CIT433027 ADLR: Project Ideas

October 16, 2025

Here, you can find several ideas for projects we collected. Take this as an inspiration for your project. Some ideas are rather "big", meaning they could result in multiple projects. After registering your team (including a draft proposal), you will discuss the extent of your final proposal with your assigned tutor.

1 Investigating Rapid Motor Adaption

Qi et al. [1] use Rapid Motor Adaption to train a robust in-hand manipulation policy in simulation, which can be deployed on the actual system without any further adaptation. This is facilitated by a dedicated part of the neural network predicting the current world in an end-to-end learned latent space to make the policy more robust - first conditioned on a set of available states in simulation, later on only those measurable when deployed on the real system. We want to find the possible performance improvements and the limits of this approach in a simulation-only setting.

- Define and set up a training environment with your tutor. For example, we are interested in generating a robust in-hand manipulation policy. Also, we will decide whether you want to use an existing RL framework or implement the algorithm from scratch, depending on your focus.
- Analyze the performance gains as well as the limits of the Rapid Motor Adapton algorithm.
- Expand the algorithm to tackle these identified problems, for example, a different encoder architecture or changed loss functions, depending on your identified problems and feedback from the tutor.

2 Non-sequential Reinforcement Learning for Hard Exploration Problems

A way of overcoming the problems associated with sparse rewards in Reinforcement Learning is to utilize "expert demonstrations" in promising regions of the state space. Bauza et al. [2] use (suboptimal) expert demonstrations for providing a curriculum of interesting states to the agent during learning. In the absence of human experts, Blau et al. [3] propose to first generate successful trajectories by RRT-based planning algorithms and then initialize a policy using that data.

Based on one of the above papers, you could e.g. study one of the following problems (or come up with your own ideas):

- Implement different heuristics for choosing states to explore from. Can you also find a metric to decide when to *stop* an episode early?
- Implement an entirely different planning algorithms for obtaining expert demonstrations. Analyse the dependency between the quality of the demonstrations and the final performance of the policy.

3 Tactile Exploration of Objects

For grasping an object with a robotic hand, often a full 3D model of the given object is required. Using a (depth-)camera, one can infer the surface of the visible part of the object. However, the surface of the occluded parts is also usually needed, e.g., to robustly grasp the object. With tactile exploration (i.e., slowly moving the robotic hand until the fingers touch the object), it is possible to observe even the occluded parts of the object. This tactile information is usually very sparse. The idea is to use a learning-based approach to complete the full shape from this sparse contact information.

- Start in 2D: Train a neural network to complete a 2D shape based on a few given points similar to Watkins-Valls et al. [4] did in the more complex 3D case.
- Develop a strategy that determines the best way for the next tactile exploration based on previous points using reinforcement learning.
- Bring it into 3D: Complete 3D shapes in the same way as done for 2D
- Combine tactile and visual depth information to infer the object's shape

4 Grasping via Online Adaptation

One way to find grasps for a given object is to train a grasp success classifier and use that as an objective function for an online grasp optimization. Specifically, one differentiates through the grasp score prediction network to optimize the grasp parameters (input to the network) such that the grasp score (output of the network) gets maximized.

- A dataset of multi-finger grasps will be provided [5]
- Preprocess the data
- Train a grasp score prediction network [6]
- Implement an optimization procedure to perform grasping inference
- Optional: Train a small generative grasping network to provide better starting points for the optimization

5 Generative Networks for Robot Grasping

Generative neural networks can be used to directly generate stable grasps for a given unknown object. For a parallel jaw gripper, the network learns a distribution over 6D end effector poses conditioned on the observed object. For this

application, different architectures are possible: Variational Autoencoder, Generative Adversarial Networks, diffusion-based architectures [7] and autoregressive architectures [5]. The goal of this project is to apply one of the generative architectures to the problem of grasp generation.

- For training, we make use of the existing public training dataset Acronym [8]
- Preprocess the data
- Adapt the generative architecture for robotic grasping
- Optional: Extend the approach to generating grasps constraint to lie on a specified part of the object [9]

6 I know, what I don't know!

Investigate deep learning model capabilities to detect what was learned and what is an novel observation. Most deep learning models trained with maximum a posteriori only output a point estimate and not a model uncertainty. To put it simply: they cannot know, what knowledge they have. The goal of the project is to evaluate Bayesian deep learning methods from the lecture and so called normalizing flows [10] in this context.

- You can either use a provided dataset for tactile material classification or generate your own (proprioceptive) data using a simple robotic simulation environment for the benchmark.
- Implement at least two classification models or generative models depending on your benchmark (one Bayesian approximation from the lecture and one flow model).
- Evaluate how well each model can detect novel observations.
- Optional: Implement more models or think about scenarios where this feature is useful.

7 Learning the Inverse Problem

Look at the possibilities for representing inverse problems with neural networks. Ardizzone et al. [11] compare different flavors of GANs, VAEs and INN(theirs) for inverse problems in general. Extend their work into the robotic context, with more complex kinematics or sensor models. For the robot kinematic we have objective metrics to describe how well the generation task was performed. How can we use this knowledge to improve the training?

- What is the best approach to represent the high dimensional nullspaces for complex robot kinematics?
- When using a simple sensor (contact measurement between two fingers) we have only limited information about the true state of the robot, can we learn the distributions leading to the observed states?
- Lembono et al. [12] use an ensemble of GANs to reduce the mode collapse. What other options do we have to improve the generative model? How to measure the performance if the real distribution is not known?

8 Coverage Path Planning

For search and rescue missions (or vacuum cleaners), exploring the environment quickly and efficiently is vital. The goal of coverage path planning [13] is to navigate collision-free in a (possibly unknown) map and visit all unseen regions.

- Build a simple test environment; start with point robot in 2D.
- Formulate the objective: how do we measure and represent good coverage?
- Add more realism by including a sensor model (e.g., avalanche beacon).
- Add the kinematics to the problem and explore with a robotic arm [14].

9 Self-Supervised Learning for Robot Motion Planning

In Optimization-based Motion Planning, one formulates the problem of moving collision-free from A to B as finding the minimum of an objective function. We can use this objective directly to train a neural network without needing expensive data generation. This was already demonstrated to solve the Inverse Kinematics of a robot [15]. The goal is to extend the idea to the full motion planning problem (a library to compute the objectives is provided).

- Investigate the different aspects of motion planning and train individual networks for self-collision, collision with the environment, dynamics, ...
- Combine the various terms into the full motion planning problem.
- Replace the default Gradient Descent to update the network weights with a constrained-based optimizer to ensure the feasibility of the predictions.

10 Learning Robot and Environment Representations

A crucial aspect when using learning-based techniques in robotics is the representation and encoding of the relevant aspects of the problem. In the context of collision avoidance, the robot itself and the environment with different obstacles need to be modeled. The latter is especially relevant when considering dynamic settings with rapid changes. The goal is to explore ONE promising representation and analyze its potential for robot motion planning.

- Learn convex decomposition of arbitrary maps [16].
- Predict the Swept Volume of robot motions [17].
- Build a robot-centered signed distance field [18].

11 Trajectory Planning with Moving Obstacles

Drones not only have to plan flight paths through static environments, but also avoid collisions with dynamic objects. To learn such trajectories, a suitable encoding of the changing environment is crucial. Start with the Basis Points Set [19] and extend it to dynamic environments. Use this representation for neural motion planning [20].

- Come up with a state representation for dynamic environments.

- Set up a simple 2D (and later 3D) environment in which an agent can navigate through moving obstacles.
- Use RL to plan optimal trajectories in this environment.
- Optional: Extend the method to work with uncertainties in the motion prediction of the collision objects.

12 Recurrent Off-Policy Reinforcement Learning in POMDPs

In partially observable Markov decision processes (POMDPs), an RL agent has to be equipped with some sort of memory in order to be able to act optimally. A well-known method addressing this issue is to encode the history of observations by recurrent neural networks (RNNs). For example, for the class of off-policy methods, Heess et al. [21] combine RNNs with the DDPG algorithm, and Kapturowski et al. [22] study the interplay of DQN-based algorithms with recurrent experience replay. Based on this work:

- Your tutor will provide you with an environment that requires the use of memory to be solved optimally.
- Implement a recurrent version of the SAC algorithm by Haarnoja et al. [23].
- Assess the effect of different design choices and hyperparameters (e.g. hidden state initialization strategy in the experience replay buffer, truncated BPTT, ...)

13 Geometric Representations in Reinforcement Learning

Note: Requires previous experience with GNNs [24] and ideally with pytorch-geometric (link).

- Similar to Wang et al. [25]. Modify MuJoCo environments (Walker, HalfCheetah, Ant, Humanoid) such that the observations contain a graph representing the robot.
- Use message passing network(s) in addition or instead of the MLP for value/Q function and policy in standard algorithms like PPO Schulman et al. [26] or SAC Haarnoja et al. [23].

14 Diffusion Policies

Diffusion-based architectures have shown exciting results in image generation. Recently, researchers started to apply them to policy learning. Chi et al. [27] show promising imitation learning results with both vision- and state-based policies.

- Apply the diffusion policy [27] to simple benchmark environments with expert offline data (link).
- Analyze if the approach is better suited for specific types of environments (e.g., planning vs. control).

- Adapt the diffusion architecture for other types of environments.
- Optional: Develop and implement ideas on how the architecture can be used in a reinforcement learning setting.

15 Casting Sim2Real as Meta-Reinforcement Learning

The RL^2 algorithm evaluated by Yu et al. [28] promise sample-efficient meta-reinforcement learning, meaning that the algorithm can quickly adapt to new unseen tasks. We would like to use the capabilities to handle heavily randomized environments occurring in simulations that are designed to allow a real-world transfer of the policy.

- Find a suitable benchmark environment.
- Implement the algorithm on top of an existing basic RL algorithm and compare the performance.
- Examine the reward for constant as well as dynamically changing environments.
- Analyze different loss terms and evaluate the performance.
- Extend the algorithm to deal with problems occurring in Sim2Sim settings with unknown disturbances.

16 Information Bottleneck / Ignoring Noise

The reinforcement learning framework allows us to specify arbitrary observation spaces. For robotic tasks, in particular, we often intuitively understand what observations might be necessary to solve a given task. However, it is usually unclear what additional (readily available) information benefits training times and real-world performance. For practical purposes, it would be convenient to pass all the available information to the agent. That raises several questions:

- Can we pass too much information?
- Is redundant information harmful or maybe beneficial (similar to over-parameterization)?
- Can the agent learn to ignore noise inputs?

Explore those questions on simple environments (link) and come up with network architectures, such as self-attention (cf. Tang et al. [29]) or simple world models (encoder/decoder nets), to fix arising problems.

17 Model Predictive Control for Robotic Manipulation

Model predictive control (MPC) is a popular control strategy in robotics. Prior implementations in simulation [30] often assume that the underlying physical model is precisely known, which is, however, rarely the case in practice. Instead, Domain Randomization (DR) should be applied to cover the range of possible physical behaviors encountered in the real world. This project aims to investigate the limitations of simple MPC-based approaches in the presence of model uncertainty.

- If you feel comfortable coding in C++, extend the implementation provided in [30] to include randomization of different physical parameters (friction, masses, contact behavior ...).
- If not, implement a simple stochastic MPC [31] in Python and a simple test environment that allows for randomization of physical parameters (with the help of your tutor).
- Analyze the sensitivity of the MPC controller to the randomization of different types of physical parameters.

18 Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience

In Chebotar et al. [32], an efficient method for solving the sim2real problem by iteratively adapting the simulation parameters to the real system is proposed.

- Create an experimental sim2sim setup because no real robot is available (i.e., try to adapt one simulation to a given one with unknown parameters).
- Re-implement their algorithm based on the relative entropy policy search or using reinforcement learning algorithms.
- Investigate the convergence of estimated parameters.
- Investigate the influence of a broader starting distribution on the final performance.
- Investigate how re-using learned policies from previous iterations for initialization of the new network shortens training time and restrains the final performance.

19 Representing Shapes as Latent Codes

Efficient representations of 3D geometries are crucial for many robotic related task. For example, in-hand manipulation or grasping can be conditioned on the geometry of the object. Another task is tactile shape detection, where the geometry of an unknown object should be determined. Park et al. [33] use a "auto-decoder" structure to find a latent parametrization of 3D geometries.

- Start off with a simple 2D data set (e.g. segmentation masks), then transfer to 3D (e.g. single category of Shapenet)
- Train an auto-decoder architecture as in [33] to find a latent space representation of shapes.
- Implement and investigate improvements on the original paper:
  - Add additional regularization on the normal vectors as suggested in Gropp et al. [34]. Investigate whether these regularization lead to a more complete and continuous latent space.
  - Add the mode switching suggested in [35]. Investigate whether this can also improve the shape reconstruction or just leads to better behavior far away from the object.

– Investigate how the position encoding, suggested in [36], helps with high frequency features.
– Think of ways to use the learned latent space for generative models. Maybe one can make the auto-decoder variational?

20 Learning the Model of a Tactile Skin

Tactile sensing is crucial for fine manipulation with robotic hands. Modeling a tactile skin sensor, however, requires soft contacts to capture the softness of the fingers. This is computationally expensive and, hence, inefficient when used together with reinforcement learning. Narang et al. [37] train a neural network to map basic contact information such as the contact point and force to the deformation and the sensor response. Our sensor model [38] includes an expensive raycast operation and an equilibrium search that slow down the simulation. This could be mitigated by learning the sensor model beforehand.

- Familiarize yourself with our sensor model (code will be provided).
- Think of an efficient representation of the local geometry. For instance, in Zobeidi and Atanasov [39] they learn a directional distance function that could replace the raycasting.
- Learn the mapping for the correct deformation. This can either be done in an end-to-end fashion or starting from the raycast that provides the local geometry.

References

[1] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning (CoRL)*, 2022.

[2] Maria Bauza, Jose Enrique Chen, Valentin Dalibard, Nimrod Gileadi, Roland Hafner, Murilo F Martins, Joss Moore, Rugile Pevceviciute, Antoine Laurens, Dushyant Rao, et al. Demostart: Demonstration-led auto-curriculum applied to sim-to-real with multi-fingered robots. *arXiv preprint arXiv:2409.06613*, 2024.

[3] Tom Blau, Philippe Morere, and Gilad Francis. Learning from demonstration without demonstrations. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[4] David Watkins-Valls, Jacob Varley, and Peter Allen. Multi-modal geometric learning for grasping and manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[5] Dominik Winkelbauer, Berthold Bäuml, Matthias Humt, Nils Theurey, and Rudolph Triebel. A two-stage learning architecture that generates high-quality grasps for a multi-fingered hand. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[6] Mark Van der Merwe, Qingkai Lu, Balakumar Sundaralingam, Martin Matak, and Tucker Hermans. Learning continuous 3d reconstructions for geometrically aware grasping. In *International Conference on Robotics and Automation (ICRA)*, 2020.

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Neural Information Processing Systems*, 2020.

[8] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. Acronym: A large-scale grasp dataset based on simulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[9] Jens Lundell, Francesco Verdoja, Tran Nguyen Le, Arsalan Mousavian, Dieter Fox, and Ville Kyrki. Constrained generative sampling of 6-dof grasps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[10] R. Mackowiak, L. Ardizzone, U. Kothe, and C. Rother. Generative Classifiers as a Basis for Trustworthy Image Classification. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2021.

[11] Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. In *International Conference on Learning Representations*, 2018.

[12] Teguh Santoso Lembono, Emmanuel Pignat, Julius Jankowski, and Sylvain Calinon. Learning Constrained Distributions of Robot Configurations with Generative Adversarial Network. *IEEE Robotics and Automation Letters*, 2021.

[13] Arvi Jonnarth, Jie Zhao, and Michael Felsberg. Learning coverage paths in unknown environments with reinforcement learning. In *International Conference on Machine Learning*, 2024.

[14] Gaetano Meli and Niels Dehio. Robot cell modeling via exploratory robot motions. *arXiv preprint arXiv:2502.01484*, 2025.

[15] Johannes Tenhumberg, Arman Mielke, and Berthold Bäuml. Efficient learning of fast inverse kinematics with collision avoidance. In *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, 2023.

[16] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[17] John Baxter, Mohammad R. Yousefi, Satomi Sugaya, Marco Morales, and Lydia Tapia. Deep prediction of swept volume geometries: Robots and resolutions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[18] Baolin Liu, Gedong Jiang, Fei Zhao, and Xuesong Mei. Collision-free motion generation based on stochastic optimization and composite signed distance field networks of articulated robot. *IEEE Robotics and Automation Letters*, 2023.

[19] Sergey Prokudin, Christoph Lassner, and Javier Romero. Efficient Learning on Point Clouds with Basis Point Sets. In *International Conference on Computer Vision*, 2019.

[20] Ahmed Hussain Qureshi, Yinglong Miao, Anthony Simeonov, and Michael C. Yip. Motion planning networks: Bridging the gap between learning-based and classical motion planners. *IEEE Transactions on Robotics (T-RO)*, 2021.

[21] Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*, 2015.

[22] Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2018.

[23] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

[24] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[25] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *International Conference on Learning Representations*, 2018.

[26] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, 2017.

[27] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[28] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *CoRR*, 2019.

[29] Yujin Tang, Duong Nguyen, and David Ha. Neuroevolution of self-interpretable agents. In *GECCO '20: Genetic and Evolutionary Computation Conference, Cancún Mexico, July 8-12, 2020*. ACM, 2020.

[30] Deepmind. Mujocompc. `https://github.com/google-deepmind/mujoco_mpc`, 2022.

[31] Taylor Howell, Nimrod Gileadi, Saran Tunyasuvunakool, Kevin Zakka, Tom Erez, and Yuval Tassa. Predictive sampling: Real-time behaviour synthesis with MuJoCo. *arXiv preprint arXiv:2212.00541*, 2022.

[32] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan D. Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. *CoRR*, 2018.

[33] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[34] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proceedings of Machine Learning and Systems*. 2020.

[35] Puze Liu, Kuo Zhang, Davide Tateo, Snehal Jauhri, Jan Peters, and Georgia Chalvatzaki. Regularized deep signed distance fields for reactive motion generation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[36] Mauro Comi, Yijiong Lin, Alex Church, Alessio Tonioni, Laurence Aitchison, and Nathan F Lepora. Touchsdf: A deepsdf approach for 3d shape reconstruction using vision-based tactile sensing. *IEEE Robotics and Automation Letters*, 2024.

[37] Yashraj Narang, Balakumar Sundaralingam, Miles Macklin, Arsalan Mousavian, and Dieter Fox. Sim-to-real for robotic tactile sensing via physics-based simulation and learned latent projections. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[38] Ulf Kasolowsky and Berthold Bäuml. Fine manipulation using a tactile skin: Learning in simulation and sim-to-real transfer. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[39] Ehsan Zobeidi and Nikolay Atanasov. A deep signed directional distance function for shape representation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.