolves issue

Fig. 2. Top: When only using samples from the workspace of the robot to train a neural network for coordinate transformation, there are areas of the workspace that require extrapolation from known samples, which can cause inaccuracies. Bottom: By using training samples from beyond the robot's workspace (spatial oversampling), larger parts of the workspace can be covered by interpolation.

data points, but they can not reliably extrapolate beyond these points [20]. This issue causes problems when training a neural network with randomly sampled points from a robot's workspace: random sampling will not completely cover the outer areas of the workspace and lead to inaccuracies of the trained model. However, by extending the sampling area beyond the workspace, a larger part of the workspace can be covered, as shown in Figure 2. The module is trained independently of the object detector; its training pairs of 2d and 3d-coordinates are generated in a simulation environment where exceeding the robot's workspace can easily be realised.

3.3 Genetic Inverse Kinematics Solver

Genetic algorithms model the evolutionary selection process on a population of individuals, modelled by their chromosomes [15]. Chromosomes can be affected by mutations, some individuals can become an elite and get special treatment, and there are niches of populations, which are isolated from the rest. All these principles are implemented in the genetic inverse kinematics solver. Each chromosome represents a joint configuration of the robot's arm. In each iteration of the algorithm, individuals are ranked according to a fitness function. In our case, this measures the distance and orientation error toward the goal pose. The fittest individuals are selected; their randomly altered (mutated) form the population of the next iteration. Genetic algorithms excel at avoiding local minima but lack an effective way to further optimise into these minima. For this purpose, we use gradient-based Sequential Least SQuares Programming (SLSQP) to minimise the error on the local minima. As this is computationally expensive, we optimise only the elite of the n-best individuals of the population. SLSQP

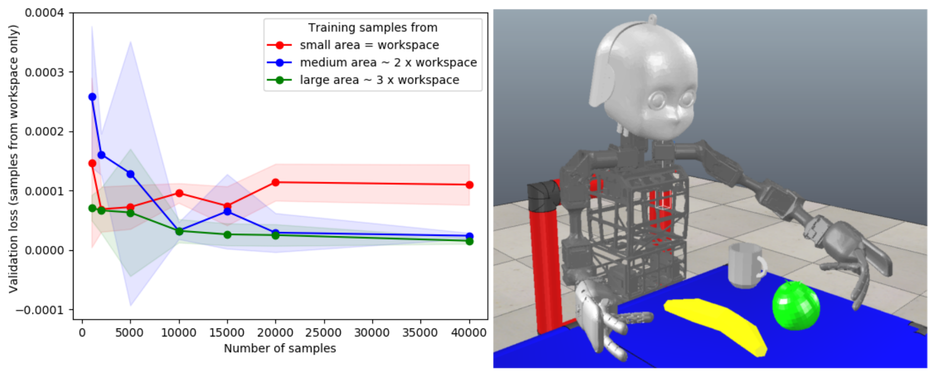

Fig. 3. Left: Validation loss computed on the actual workspace of the robot for different sampling areas and dataset sizes. Right: NICO in a virtual training environment.

tends to get lost in local minima but works very well if it gets initialised via the genetic algorithm.

To take full advantage of multiple processing cores, we use the genetic niche concept, running one evolutionary niche on every available CPU, like on an isolated island. This uses multiple cores effectively as there is minimal management overhead.

4 EXPERIMENTS AND RESULTS

The evaluation leverages one of the main advantages of the modular architecture; each module can be analysed separately. We report the evaluation of the neural *Coordinate Transformation* with a focus on the effect of the spatial oversampling strategy and the *Genetic Inverse Kinematics Solver*. For the object detector RetinaNet [14], we refer to the results published by Lin et al. and assume that objects on a non-cluttered desk can be correctly classified and localised.

4.1 Experimental Setup: NICO Robot and Simulation Environment

Our approach is realised and evaluated on a virtual version of NICO, the Neuro Inspired COmpanion [9], a child-sized humanoid. We use the V-REP environment with a physics engine shown in Figure 3, where NICO is seated at a children's desk with several objects in its 40 × 60 cm workspace. In contrast to previous work [10, 8, 4], where end-to-end grasping approaches were realised on the physical NICO by using its ability to place the training object, we chose a virtual environment as the spatial oversampling can be realised more efficiently. Moreover, our architecture allows a fully decoupled training of the vision component (trained on real-world images) from the coordinate transformer (trained in a virtual environment), avoiding the so-called sim-to-real gap. To gather the training data for the coordinate transformer, a visually salient grasping object is moved through NICO's workspace and the extended workspaces, respectively.

The object's 3d position, as well as a visual image from the perspective of the robot, is recorded. As the grasping object has a clear contrast to the background, a simple colour-based object detector is used to determine the centroid of the object in the visual image for purposes of gathering training data.

4.2 Neural Coordinate Transformation

The *Neural Coordinate Transformation* creates a link between the vision and the motoric modules by transforming the position of an object in the visual input to 3d-coordinates. Extrapolation beyond known data points can be problematic for neural networks. Preliminary experiments show that despite a good overall accuracy, the points at the fringes of the workspace are pulled towards the centre, as shown in Figure 4 To address this systematic bias, samples from outside the robot's workspace were used. While the workspace is limited to an area of 40 $\times$ 60 cm, two different spatial oversampling strategies were evaluated by sampling from wider areas of about two and three times the original workspace. Forty thousand samples were collected from each area.

Hyperparameter Optimisation Hyperparameters were optimised independently for all three sampling conditions using Hyperopt [2]. Samples from each area were split into 75% training and 25% validation data. Table 1 shows the ranges and results for the parameters: number of layers (2 or 3), optimiser (adadelta or adam with learning rates 0.01,0.001 and 0.0001), neurons per layer (20, 30 or 40), dropout (0 or 0.2), batch size (5, 10 and 15) and the number of epochs (100, 150 and 200). While the hyperparameter optimisation was performed for all three sampling areas independently, the validation loss was calculated only for samples from the robot's actual workspace area for all three networks. The resulting hyperparameters are shown in Table 1. Optimisation was performed for 100 trials for each condition. We notice that the larger areas have a lower learning rate and use three layers instead of two.

Table 1. Hyperparameter optimisation: range of parameters and results for all three area sizes. *The learning rate was only adjusted for the adam optimiser.

Hyperparameter	**Range**	**Large area**	**Medium area**	**Small area**
Batch size	5, 10, 25	10	5	15
Dropout	0, 0.2	0	0	0
Learning rate*	0.01, 0.001, 0.0001	0.001	0.001	0.01
Optimiser	adam, adadelta	adam	adam	adam
Number of layers	2, 3	3	3	2
Units per layer	20, 30, 40	30	40	40
number of epochs	100, 150, 200	200	100	200

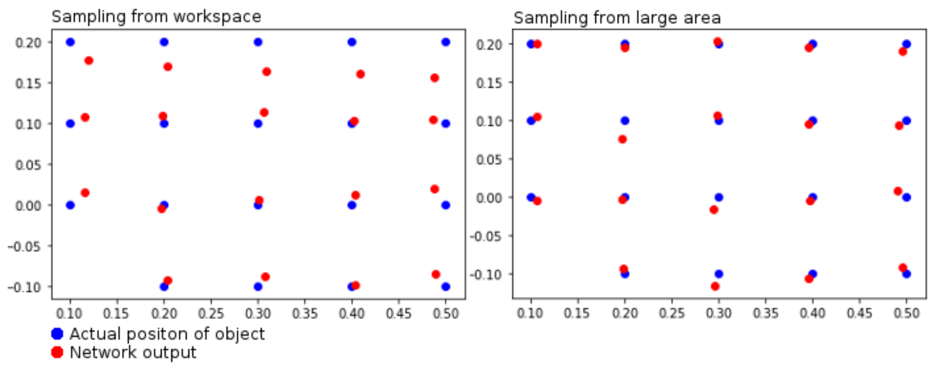

Fig. 4. Actual (blue) and computed (red) positions by the neural coordinate transformation in NICO's 60cm × 40cm workspace. Left: The best model trained with samples from the workspace. Points near the fringe converge towards the middle. Right: The best model trained with spatial oversampling from a three-times larger area.

Spatial Oversampling Comparison and Evaluation All sampling strategies were evaluated with individually optimised hyperparameters. Figure 3 shows averaged results for 1000 to 40000 samples taken from the original workspace and the two larger areas. Each experiment was repeated with ten-fold cross-validation with a split of 90% test and 10% validation data. The validation loss is computed on the actual workspace of the robot (60cm × 40cm) for all sampling areas and dataset sizes. One could assume that using only samples only from this workspace, against which the model is validated, yielded the best result. However, we observe that at 5000 samples, taking more samples from the original workspace stops to improve the result. Instead of using samples from outside the workspace (spatial oversampling) results in a lower loss. The best averaged MSE over ten trials of $1.55E - 05$ (SD $5.07E - 06$) is achieved with 40000 samples from the largest area, which is negligible for the grasping task.

4.3 Genetic IK

Based on previous work [10] we estimate the accuracy that NICO requires for grasping: we define errors in position < 10 mm and errors in orientation with a sum of < 20 degrees as a successful trial. This accuracy range is sufficient for grasping, as the robot's hand has an opening of around 40 mm. As we limited the joints on NICO for human-like movements, the genetic IK needs larger populations and larger numbers of generations, causing possible run time issues. However, good Human-Robot Interaction requires a result in a time that enables interaction; we used the options of 1s, 2.5s and 4s with the hyperparameters optimised for these time constraints. The algorithm has run on a Linux server with an Intel Xeon CPU E5-2630, v4 by 2.20GHz with a maximum use of 8 cores. On a Dell laptop with an Intel i7-9750H CPU, we see very similar running times, so the running time is representative for standard pc environments. Storing past solutions in a cache and ending the calculations after the specified accuracy is reached can further speed up the algorithm.

Table 2. Optimised hyperparameters for genetic algorithm for different time goals. TPE optimiser, 1000 trials

Time goal	**Range**	**1s goal**	**2.5s goal**	**4s goal**
Population size	4-50	4	8	4
Number of generations	4-50	4	4	16
Number of elites	1-8	2	2	2
Mutation rate	0-1	0.18	0.36	0.36
Orientation weight	0-1	0.516	0.449	0.27
Max. iterations (SLSQP)	10-1000	880	820	740

Hyperparameter optimisation The hyper-parameters (population size, number of generations, mutation rate, number of elites) were optimised to ensure a balance of run time and accuracy using a Tree Parzen Estimator (TPE) with the goal to deliver the results 1) with a position error of less than 10mm and 2) with an additive orientation error of fewer than 20 degrees (0.349 rad) and 3) and appropriate computing time. We optimised for the three variants 1, 2.5 and 4 seconds to compare the results. The calculation times are all adequate for Human-Robot Interaction, but a lower value would enhance the robot's responsiveness in a Human-Robot Interaction scenario. Table 2 shows the standard parameters for population size, number of generations, number of elites, mutation rate, orientation weight and the maximum number or SLSQP iterations. Furthermore, the orientation weight, which controls the ratio in the fitness function between position accuracy and orientation accuracy. In the results of the hyper-optimisation, we see that the population size and the number of generations are the main factors to decrease computing times; nonetheless, different strategies are possible. For the 2.5 second goal, the optimiser locked in for a high population and a low number of generations, while it went for a high number of generations strategy for the four seconds goal.

Evaluation We generated 1000 samples with the forward kinematics in the robot's workspace on the table to ensure all generated positions are reachable. With these samples, we tested our three different parameter sets for a calculation goal time of 1s, 2.5. and 4s. Furthermore, an optimisation using the SLSQP without the genetic algorithm (max of 100000 iterations). Table 3 shows the results. The more calculation time is invested for the genetic algorithm, the better the accuracy. The results for the SLSQP show that the optimiser performs poorly without the genetic algorithm.

Table 3. Results of three different hyperparamter sets tuned for running times of 1s, 2.5s, 4, an pure SLSQP optimisation and finetuned hyperparameters

Algorithm and parameter set	**ga(1s)**	**ga(2.5s)**	**ga(4s)**	**SLSQP**
Position error (m) (mean)	2.778 x 10-4	1.627 x 10-4	2.161 x 10-5	1.557 x 10-03
Orientation error (rad) (mean)	1.507 x 10-1	1.077 x 10-1	8.912 x 10-2	5.47 x 10-1
Time (s) (mean)	0.817	1.72	2.91	0.241
Error rate	0.115	0.078	0.057	0.528

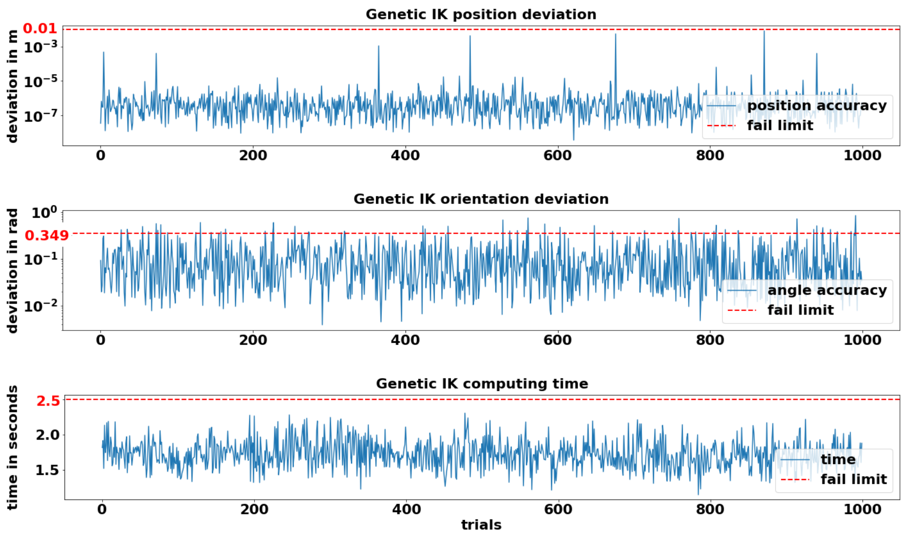

Fig. 5. Position error, orientation error and running time for 1000 random positions in the robot's workspace on the table.

We choose a compromise between speed and accuracy with the 2.5s model (average calculation time 1.72s). Figure 5 shows the position errors, orientation errors, and the calculation time for this parameter set. The red lines mark our success definitions. Figure 6 shows a visualisation of the position and orientation rotation errors depending on the position in the workspace. While the position accuracy is relatively uniform with some patches of higher error near the fringes, the orientation accuracy improves in the top right area while it decreases in the lower-left area. We attribute this tendency to the kinematics of the robot. While its hand can reach all positions in the workspace, it can not do so everywhere with the desired rotation, which increases the difficulty of finding solutions in these areas.

4.4 Discussion on the Hybrid Neuro-Genetic Approach

Comparing the results from the neural coordinate transformation and the genetic inverse kinematics solver, we see two tendencies: The neural approach's accuracy diminishes towards the fringes of the workspace. This error can, however, be mostly be compensated through spatial oversampling. In contrast, the orientation accuracy of the inverse kinematics solver changes along a diagonal axis, which we attribute to the kinematic constraints of the robot. This detailed analysis is possible because of the modular nature of the architecture. In a neural end-to-end learning approach, an alternative hypothesis, which we can rule out, would be an imbalance in the training data, with more or better samples towards the upper right region.

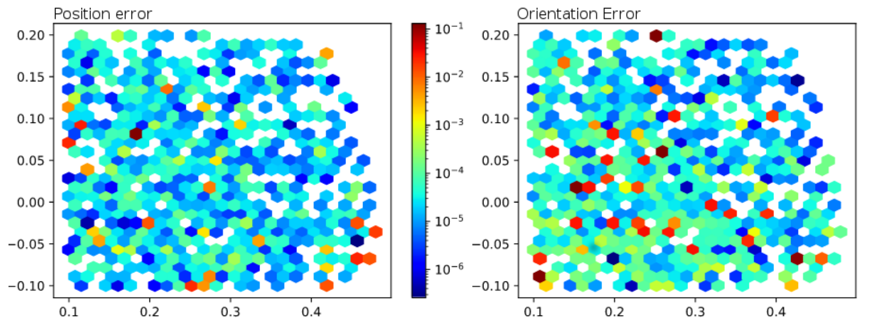

Fig. 6. Position (left) and orientation (right) error of the genetic inverse kinematics solver in relation to the position in the robot's workspace. The orientation errors are higher in the lower-left area.

5 Conclusion

Our main contribution is a neuro-genetic visuomotor architecture for robotic grasping that leverages the strength of neural networks while compensating their possible shortcomings through hybridisation with a genetic algorithm. While neural networks excel at object detection and, due to spatial oversampling, also achieve good results on coordinate transformation, a genetic inverse kinematics solver in combination with a numerical optimisation is used to address the inaccuracy of neural approaches for humanoid kinematics. The modularity of the architecture allows attributing inaccuracies to different modules and, in our case, to the robot's human-like joint angle limitation. An added advantage of the decoupled neural modules is that the object detector can use pretrained models or existing datasets, while the coordinate transformation is trained in simulation. We also contribute a novel spatial oversampling method that avoids extrapolation beyond known data points for the spatial task of coordinate transformation, thus increasing the accuracy. We evaluate the neural coordinate transformation and the genetic inverse kinematics solver in a simulation environment; in future work, we will implement the architecture on a physical robotic platform and compare its performance to related approaches. We show that our novel neuro-genetic architecture combines the advantages of the two approaches and yields an additive error that is appropriate for robotic grasping.

References

1. Aristidou, A., Lasenby, J.: Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver (2009)
2. Bergstra, J., Yamins, D., Cox, D.: Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In: 30th International Conference on Machine Learning (ICML 2013). pp. 115–123 (2013)
3. Daya, B., Khawandi, S., Akoum, M.: Applying neural network architecture for inverse kinematics problem in robotics. Journal of Software Engineering and Applications **3**(03), 230 (2010)
4. Eppe, M., Kerzel, M., Griffiths, S., Ng, H.G., Wermter, S.: Combining deep learning for visuomotor coordination with object identification to realize a high-level

interface for robot object-picking. In: IEEE-RAS International Conference on Humanoid Robots (Humanoids). pp. 612–617 (2017)

5. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 580–587 (2014)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
7. Jamone, L., Natale, L., Nori, F., Metta, G., Sandini, G.: Autonomous online learning of reaching behavior in a humanoid robot. International Journal of Humanoid Robotics **9**(03), 1250017 (2012)
8. Kerzel, M., Eppe, M., Heinrich, S., Abawi, F., Wermter, S.: Neurocognitive shared visuomotor network for end-to-end learning of object identification, localization and grasping on a humanoid. In: Proceedings of the 9th Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob). pp. 19–24 (Sep 2019)
9. 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: IEEE Int. Symposium on Robot and Human Interactive Communication (RO-MAN). pp. 113–120. IEEE (2017)
10. Kerzel, M., Wermter, S.: Neural end-to-end self-learning of visuomotor skills by environment interaction. In: Lintas A., Rovetta S., V.P.V.A. (ed.) Artificial Neural Networks and Machine Learning – ICANN 2017. Lecture Notes in Computer Science, vol. 10613, pp. 27–34. Springer, Cham (Sep 2017)
11. Köker, R.: A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. Information Sciences **222**, 528–543 (2013)
12. Leitner, J., Harding, S., Förster, A., Corke, P.: a modular software framework for eye–hand coordination in humanoid robots. Frontiers in Robotics and AI **3**, 26 (2016)
13. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. The Journal of Machine Learning Research **17**(1), 1334–1373 (2016)
14. Lin, T.Y., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. 2017 IEEE Int. Conf. on Computer Vision (ICCV) pp. 2999–3007 (2017)
15. Marsland, S.: Machine Learning: An Algorithmic Perspective, Second Edition. Chapman Hall/CRC, 2nd edn. (2014)
16. Quillen, D., Jang, E., Nachum, O., Finn, C., Ibarz, J., Levine, S.: Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 6284–6291. IEEE (2018)
17. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems. pp. 91–99 (2015)
18. Savastano, P., Nolfi, S.: A robotic model of reaching and grasping development. IEEE transactions on autonomous mental development **5**(4), 326–336 (2013)
19. Starke, S., Hendrich, N., Magg, S., Zhang, J.: An efficient hybridization of genetic algorithms and particle swarm optimization for inverse kinematics. In: 2016 IEEE Int. Conf. on Robotics and Biomimetics (ROBIO). pp. 1782–1789. IEEE (2016)
20. Trask, A., Hill, F., Reed, S.E., Rae, J., Dyer, C., Blunsom, P.: Neural arithmetic logic units. In: Advances in Neural Information Processing Systems. pp. 8035–8044 (2018)