CIT433027 ADLR: Project Ideas

April 18, 2024

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

In Chebotar et al. [1], 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.

2 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. [2] 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

3 Self-supervised learning for robot grasping

Although grasping training data can be generated in simulation, it is computationally expensive to find high-quality grasps. Therefore, the training process should be as data-efficient as possible. In Deep Learning, self-supervised learning is one prominent way to reduce the required labeled data. This has been shown to generate useful features across various modalities like images, audio, and text. In this project, the goal is to apply self-supervised learning to 3D objects and use the resulting features for grasping.

- Familiarize yourself with the data2vec framework [3]
- Reimplement data2vec for 3D objects using a CNN
- Evaluate the learned features in grasping as a downstream task (data will be provided [4])
- Optional: Replace the CNN with a transformer

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 [4]
- Preprocess the data
- Train a grasp score prediction network [5]
- 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 [6] and autoregressive architectures [4]. 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 [7]
- 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 [8]

6 Tactile Material Classification

Tulbure and Bäuml [9] (as mentioned in the lecture) proved that one can classify a vast number of materials robustly ($\approx 95\%$) using only the spatio-temporal signal of a tactile skin.

- You will get the data of Tulbure and Bäuml [9]
- It is way easier to collect unlabeled samples than labeled. Can we learn a representation without the labels? Recent work in this direction is, e.g., a Joint Embedded Predictive Architecture (Assran et al. [10]).
- Instead of a Transform model, you should start with the TacNet (CNN) architecture of Tulbure and Bäuml [9] and check if you can validate the claims of Assran et al. [10] for the spatio-temporal tactile signals.

7 Unsupervised Skill Discovery / Curiosity

Pretraining neural networks in an unsupervised setting is extremely effective for language models. Models such as GPT or Bert can be fine-tuned or used directly on downstream tasks. Similarly, RL agents can be pretrained in an environment without an extrinsic reward signal and later be adapted to specific tasks. Laskin et al. [11] compare many different approaches on a unified benchmark (URLB).

- Skim recent literature on unsupervised RL (e.g. [12, 13, 14])
- Choose one/come up with your own ideas or modifications.
- Implement and compare them with the results reported by URLB.

8 Learning the Inverse Kinematics

Look at the possibilities for representing inverse problems with neural networks. Ardizzone et al. [15] compare different flavors of GANs, VAEs and INN(theirs) for inverse problems in general. Extend their simple robotic example of a planner arm to 3D, more DoFs, or multiple TCPs. Unlike in computer vision, for the robot kinematic we have solid metrics to describe how well the generation task was performed. How can we use this knowledge to our advantage?

- What is the best approach to represent the high dimensional nullspaces for complex robot geometries?
- Lembono et al. [16] 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 nullspace is not known?
- Predict not only the position of the TCP but also its rotation. How can one best represent the $SO(3)$?

9 Motion Planning with Diffusion

Look into generative models for robotic motion planning based on the work by Janner et al. [17]. The core of their approach lies in a diffusion probabilistic model that plans by iteratively denoising trajectories.

- Use a simple 2D robot to get familiar with the diffusion approach in the robotic context.
- Include the changing environment as a central part of the planning.
- Extend the framework to more complex robots.
- Alternatively, the Inverse Kinematics, with its inherent ambiguity, is a second promising testbed for the diffusion models.

10 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 [18] is to navigate collision-free in a (possibly unknown) map and visit all unseen regions.

- Build a simple test environment; focus on simple point robots in 2D/ 3D, no additional complexity through kinematics.
- Formulate the objective: how do we measure and represent good coverage?
- Add more realism by incorporating a sensor model (e.g., avalanche beacon).

11 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 [19].
- Predict the Swept Volume of robot motions [20].
- Build a robot-centered signed distance field [21].

12 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 Prokudin et al. [22] and extend it to dynamic environments. Use this representation for neural motion planning Qureshi et al. [23].

- 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.

13 Learning to Fly Beyond RL – Analytic Policy Gradient

Fixed-wing VTOL (Vertical Take-Off and Landing) drones combine the ability to take off vertically and exploit lift for efficient cross-country flight. The disadvantage is more complicated control. Learned controllers are a promising solution here. Wiedemann et al. [24] presents a method to directly exploit the differentiability of models and thus skip the detour via reinforcement learning.

- Setting up the learning pipeline with a simple quadrotor model (simple control).
- Replace the policy with a PID controller. Exploit the pipeline for automatic parameter tuning.

14 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 the class of off-policy methods, Heess et al. [25] combine RNNs with the DDPG algorithm, and Kapturowski et al. [26] 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. [27].
- Assess the effect of different design choices and hyperparameters (e.g. hidden state initialization strategy in the experience replay buffer, truncated BPTT, ...)

15 Differentiable Bayesian Filters

Prior knowledge of Bayesian filters highly recommended
Differentiable filters are a promising approach for combining the algorithmic structure of Bayesian filter techniques with the power of learning-based methods (for an overview of existing methods, see, e.g., Kloss et al. [28]). Importantly, differentiable filters offer a systematic way of dealing with aleatoric uncertainty in state estimation problems.

- Implement a *differentiable* filter of your choice, for example, EKF, UKF, or Particle Filter.

- Consider the simple tracking experiment VI from Kloss et al. [28]. Implement interesting modifications to the experiment. For example:
  - use the distance to (a subset of) beacons instead of images as measurements
  - implement collisions
  - additionally, estimate parameters from the system dynamics on-the-fly
- Compare your filter to recurrent neural networks and discuss the pros and cons.

16 Diffusion Policies

Diffusion-based architectures have shown exciting results in image generation. Recently, researchers started to apply them to policy learning. Chi et al. [29] show promising imitation learning results with both vision- and state-based policies.

- Apply the diffusion policy [29] 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.

17 Casting Sim2Real as Meta-Reinforcement Learning

The RL^2 algorithm evaluated by Yu et al. [30] promise sample-efficient meta-reinforcement learning, meaning that the algorithm can quickly adapt to new unseen tasks. We would like 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.

18 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. [31]) or simple world models (encoder/decoder nets), to fix arising problems.

19 Model Predictive Control for Robotic Manipulation

Model predictive control (MPC) is a popular control strategy in robotics. Prior implementations in simulation [32] 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 [32] to include randomization of different physical parameters (friction, masses, contact behavior ...).
- If not, implement a simple stochastic MPC [33] 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.

20 Investigating Rapid Motor Adaption

Qi et al. [34] 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.

21 Factory Manipulation Challenge

For this project, we will provide a MuJoCo environment with two robot arms and a conveyor belt: `https://youtu.be/IpwQPmgOLW0?si=mNzBQyJiv-8xCxzo/`. The task is to pick as many objects as possible from the conveyor belt and put them into a basket. We also provide you with a baseline policy that solves the task using classic methods from robotic control.

If you decide to take on this project, we encourage you to tackle ONE of the following problems:

- Use Reinforcement Learning (RL) from scratch to improve upon the provided baseline. This will involve designing a reward function and termination criteria, analyzing different input representations and network architectures, and choosing RL algorithms.
- Use Imitation Learning techniques (e.g. [35]), leveraging the provided baseline policy as a teacher. Can you use its demonstrations while still arriving at a better final policy?
- Consider the setting where the two robots work together to maximize the total number of objects collected. Train an RL policy to operate cooperatively alongside the provided baseline policy and analyze the emergent behavior.

References

[1] 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.

[2] David Watkins-Valls, Jacob Varley, and Peter Allen. Multi-modal geometric learning for grasping and manipulation. In *IEEE International conference on robotics and automation*, 2019.

[3] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. Data2vec: A general framework for self-supervised learning in speech, vision and language. In *International Conference on Machine Learning*, 2022.

[4] 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*, 2022.

[5] 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*, 2020.

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

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

[8] Jens Lundell, Francesco Verdoja, Tran Nguyen Le, Arsalan Mousavian, Dieter Fox, and Ville Kyrki. Constrained generative sampling of 6-dof grasps, 2023.

[9] Andreea Tulbure and Berthold Bäuml. Superhuman performance in tactile material classification and differentiation with a flexible pressure-sensitive skin. In *Proc. IEEE/RAS International Conference on Humanoid Robots*, 2018.

[10] M. Assran, Q. Duval, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, Y. LeCun, and N. Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15619–15629, Los Alamitos, CA, USA, jun 2023. IEEE Computer Society. doi: 10.1109/CVPR52729.2023.01499. URL `https://doi.ieeecomputersociety.org/10.1109/CVPR52729.2023.01499`.

[11] Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. URLB: unsupervised reinforcement learning benchmark. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1*, 2021.

[12] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse domains through world models. *CoRR*, abs/2301.04104, 2023.

[13] Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. Unsupervised reinforcement learning with contrastive intrinsic control. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 34478–34491. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/debf482a7dbdc401f9052dbe15702837-Paper-Conference.pdf`.

[14] Mengdi Li, Xufeng Zhao, Jae Hee Lee, Cornelius Weber, and Stefan Wermter. Internally rewarded reinforcement learning. In *Proceedings of the*

40th International Conference on Machine Learning, volume 202 of *Proceedings of Machine Learning Research*, pages 20556–20574. PMLR, 2023.

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

[16] 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.

[17] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*. PMLR, 2022.

[18] Arvi Jonnarth, Jie Zhao, and Michael Felsberg. Learning coverage paths in unknown environments with reinforcement learning. In *under review for ICLR 2024*, 2024.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] Nina Wiedemann, Valentin Wüest, Antonio Loquercio, Matthias Müller, Dario Floreano, and Davide Scaramuzza. Training efficient controllers via analytic policy gradient, 2023.

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

[26] 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.

[27] 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.

[28] Alina Kloss, Georg Martius, and Jeannette Bohg. How to train your differentiable filter. *Autonomous Robots*, pages 1–18, 2021.

[29] 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.

[30] 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.

[31] Yujin Tang, Duong Nguyen, and David Ha. Neuroevolution of self-interpretable agents. In Carlos Artemio Coello Coello, editor, *GECCO '20: Genetic and Evolutionary Computation Conference, Cancún Mexico, July 8-12, 2020*, pages 414–424. ACM, 2020. doi: 10.1145/3377930.3389847.

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

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

[34] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *6th Annual Conference on Robot Learning*, 2022. URL `https://openreview.net/forum?id=Xux9gSS7WE0`.

[35] Stephane Ross, Geoffrey J Gordon, and J Andrew Bagnell. A reduction of imitation learning and structured prediction to No-Regret online learning. 2010.