EE 290-005 Integrated Perception, Learning, and Control **Spring 2021**

Lecture 15: Vision-based Robotic Manipulation

Scribes: Albert Qü, Allen Shen

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

This section was presented by **Yahav Avigal** and **Shinwoo Choi**.

15.1.1 Sim-to-Real Overview

15.1.1.1 Motivation

Often training robots to perform tasks in real world environment can be costly, and it is in general difficult to transfer between tasks. Especially for tasks that require fine grained control and manipulations, classical control methods have limitations. The ability to train robots in a simulation environment and to directly adapt to a real environment yields a solution that is inexpensive, safe, and consistent while providing the ground truth. Simulation allows us to design, validate, and continuously update complex processes.

15.1.1.2 Problem

- Most simulators have limited modelling accuracy for real world characteristics such as friction, impact, and deformation. This problem is especially prominent for the **actuation** side.
- There are in general gaps between sensors and actuators in the real world versus perception in a simulated environment.

15.1.1.3 Solution

One prominent idea is to generate a policy given the physical model of the world (i.e. a simulation environment) and execute in the real world with robots. The difference between real world executed trajectory and simulated trajectory is then used to improve the model by changing model parameters.

- System Identification: Identify physical parameters of the environment relevant to the task and model it in simulation by observing the system's input and output relationships.
- Domain Adaptation: Transfer learning techniques for transferring the policies trained in simulation (source domain) to a real world environment (target domain).
- Domain Randomization: Randomize the simulations to cover reality as one of the variations.

15.1.2 Simulation-Based Design of Dynamic Controllers for Humanoid Balancing [1]

15.1.2.1 Key Challenge

Designing full reference trajectory offline robust to perturbations through iterated **system identification**.

15.1.2.2 Proposed Approach

The paper computes trajectories in simulation and executes them in a real environment; it utilizes the success information as feedback to update the simulation parameters. At each iteration, configurations are sampled, and trajectories are optimized and executed. Then, the real-world trajectory is evaluated against the simulation trajectory in simulation calibration to update simulation parameters. This process is iterated till convergence. (15.1)

- Physical simulation (DART): The paper simulates the dynamics of the robot and its interaction with the environment using DART. On the robot side, the robot is modeled as a rigid body satisfying a set of dynamic equations and non-penetration and linear complementarity conditions for contact points. $\mathbf{M}(\mathbf{q})$: mass matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$: Coriolis and centrifugal force, $\boldsymbol{\tau}$: joint torque, $\mathbf{K}$: friction cone, $\mathbf{f}$: contact force, $\mathbf{d}(\mathbf{q})$: distance of the contact to ground, $\mathbf{J}(\mathbf{q})$: Jacobian.

$$\begin{aligned} \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) &= \boldsymbol{\tau} + \mathbf{J}(\mathbf{q})^T \mathbf{f} \\ \mathbf{d}(\mathbf{q}) &\geq \mathbf{0} \\ \mathbf{d}(\mathbf{q})^T \mathbf{f} &= \mathbf{0} \\ \mathbf{f} &\in \mathbf{K} \end{aligned} \tag{1}$$

  Another key component is the actuator model based on the ideal DC motor assumption and the specification of the servo specified as follows. $\overline{q}$: reference input angle, k_p, k_d, k_c: actuator gains.

$$\tau = -k_p(q - \overline{q}) - k_d\dot{q} - k_c \text{sgn}(\dot{q}) \tag{2}$$

- Trajectory optimization algorithm: The reference trajectory is parameterized with a sparse set of keyframes. Only open loop trajectories are used because this poses a stronger requirement on the accuracy of the simulation. A sample-based stochastic optimization algorithm, Covariance Matrix Adaptation (CMA) [2], is used to optimize the reference trajectory according to a fitness function, which rewards the robot for remaining stably upright throughout the entire motion.
- Simulation calibration process: this is the major system identification component which minimizes model consistency. One critical challenge posed by the paper here is that a lot of system specifications described in the open source CAD file for robots are largely inaccurate. They parameterize the simulation model and minimize the differences between simulation trajectories and real-world trajectories.

15.1.3 Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping [3]

15.1.3.1 Key Challenge

This paper follows from Levine et al. 2016 [4] where large scale data collection was performed with over 800,000 grasp attempts. To increase data collection efficiency, they propose to train policy purely in simulation. However, grasp planners trained purely in simulation fail to generalize to the real world.

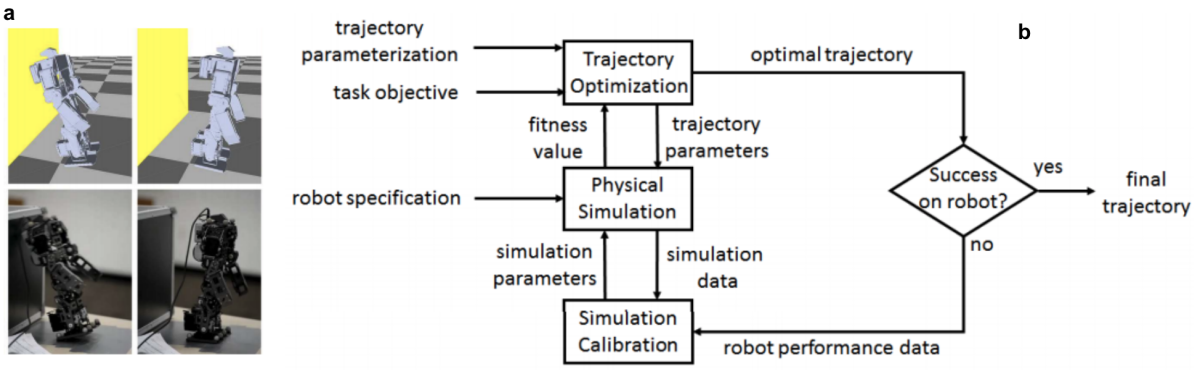


Figure 15.1: a. example of robot performing "Lean-to-Stand", top: training in simulation environment, bottom: testing in real world. b. Diagram for major component of the system.

15.1.3.2 Proposed Approach

The paper trains a grasp planner in simulation and uses **domain adaptation** to increase efficiency. More specifically, using both feature-level and pixel-level domain adaptation, they generate adapted images that are more realistic from synthetic data. (15.2 a)

- pixel-level: they developed a pixel-level domain adaptation method (GraspGAN) to train a grasping system to grasp novel objects from raw monocular RGB images. (15.2 b)
- feature-level: they used a DANN model that takes in x_0^f, x_c^f (the initial bin FOV without the gripper and the current bin FOV with the gripper respectively) and v_c^f (motor command) for both fake and real images and generates an evaluation of grasp accuracy. (15.2 c)

15.1.3.3 Comment

Prof. Malik: One meta comment about Sim-to-Real is that there are aspects of both perception and actuation. Generally we can do better job of perception in simulation because we understand the physics of light, and we can generate photo-realistic images. This paper doesn't do that, but it is actually possible. In general, I think some of the adaptation should not be on the perception side; you should just make the perception in simulation better. The actuation side is much more challenging because the physics of simulating deformation, contact and friction is harder, and we don't know those parameters. There we have to fight the battle with the sim-to-real methods.
Prof. Ma: I agree. We should focus on learning parts that we cannot simulate very well, which is the actuation part and not the perception part. The following link gives an idea how realistic simulated graphics can emulate real world scenes now: `https://structured3d-dataset.org`.

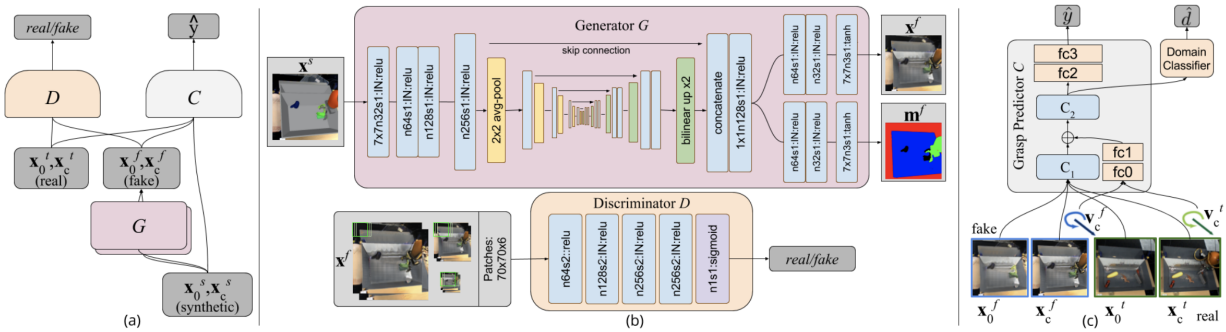


Fig. 4: **Our proposed approach: (a)** Overview of our pixel-level domain adaptation model, GraspGAN. Tuples of images from the simulation $\mathbf{x}^s$ are fed into the generator G to produce realistic versions $\mathbf{x}^f$. The discriminator D gets unlabeled real world images $\mathbf{x}^t$ and $\mathbf{x}^f$ and is trained to distinguish them. Real and adapted images are also fed into the grasp success prediction network C, trained in parallel (motion commands $\mathbf{v}$ are not shown to reduce clutter). G, thus, gets feedback from D and C to make adapted images look real and maintain the semantic information. **(b)** Architectures for G and D. Blue boxes denote convolution/normalization/activation-layers, where *n64s2:IN:relu* means 64 filters, stride 2, instance normalization *IN* and *relu* activation. Unless specified all convolutions are 3×3 in G and 4×4 in D. **(c)** DANN model: C_1 has 7 conv layers and C_2 has 9 conv layers. Further details can be found in [6]. Domain classifier uses GRL and two 100 unit layers.

Figure 15.2: excerpt visualization of the pipeline from GraspGAN paper

15.1.4 Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World [5]

15.1.4.1 Key Challenge

Specifically the authors focus on object location (translation) detection given an RGB image that is robust to noise or occlusion.

15.1.4.2 Proposed approach

The authors apply domain randomization and randomize the simulation environment in a parameter space (e.g. number and shape of distractor objects on the table, position and texture of all objects on the table, and textures of the table, floor, skybox, and robot) (15.3) so that real world situations are covered in the range of their simulation. They train in simulation, apply domain randomization during training, and test in the real world. With the vast simulation space, they effectively eliminate the need for accurate calibration in real world. In addition, they measured the following:

- Evaluate the localization accuracy of our trained detectors in the real world, including in the presence of distractor objects and partial occlusions
- Assess which elements of our approach are most critical for achieving transfer from simulation to the real world
- Determine whether the learned detectors are accurate enough to perform robotic manipulation tasks

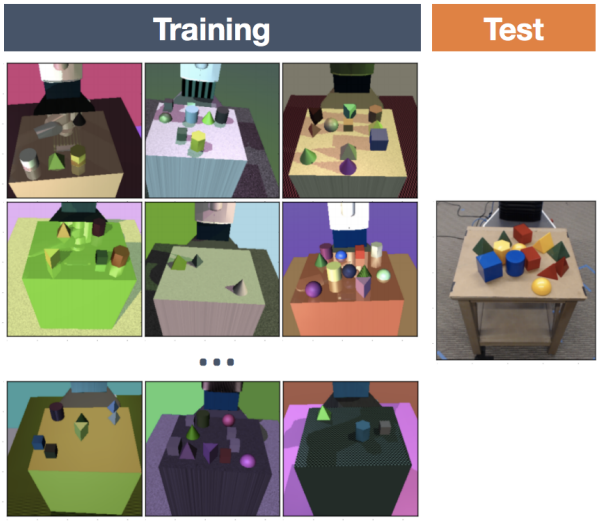


Figure 15.3: illustration of the approach. The object detector is trained on a vast set of simulated images.

15.1.4.3 Comments

Prof. Malik: Again, this is an example of not using CV knowledge. In this setting, the vision or the ability to detect these objects are being done tabula rasa. And then they encounter these problems which they deal with using domain randomization. However, one could bring an object detector such as a Mask R-CNN trained on realistic real world images. Part of the problem here is to not make use of a vision module and to do all the visual learning as part of an end-to-end task.
Yahav Avigal: Are you saying in general using domain randomization on vision is not necessary?
Prof. Malik: I would not make a general statement, but for the two examples today it is not necessary. One aspect is what Professor Ma and I talked about, which is that one can use much better physical simulations for synthetic images, so they don't look so bland. The second aspect is that vision can be brought in as a module without having to be trained to recognize objects as required in this task. I am all for domain randomization, but I would like to see it on motor/actuation side.

15.1.5 EPOPT: Learning Robust Neural Network Policies Using Model Ensembles

15.1.5.1 Key Challenge

There are discrepancies between the simulated source domain and the target domain. This is one critical aspect of the Sim-to-Real gap.

15.1.5.2 Proposed Approach

The Ensemble Policy Optimization (EPOpt) method optimizes a policy over an ensemble of models in an adversarial fashion. Then, it adapts the model distribution using data from the target domain. "Adversarial fashion" means that model instances on which the policy performs poorly in the source distribution are

sampled more often in order to encourage learning of policies that perform well for a wide range of model instances. Comparisons with standard policy search and evaluations on different models displayed robust and stable behaviors. The method composes of the following aspects:

- Robust Policy Search: They use batch policy optimization methods as a subroutine. To explicitly seek a robust policy, they use a softer version of the max-min objective suggested in robust control and optimize for the conditional value at risk (CVaR). Specially, they define a probability distribution with density $\mathcal{P}$ on p and the parameter of models $\mathcal{M}(p)$. Naturally, P defines a random variable of the parameter of the models. To evaluate a policy $\pi(\theta)$, they compute the following expected return (noted as $\eta_{\mathcal{M}}(\theta, p)$) over all random instances of the models at critical risk level ϵ with extremely low return. To solve the problem, they employed the EPOpt algorithm which is defined in 15.4:

$$\max_{\theta, y} \int_{\mathcal{F}(\theta)} \eta_{\mathcal{M}}(\theta, p)\mathcal{P}(p)dp, \ \ s.t.\ \mathbf{P}(\eta_{\mathcal{M}}(\theta, P) \leq y) = \epsilon.$$

- Adapting the source domain distribution: to update the model parameter distribution $\mathcal{P}$ and match to the target domain $\mathcal{P}^*$, the paper employs Bayesian inference with a particle filter, iteratively altering the ensemble distribution after observing trajectory data from the target domain.

Algorithm 1: EPOpt–ϵ for Robust Policy Search

Input: ψ, θ_0, $niter$, N, ϵ
for *iteration* $i = 0, 1, 2, \ldots niter$ **do**
  for $k = 1, 2, \ldots N$ **do**
    sample model parameters $p_k \sim \mathcal{P}_\psi$
    sample a trajectory $\tau_k = \{s_t, a_t, r_t, s_{t+1}\}_{t=0}^{T-1}$ from $\mathcal{M}(p_k)$ using policy $\pi(\theta_i)$
  end
  compute $Q_\epsilon = \epsilon$ percentile of $\{R(\tau_k)\}_{k=1}^{N}$
  select sub-set $\mathbb{T} = \{\tau_k : R(\tau_k) \leq Q_\epsilon\}$
  Update policy: $\theta_{i+1} = \text{BatchPolOpt}(\theta_i, \mathbb{T})$
end

Figure 15.4: Description of the EPOpt-ϵ algorithm for robust policy search

15.1.6 Auto-tuned Sim-to-Real transfer [6]

15.1.6.1 Key Challenge

Again, the authors attempt to learn a policy that is robust to Sim-to-Real transfer.

15.1.6.2 Proposed Approach

To automatically tune the parameters of the simulator to match the real world using only raw RGB images, the authors of the paper propose a Search Parameter Model (SPM) that re-frames the auto-tuning of parameters as a search problem where the simulation system parameters are iteratively shifted to approach the real world system parameters. (15.5) The approach works quite well in various task environments like "Peg-in-Hole", "Cheetah Run", "Open Cabinet", etc.

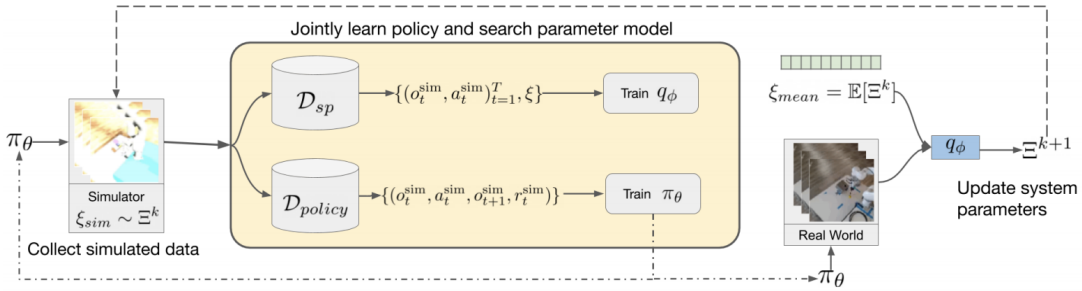


Fig. 2: Overall System: Using any off-the-shelf RL algorithm, we use simulated data to train both our policy and a Search Parameter Model (SPM) q_ϕ which predicts whether a candidate set of system parameters ξ is higher or lower than those which produced an observed trajectory. We iteratively update our simulation by running our policy in the real world and using our SPM to predict which direction to update our simulator to make it closer to the real world.

Figure 15.5: Description for the full Search Parameter Model.

15.1.7 Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience [7]

15.1.7.1 Key Challenge

Quite similar to aforementioned papers, the paper tries to transfer policies trained in simulated environment to the real world to close the Sim-to-Real gap.

15.1.7.2 Proposed Approach

One common theme of the literature in Sim-to-Real is to optimize over simulation parameter space to match the simulated model to target dynamics in the real world. Similarly, this paper uses **domain randomization** to represent stochastic dynamics, explicitly evaluates the discrepancies between simulated and executed trajectories, and iteratively updates the source distribution to match real-world dynamics. (15.6)

- reparameterization trick: in order to produce stochastic dynamics with a deterministic simulation engine, the paper samples the parameter $\xi \sim p_\phi(\xi)$. The simulation transition dynamics becomes $P(s_{t+1}|s_t, a_t, \xi)$.
- parameter optimization to match target dynamics: this is to make the policy generate observations close to those of the real world. At each iteration, real world trajectories and simulated trajectories are sampled using policy $\pi(\theta, p_\phi)$, and the dynamics distribution $P_{\xi \sim p_\phi}$ is compared using a discrepancy measure, which is a weighted combination of the l_1 and l_2 norm of the importance weighted trajectory difference:

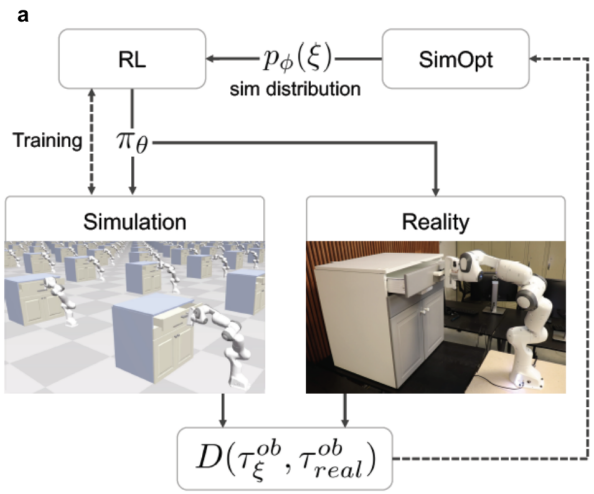
$$D(\tau_\xi^{ob}, \tau_{real}^{ob}) = w_{l_1} \sum_{i=0}^{T} |W(o_{i,\xi} - o_{i,real})| + w_{l_2} \sum_{i=0}^{T} ||W(o_{i,\xi} - o_{i,real})||_2^2$$

Then, the new parameter ϕ_{i+1} is learned through the following optimization problem subject to the KL-divergence step ϵ constraint, which keep the new distribution in a trust region:

$$\min_{\phi_{i+1}} \mathbb{E}_{P_{\xi_{i+1} \sim p_{\phi_{i+1}}}} [\mathbb{E}_{\pi_{\theta, p_{\phi_i}}} [D(\tau_{\xi_i}^{ob}, \tau_{real}^{ob})]], \ s.t. \ D_{KL}(\phi_{i+1} || \phi_i) \leq \epsilon$$

15.1.7.3 Results

- Implementation: the task execution is carried out with 7-DoF Franka Panda and ABB Yumi robots. The RL training and SimOpt simulation parameter sampling operate on a cluster of 64 GPUs for running the simulator with 150 simulated agents per GPU. To observe the real world trajectory, the paper performs object tracking with DART.
- Peg-in-hole: at each iteration, the experiment performs 100 iterations of RL, 3 roll-outs on the real robot, and runs 3 update steps of the distribution with 9600 simulation samples per update. After two SimOpt iterations, the trained policy is able to swing the peg into the hole with 90% success rate over 20 trials.
- Drawer-opening: at each iteration, the experiment performs 200 iterations of RL, 3 roll-outs on the real robot, and runs 20 update steps of the distribution with 9600 simulation samples per update. After one SimOpt iteration, the robot is able to better control its gripper orientation with 100% success rate over 20 trials.
- Evaluation: The paper showed that policies can be transferred with only a few iterations of simulation updates and a small number of real robot trials. But currently the paper only learned uni-modal simulation parameter distributions, which could only represent a restricted subset of structures in real world dynamics.



b

Algorithm 1 SimOpt framework

$p_{\phi_0} \leftarrow$ Initial simulation parameter distribution
$\epsilon \leftarrow$ KL-divergence step for updating p_ϕ
for iteration $i \in \{0, \ldots, N\}$ **do**
  env $\leftarrow$ Simulation(p_{ϕ_i})
  $\pi_{\theta, p_{\phi_i}} \leftarrow$ RL(env)
  $\tau_{real}^{ob} \sim$ RealRollout$(\pi_{\theta, p_{\phi_i}})$
  $\xi \sim$ Sample(p_{ϕ_i})
  $\tau_\xi^{ob} \sim$ SimRollout$(\pi_{\theta, p_{\phi_i}}, \xi)$
  $c(\xi) \leftarrow D(\tau_\xi^{ob}, \tau_{real}^{ob})$
  $p_{\phi_{i+1}} \leftarrow$ UpdateDistribution$(p_{\phi_i}, \xi, c(\xi), \epsilon)$

Figure 15.6: a. schematics for SimOPT. b. algorithm formulation of SimOpt.

15.1.7.4 Comments and Questions

Question (Albert Qü): How were the importance weights generated in the discrepancy measure?
Prof. Ma: At the high level this really is an adapted simulation that tunes the parameters with which the simulation is performed so that the policy executed is computed in the best emulated simulation environment. But I guess the discrepancy measure may not matter too much given that they are proper norms.
Yahav Avigal: I think they adopted this discrepancy measure from another paper and used the same parameters.
Prof. Ma: Yeah, I doubt this would matter too much as long as they are proper norms. But the idea here is very typical of how we perform control where we observe the discrepancy between the expected and actual

response after execution of a policy, and then we use this discrepancy as a feedback signal that optimizes the control or policy.

Question (Allen Shen): How did they initialize the simulation parameter distribution?
Answer: A human is still needed to initialize the covariances of the parameters. Although the method still requires manual parameter selection, they found that finding a conservative distribution is easier than designing a wide distribution [8].

Comment (Prof. Tomlin): We have a project which is a manufacturing assembly problem with a real ABB robotic arm on Ford's production line. Certain things on the production line are human operated because they do not know how to automate them. For instance, with fragile pieces like a dashboard with pins that break easily, one has to have a lot of fine grained and careful handling to see if the pins fit in and then snap them in appropriately. As you see in the policies learned from the demos in these papers, you have grippers that come in at a lower position but slide to feel for the handle. It is those kinds of learned policies that are exciting. These are not things that we code into control algorithms (maybe we could and certainly we could do a lot with control), but we want to capture fine grained wiggling and gentle pushing in a way that is intuitive to human actions because this is what the person on the line at Ford is doing. I would really love to have this kind of architecture where one gets learning inserted in these places in addition to what control has to offer to us. This can be really useful in generating strong policies.
Prof. Malik: I have a similar observation from our work in legged locomotion. When you do RL, which is basically a lot of trial and error, you try out certain things that may turn out to be good strategies which you might not have thought of in advance. One of the things that emerged to us was that this robot, while trying to get over a rock, lifts its legs incrementally higher until it can cross. And this is not programmed but instead arose because this was what the algorithm discovered and learned in the simulator environment with different random terrains.
Prof. Ma: Also, I think there is a little more to what Professor Tomlin just said. No matter how rich the simulation environment is, there is still need for instantaneous real world feedback. We can always try to optimize through either re-training or data augmentation. However, no matter how we randomize the domain, there is always going to be some discrepancy, which we can only get through policy execution. From the instantaneous feedback, there should be a mechanism that informs us how to adapt on the fly. And that is something that we could not quite derive from the current learning yet. That is why maybe only humans can work on certain tasks because those tasks require the agent's capability to assess the current situation better in real time based on experience.

15.2 Vision Based Control for Hands

We transition to vision based control for multi-fingered hands with a focus on the DexPilot paper by Handa, Van Wyk, et al. [9]. Hands are interesting because they appear in many disciplines (i.e. evolution, biomechanics, philosophy, computer science, etc.). This section was presented by Ilija Radosavovic and Zhe Cao.

15.2.1 Historical Background

We briefly discuss the evolution of the human hand throughout history.

Australopithecus afarensis is one of the best-known early human species with fossils dating to around 3.2 million years ago [10]. Discovered in 1974, Lucy is one of the members of this species, and we can consider

her as the earliest bipedal predecessor of humans. Since she was bipedal, her hands were freed up for non-locomotive tasks such as tool building and other interactions with her surrounding environment. In 1924, Raymond Dart hypothesized that these complex interactions with the environment using hands led to the development of human brains [11]. Even though chimpanzees and Lucy have similar brain size, they both have a much smaller brain compared to a modern day human [12]. This gives credence to Dart's hypothesis; however, there is no definitive proof that shows that the hypothesis is true, and there is still ongoing debate.

In Greek philosophy, there is also some debate with respect to hands. Aristotle famously claimed: "Anaxagoras says that man is the most intelligent of the animals because he has hands, but it would be better to say that he has hands because he is the most intelligent." [13]

Furthermore, there are many examples of hands in art. These include:

- Cueva de las Manos, Argentina, 11000 BC. This is an example of hand stencils inside a cave.
- Death of Agamemnon, 460 BC.
- Leonardo da Vinci, Study of Hands, 1474. Although hands were usually depicted as simplistic in medieval times, Leonardo da Vinci highlighted the anatomical structure of hands in his drawings. At around this time, hands began to be depicted more realistically.
- Michelangelo, Creation of Adam, 1512. This is perhaps the most famous depiction of hands in art.

In the 16th century, mechanical hands began to appear starting with the mechanical hand developed by Ambroise Paré in 1564; this mechanical hand was designed to replace limbs that soldiers lost in war. More recent examples of robotic hands include a hand developed by Professor J. Kenneth Salisbury at Stanford in 1982, a hand developed by a group at Utah/MIT in 1989, and the DLR hand developed in 1998 from Germany. There are also dozens of other examples of robotic hands developed in the past century. (The content in this paragraph is borrowed from Professor Sastry's lectures in the EECS 106/206 series [14] [15]).

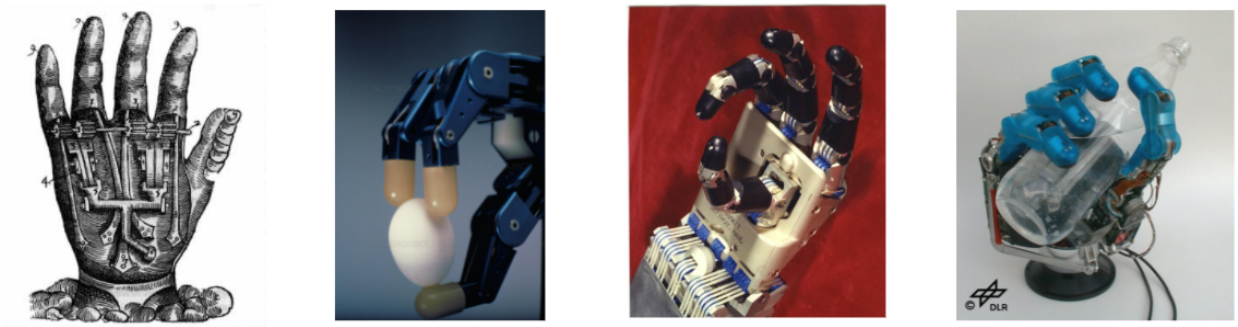


Figure 15.7: Examples of robotic hands

Robots deployed in factories today do not use multi-fingered hands because they are not necessary for most tasks in automation (i.e. picking and placing parts of a car). However, we might need to develop more sophisticated robots with multi-fingered hands to accomplish the diverse set of human tasks.

15.2.2 Generality Cost

There are three types of cost that we need to consider in order to design robotic hands for more general tasks.

- Mechanics. Human hands are very complex with many joints and muscles, so building a mechanical system for a hand that allows for actuation is a challenging task.
- Sensing. Human hands are equipped with thousands of touch sensors that collaborate with visual inputs.
- Control. It is believed that around 30 to 40% of the motor cortex is spent on controlling hands. We will focus on this portion for the rest of the lecture.

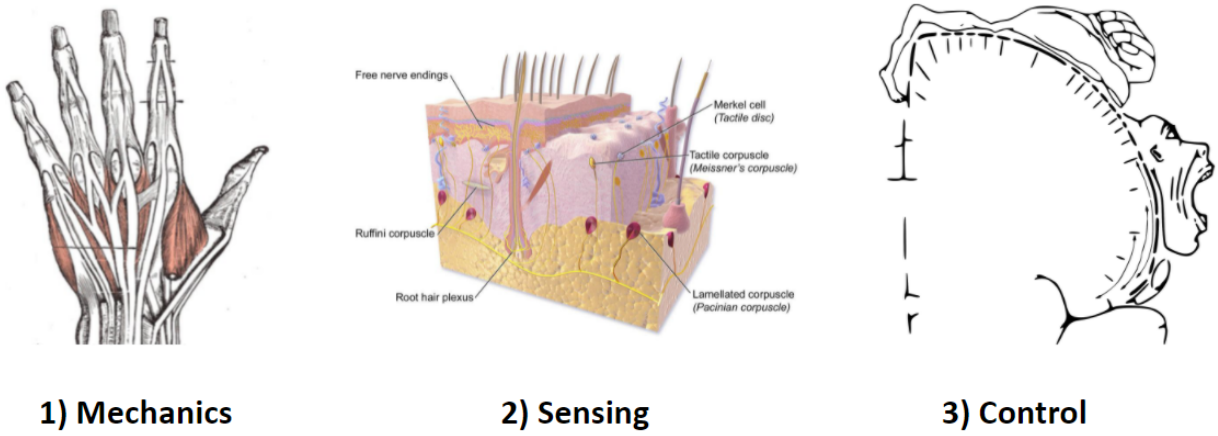


Figure 15.8: Three types of cost for generality

15.2.3 Trajectory Optimization

Traditionally, control has been tackled using a trajectory-based optimization approach, which assumes maintained contact with an object [16] [17]. Although these approaches work well in certain settings, it is hard to formulate these methods for complex tasks.

Recently, there have been a lot of interest in reinforcement learning approaches, which learn from trial and error by interacting with the environment. The previous work discussed in this lecture roughly falls under this realm while the rest of this lecture focuses on a vision-based approach for both state estimation and learning. We also examine imitation learning, which is a paradigm that allows a robot to replicate actions from a human instructor.

15.2.4 DexPilot

DexPilot uses vision based teleoperation, which enables a robotic, multi-fingered, dexterous hand to replicate the motions of a human hand [9]. The following video shows an example of an experiment with DexPilot:

`https://www.youtube.com/watch?v=qGE-deYfb8I`. The video is sped up, so in reality the human has to be extremely careful with their movements.

Q (Professor Ma): Why does the human have to perform their actions in the air instead of on objects? This seems counter-intuitive because the human has to imagine where the object is located, and humans are better at handling movements to assess the right control.

A (Ilija Radosavovic and Zhe Cao): Objects could make the state and pose estimation of the hand more difficult. Static objects could be added to the human area, but between the human and robot, different actions could result in different consequences for movable objects.

Comment (Professor Ma): Humans are learning a more difficult task to send the right signal to control the robot hand. Nevertheless, the control of the robot hand is an impressive feat.

15.2.4.1 Related Work

Prior work has required the use of gloves for teleoperation [18] [19]. These gloves would be accompanied by IMU and touch sensors, or they would be setup in a VR environment to make the task of recognizing and tracking the hand easier. The DexPilot work seeks to relax the glove constraint and to use bare hands for teleoperation.

15.2.4.2 System Setup

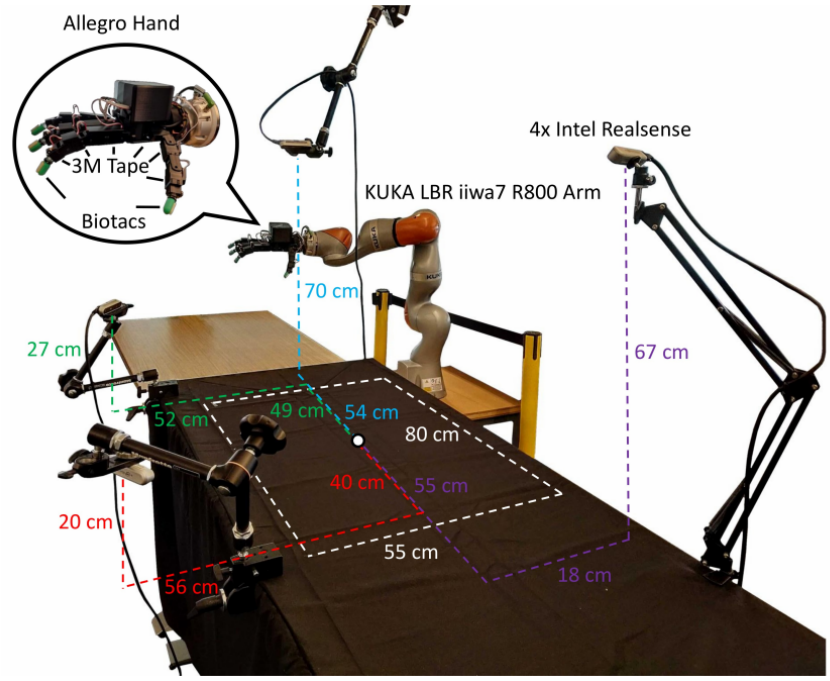


Figure 15.9: System setup for teleoperation. The human performs actions in a human arena (on the bottom right), and the robot replicates these actions (on the top left). The four RGB cameras obtain a multi-view coverage of the space at the appropriate distance. The tactile sensors on the Allegro hand are not used in this project, but in the future they could facilitate data collection for imitation learning.

15.2.4.3 Method Pipeline

The method pipeline for DexPilot consists of the following three threads that operate on three different computers.

- Learning Thread. In this thread, the authors train a neural network to perform a perception task. The input of the system is a point cloud obtained by 4 depth sensors, and the system attempts to estimate hand pose and joint angles using segmentation and pose estimation.
  - Their pose estimation method contains two stages. The first stage takes a point cloud input which contains part of the human body, the arm, and the hand; it seeks to segment out the hand from the rest of the body. The second stage takes the segmented hand and pose as input, and it performs a more fine-grained estimation of hand pose. There is also a third network to map the hand pose from the joint space to the angle space.
  - For data collection, the authors use gloves with colored spots corresponding to different points on the human hand. There are three spots on the back of the hand to estimate the overall global orientation of the 3D hand. They use color thresholding to collect 2D hand pose data and hand segmentation. Using depth values from the depth sensor, they convert the 2D data to 3D hand poses.
- Tracking Thread. They use depth sensor input and articulated hand model tracking to map from the human hand to the robot joint space.
  - Prior work on depth-based hand tracking includes DART from Schmidt, et al. [20]. In this work, the authors use articulated tracking to minimize the distance between the hand model and observations of a point cloud with a signed distance function field. DART generally works pretty well; however, it is dependent on proper initialization, and the whole tracking can fail if the method fails on a few frames.
  - To overcome the initialization issue, the DexPilot authors train a neural network for pose estimation to obtain the appropriate initialization for tracking. They use DART tracking after receiving the initialization from the previous stage.
  - The authors use kinematic retargeting to map finger tips from a five finger human hand to a four finger robot hand. They look at the distance and direction of the following two types of vectors:
    - Vectors that originate from a primary finger (i.e. one of the middle three fingers) and point to the thumb.
    - Vectors between two primary fingers.
- Control Thread. This module contains Riemannian motion policies that map from the joint angle space to the control action space.
  - The thread provides the target joint angles for the robot arm and hand given the estimated human hand pose and joint angles. The Riemannian motion policies combine trajectory generation and collision avoidance in the same optimization framework.

15.2.4.4 Evaluation

The authors evaluate their method with a wide variety of manipulation tasks (i.e. stacking blocks and pouring beads) that require different manipulation skills (i.e. grasping and releasing). Using demonstrations from two human pilots, they measured the mean completion time and success rate of these manipulation

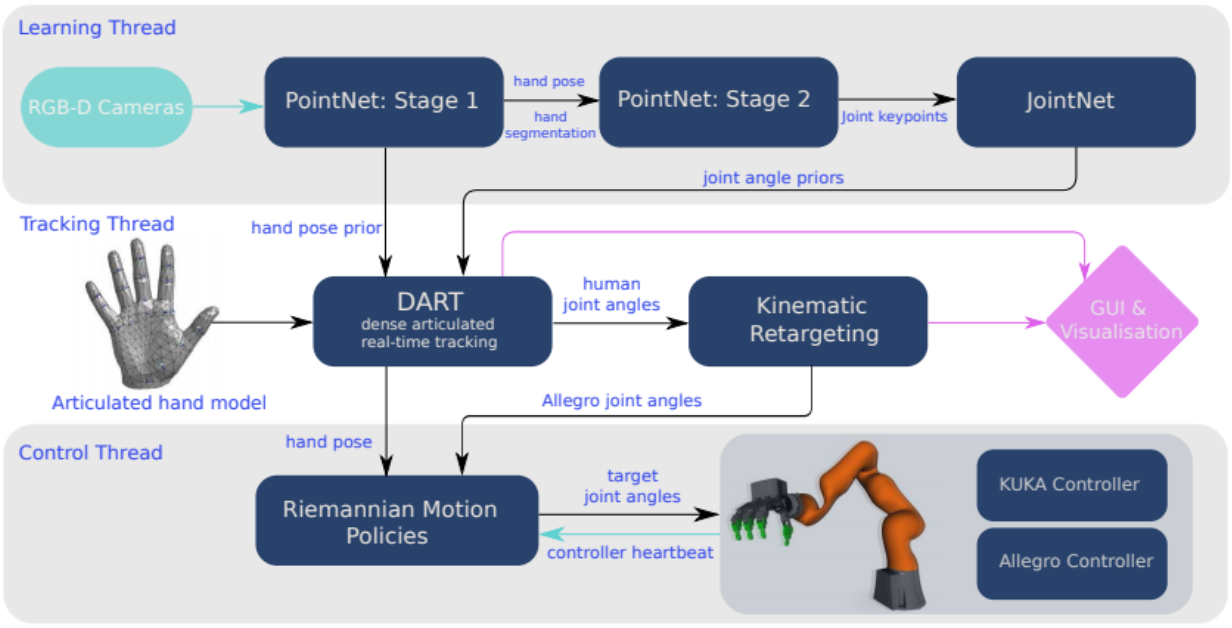


Figure 15.10: Method pipeline for DexPilot

tasks. For some tasks, there is a lot of variance between the completion times of the two human pilots; this is potentially caused by minor differences in the human instructions. Furthermore, we can see that completion time is highly correlated with the difficulty of the task. For example, the task of opening a wallet and pulling out paper money from the wallet takes takes much longer than the task that involves pouring beads. The task with the highest mean completion time is a complex multi-stage container task, which consists of opening a plastic container, extracting a cardboard box, and opening the box. A video of the container task can be seen here: `https://www.youtube.com/watch?v=P2MdhmDPw9w`.

Comment (Professor Malik and Zhe Cao): Some of these tasks can be bi-manual tasks that involve both hands instead of just one. It is very difficult to open things with one hand; this makes some tasks such as the wallet task very challenging.

In general, the success rate is very high for both pilots, and there is not a lot of variance between pilots. One possible reason for the high success rate is that DexPilot uses the human perception system, which provides feedback that allows the robot to perceive human reactions and to perform any necessary refinements.

Comment (Professor Malik, Professor Ma, and Zhe Cao): This seems more like an advertisement for the hand hardware since nearly everything is being done by the human. With enough degrees of freedom, we can design a system that completes these tasks. Overall, this system is an example of a leader-follower model; however, one artificial aspect is that the human is not getting tactile feedback while the robotic hand could potentially receive tactile feedback if there was tactile sensing. This is also a learning experience for the human pilot; a randomly chosen human would probably not succeed in performing these teleoperation tasks (unless they are a mime artist).

Other videos of the teleoperation tasks can be found on the DexPilot website: `https://sites.google.com/view/dex-pilot`.

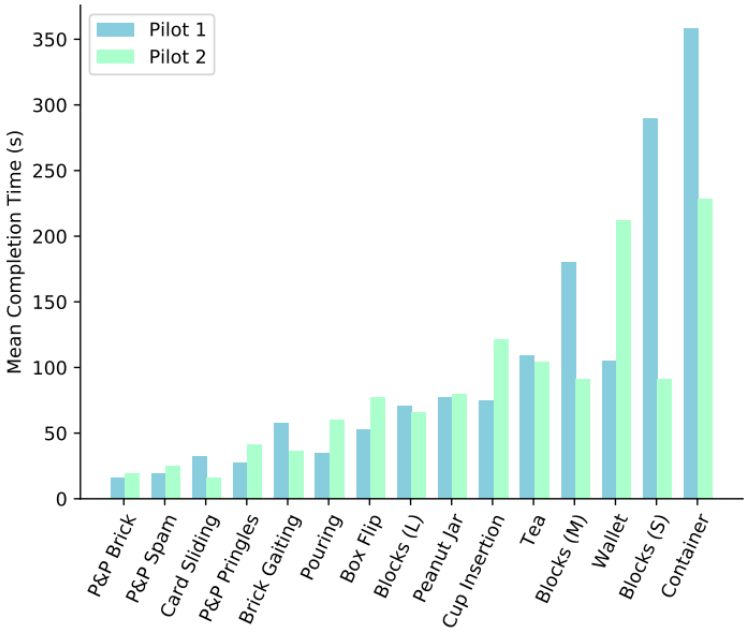


Figure 15.11: Mean completion time of various teleoperation tasks across two pilots run over five successive trials without reset.

15.2.4.5 Limitations

- This method requires four depth sensors to obtain full coverage of the workspace in the human arena. These sensors are limited in range, so the quality of hand estimation would begin to deteriorate beyond a certain range. For future work with better RGB-D cameras, the observable work volume of the pilot could be enlarged to allow for tasks that cover greater distances.
- The projection schemes in kinematic retargeting can interfere with finger gaiting tasks and timely releasing grasps on small objects. It seems to be easier to pick up objects compared to releasing objects with this method.
- The lack of tactile feedback makes precision tasks difficult to complete.
- In general, the system is very slow due to all of the modules involved. For future work, the overall system latency could be reduced to enable faster reactions.
- The demonstrations are limited for industrial applications because it is time-consuming to collect all of the human demonstration data.
  - Nevertheless, the paper provides a mapping from human arm space to robot arm space. This system could potentially be used for reinforcement learning, and demonstration videos could help speed up the imitation learning pipeline.

15.2.4.6 Contributions

- Markerless vision-based teleoperation system for a dexterous robot hand and arm
- A technique for kinematic retargeting of human hand joints to robot hand joints

- Demonstration of teleoperation system on a wide variety of challenging object manipulation tasks

15.2.5 Discussion

Comment (Professor Ma): Spinning a pencil in your hand is a challenging task that requires multi-finger coordination in both space and time. Tactile sensing could be important because we need a subtle feeling of friction and weight. Visual cues do not help us determine the center of mass of an object while tactile sensing does.

Comment (Professor Malik): We lose a lot of information if we ignore tactile information. When an object is in your hand, vision is hampered due to occlusion, but you still have tactile information. On the other hand, vision is used for sensing at longer distances. In reality, vision and touch are complementary, and we should be able to perform more complex tasks by combining the two.

Comment (Professor Sastry and Malik): The technology of tactile sensing is underdeveloped, but we should still try to build tactile sensors even if they are low resolution. At Berkeley, Professor Ana Arias, who works on flexible electronics, and Professor Ali Javey could be good leads to construct a tactile skin on the hand for experimental purposes.

Comment and Question (Professor Ma): We mostly use hands to manipulate objects as tools (i.e. writing with a pencil); picking up objects is an easier task that a bird's beak can do. Without sensing pressure and friction, the human hand is very limited.

Comment (Professor Malik): Roberto Calandra, a Facebook AI researcher, works on tactile sensing hardware as well as simulation environments for tactile sensing. In general, the tactile sensing space is very undeveloped, and not many other people work in the area. There is a bit of a chicken and egg problem because robotic hands are very expensive and do not have great tactile sensing.

Comment (Professor Malik): I started working with vision and robotics around 40 years ago when there were no digital cameras. There was only a small set of images with which the entire computer vision community did research. Later, we got CMOS cameras, and many individuals began to own personal HD cameras. Despite the drastic development in cameras, we have been using parallel jaw grippers for around 40 years. There is a whole avenue of research that could happen with more advanced robotic tools.

Comment (Professor Sastry and Ma): When we return to Berkeley, we need to do more experiments in both walking and grasping, especially with multi-fingered hands. We should combine ideas from tactile sensing, vision, control, and learning to obtain the best results.

Comment (Jay Monga): I understand that Riemannian motion policies are relatively new in robotics, and the NVIDIA group which produced the DexPilot paper is using these motion policies in a lot of their research.

Comment (Professor Sastry, Ma, and Malik): The purpose of the parallel jaw gripper is to immobilize objects (i.e. power grasping). However, manipulating objects with your fingers is a much more dexterous task than power grasping. After all, many other animals can power grasp, and chopsticks can be used to pick things up. Power grasping is the first degree of freedom that kids develop within their first 6 months. After this period, children develop skills that require more precision.

Comment (Ayush Agrawal): There is some recent work from CMU on tactile sensing by Yamaguchi and Atkeson [21]. They use vision-based tacile sensing similar to GelSight tactile sensing by Li, et al. from MIT [22].

References

[1] J. Tan, Z. Xie, B. Boots, and C. K. Liu, "Simulation-based design of dynamic controllers for humanoid balancing," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2729–2736.

[2] N. Hansen, "The CMA evolution strategy: A tutorial," *CoRR*, vol. abs/1604.00772, 2016. [Online]. Available: http://arxiv.org/abs/1604.00772

[3] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," 2017.

[4] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *CoRR*, vol. abs/1603.02199, 2016. [Online]. Available: http://arxiv.org/abs/1603.02199

[5] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," 2017.

[6] Y. Du, O. Watkins, T. Darrell, P. Abbeel, and D. Pathak, "Auto-tuned sim-to-real transfer," 2021.

[7] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," 2019.

[8] Simopt. [Online]. Available: https://sites.google.com/view/simopt

[9] A. Handa, K. Van Wyk, and et al., "Dexpilot: Vision based teleoperation of dexterous robotic hand-arm system," *IEEE Conference in Robotics and Automation (ICRA)*, 2020.

[10] Australopithecus afarensis. [Online]. Available: https://humanorigins.si.edu/evidence/human-fossils/species/australopithecus-afarensis

[11] Bipedalism vs. brain size. [Online]. Available: http://efossils.org/book/bipedalism-vs-brain-size

[12] How big was lucy's brain? [Online]. Available: http://www.efossils.org/book/how-big-was-lucys-brain

[13] Aristotle, *On the Parts of Animals*, 350 BC.

[14] Introduction to robotics. [Online]. Available: https://ucb-ee106.github.io/106a-fa20site/

[15] Robotic manipulation and interaction. [Online]. Available: https://ucb-ee106.github.io/106b-sp21site/

[16] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *SIGGRAPH*, 2012.

[17] B. Sundaralingam and T. Hermans, "Relaxed-rigidity constraints: In-grasp manipulationusing purely kinematic trajectory optimization," *Robotics: Science and Systems (RSS)*, 2017.

[18] H. Liu and et al., "High-fidelity grasping in virtual reality using a glove-based system," *International Conference on Robotics and Automation (ICRA)*, 2019.

[19] H. Liu, X. Xie, and et al., "A glove-based system for studying hand-object manipulation via joint pose and forcesensing," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[20] T. Schmidt, R. Newcombe, and D. Fox, "Dart: Dense articulated real-time tracking," *Robotics: Science and Systems (RSS)*, 2014.

[21] A. Yamaguchi and C. G. Atkeson, "Tactile behaviors with the vision-based tactile sensor fingervision," *IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2019.

[22] R. Li and et al., "Localization and manipulation of small parts using gelsight tactile sensing," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.