

Learning for Manipulation



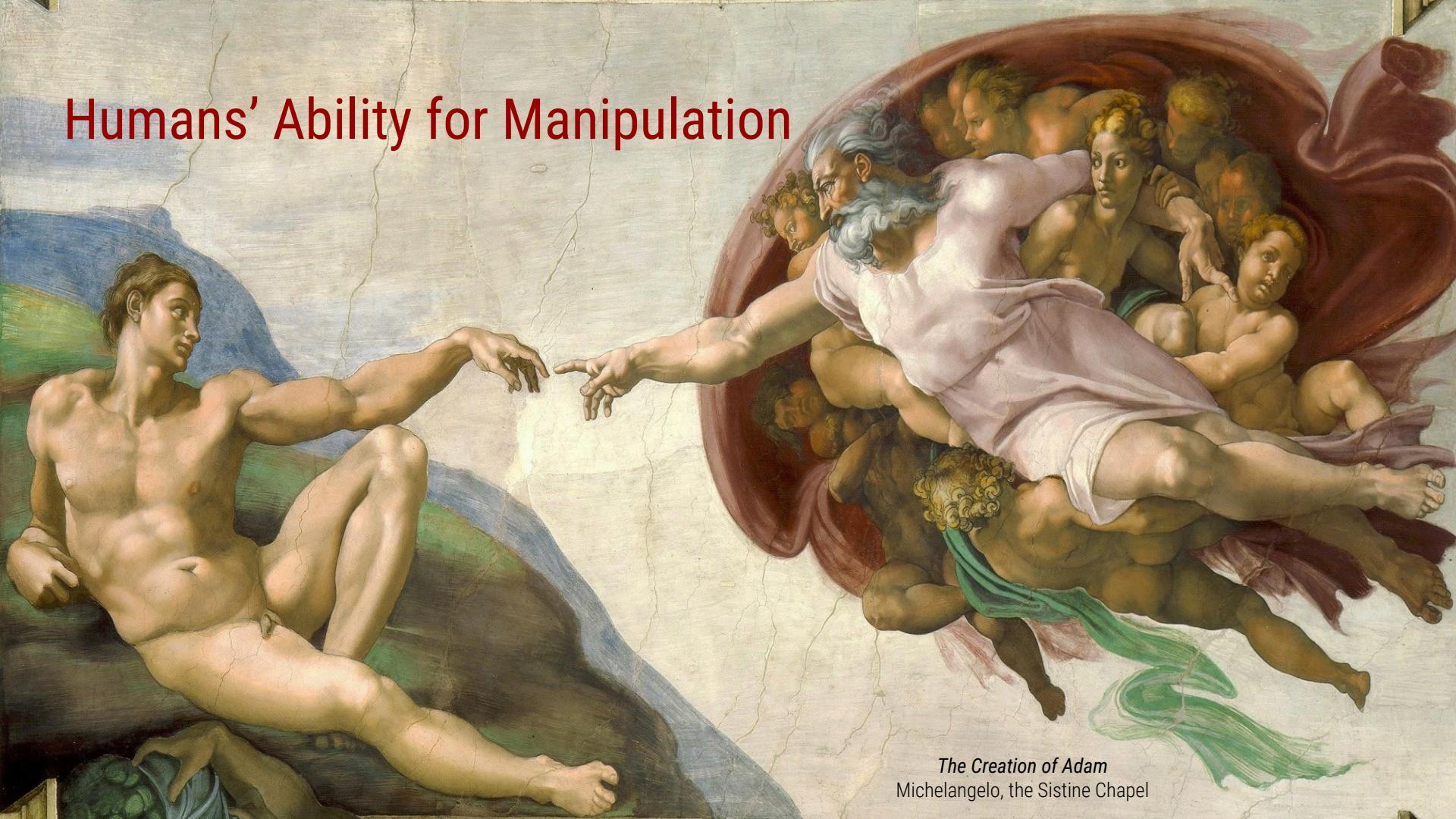
Grace Eysenbach, Nao Ouyang,
Maggie Wang, Michael Emanuel

Outline

1. Background and Motivation
2. Learning Hand-Eye Coordination for Robot Grasping with DL and Large-Scale Data Collection (Google)
3. Learning Dexterous In-Hand Manipulation (OpenAI)
4. Contributions, Context, Limitations, Future Work
5. Questions

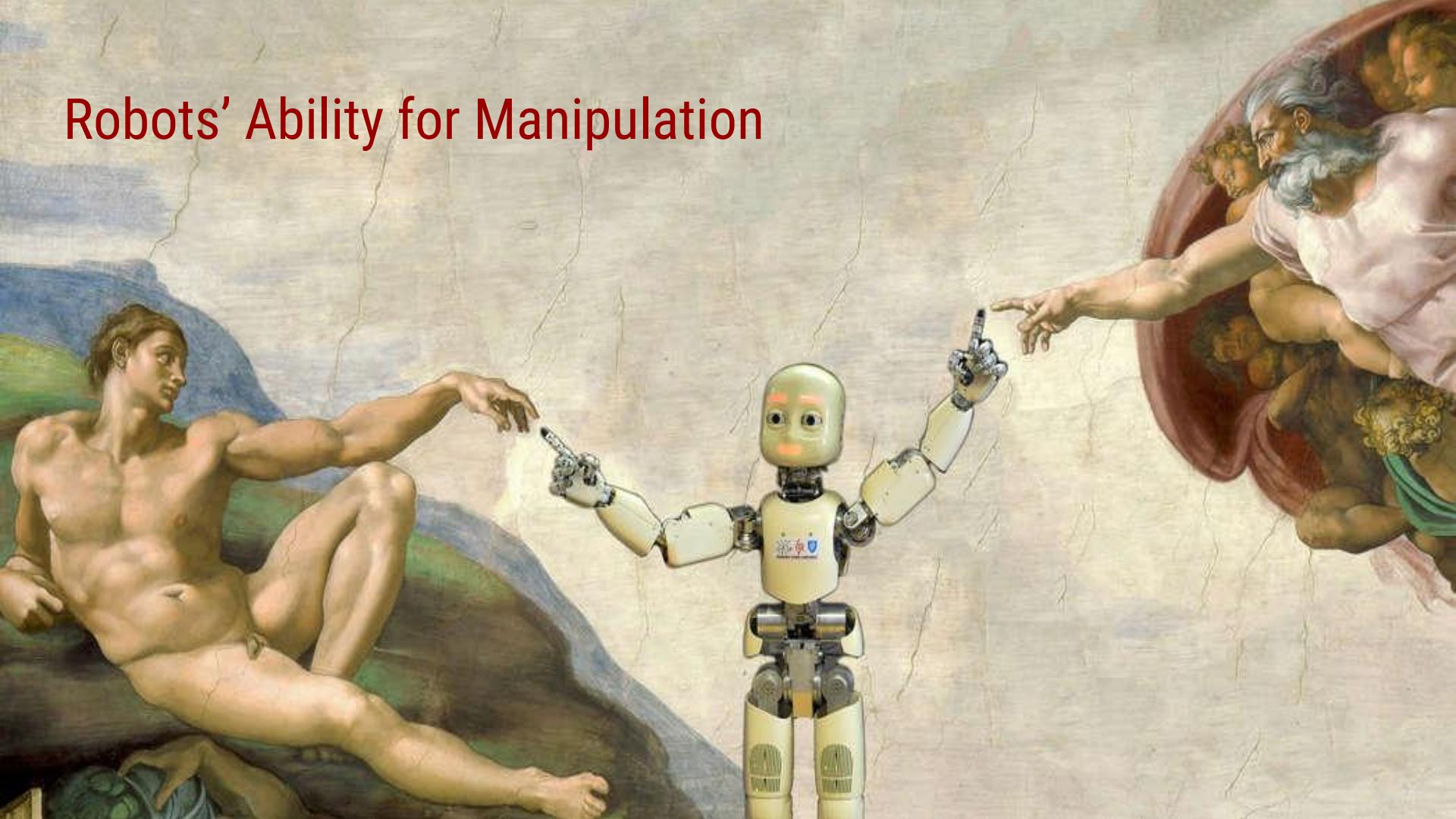
Motivation

Humans' Ability for Manipulation



The Creation of Adam
Michelangelo, the Sistine Chapel

Robots' Ability for Manipulation



The Robot Grasping Problem

- Why is this so hard? **Huge control space!**
- The problem is easy to state, hard to get right:
- We are still very far from robots that can perform even routine household dexterous tasks, e.g. laundry folding

The Robot Grasping Problem

- Why is this so hard? **Huge control space!**
- The problem is easy to state, hard to get right:
- We are still very far from robots that can perform even routine household dexterous tasks, e.g. laundry folding

The Robot Grasping Problem

- Why is this so hard? Huge control space!
- The problem is easy to state, hard to get right:
- We are still very far from robots that can perform even routine household dexterous tasks, e.g. laundry folding



Foldimate laundry folding machine: \$1000, no hands, size of a microwave

Robotic Manipulation in the Real World

- Industrial robots have impressive achievements
- But they are **very specialized** and work only in highly **controlled** environments
- Humans can handle a far greater range of tasks
- Applications:
 - **Warehouse robots**
 - **Package delivery**
 - **Hospitals:** assisting nursing staff with bed linens
 - **Home:** Roomba is the world's top selling robot



Industrial robots for automotive assembly



Warehouse robots for moving boxes onto pallets

Robotic Manipulation in the Real World

- Industrial robots have impressive achievements
- But they are **very specialized** and work only in highly **controlled** environments
- Humans can handle a far greater range of tasks
- Applications:
 - **Warehouse robots**
 - **Package delivery**
 - **Hospitals:** assisting nursing staff with bed linens
 - **Home:** Roomba is the world's top selling robot



Industrial robots for automotive assembly



Warehouse robots for moving boxes onto pallets

Robotic Manipulation in the Real World

- Industrial robots have impressive achievements
- But they are **very specialized** and work only in highly **controlled** environments
- Humans can handle a far greater range of tasks
- Applications:
 - **Warehouse robots**
 - **Package delivery**
 - **Hospitals:** assisting nursing staff with bed linens
 - **Home:** Roomba is the world's top selling robot



Industrial robots for automotive assembly



Warehouse robots for moving boxes onto pallets

Robotic Manipulation in the Real World

- Industrial robots have impressive achievements
- But they are **very specialized** and work only in highly **controlled** environments
- Humans can handle a far greater range of tasks
- Applications:
 - **Warehouse robots**
 - **Package delivery**
 - **Hospitals:** assisting nursing staff with bed linens
 - **Home:** Roomba is the world's top selling robot



Industrial robots for automotive assembly



Warehouse robots for moving boxes onto pallets

Making Robots Generalizable

- A major goal is to build robots with flexible, **generalizable** grasping + manipulation skills
- Challenging technical aspects of expanding out of controlled industrial environment
 - Huge control space
 - Huge variety of objects in the world
 - Huge variety of domains

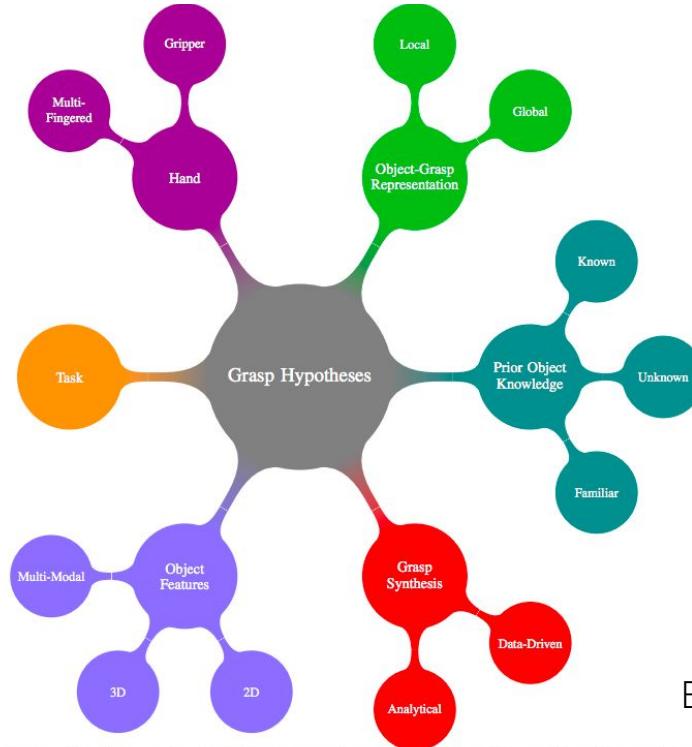
Making Robots Generalizable

- A major goal is to build robots with flexible, **generalizable** grasping + manipulation skills
- Challenging technical aspects of expanding out of controlled industrial environment
 - Huge control space
 - Huge variety of objects in the world
 - Huge variety of domains

Background:
Grasping and Manipulation

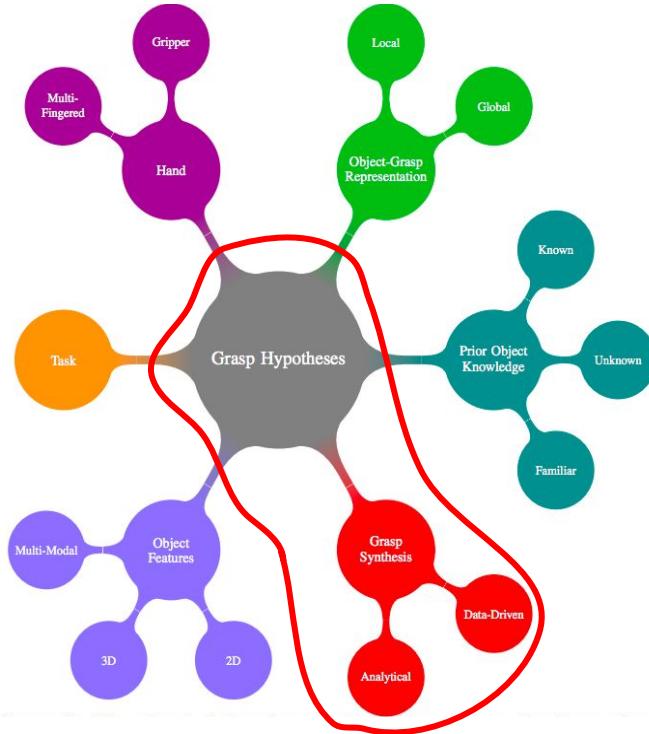
Grasping

Visualizing the Robotics Grasping Challenge



Bohg et al., 2014

Grasp Synthesis: Analytical vs. Data-Driven



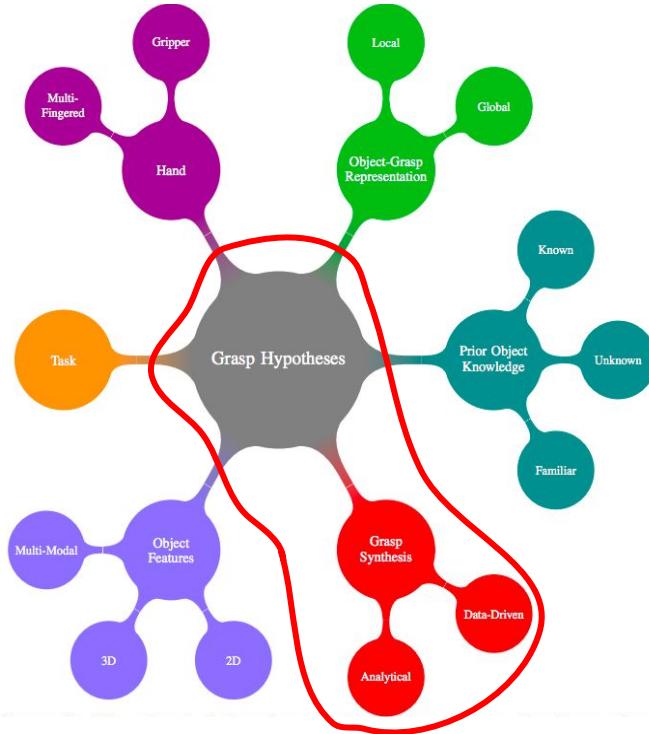
- **Analytical** → Creating kinematic and dynamic models to apply to grasping

(+) Good for tasks that where object positions and environments are predictable

(-) Inaccurate models + noisy sensors leads to lots of positioning errors

Dominated the field until early 2000s

Grasp Synthesis: Analytical vs. Data-Driven



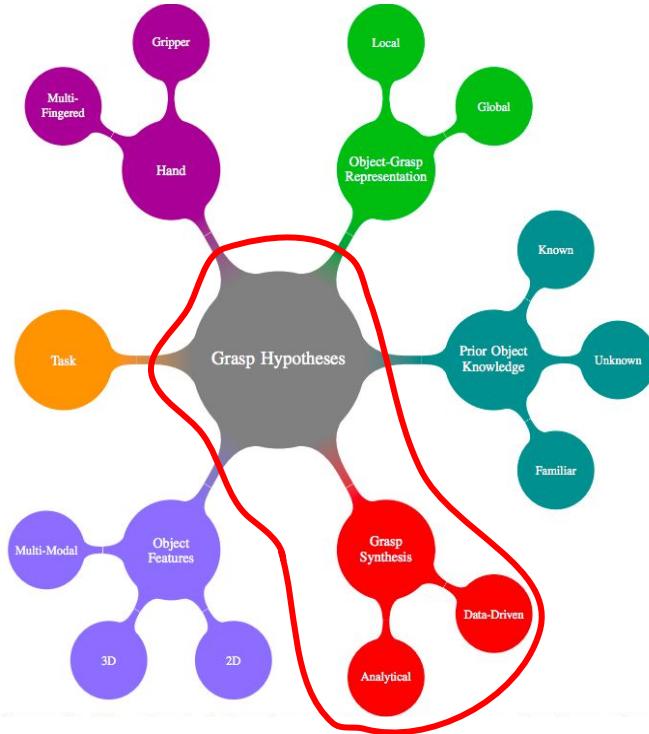
- **Analytical** → Creating kinematic and dynamic models to apply to grasping

(+) Good for tasks that where object positions and environments are predictable

(-) Inaccurate models + noisy sensors leads to lots of positioning errors

Dominated the field until early 2000s

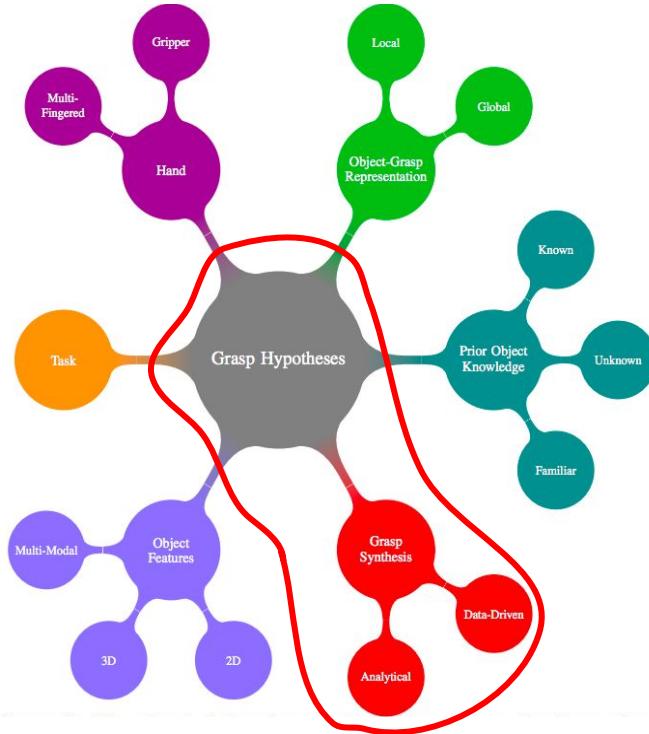
Grasp Synthesis: Analytical vs. Data-Driven



- Analytical → Creating kinematic and dynamic models to apply to grasping
 - (+) Good for tasks where object positions and environments are predictable
 - (-) Inaccurate models + noisy sensors leads to lots of positioning errors

Dominated the field until early 2000s

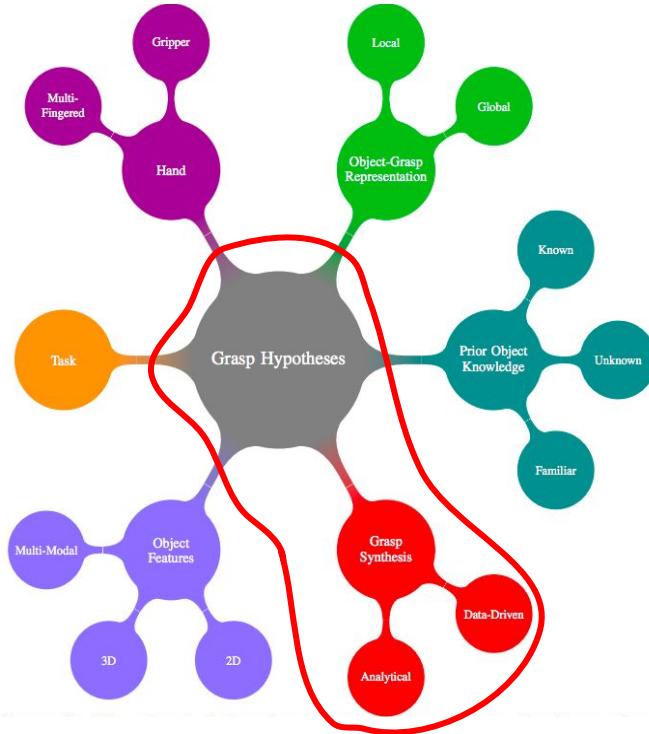
Grasp Synthesis: Analytical vs. Data-Driven



- Analytical → Creating kinematic and dynamic models to apply to grasping
 - (+) Good for tasks where object positions and environments are predictable
 - (-) Inaccurate models + noisy sensors leads to lots of positioning errors

Dominated the field until early 2000s

Grasp Synthesis: Analytical vs. Data-Driven

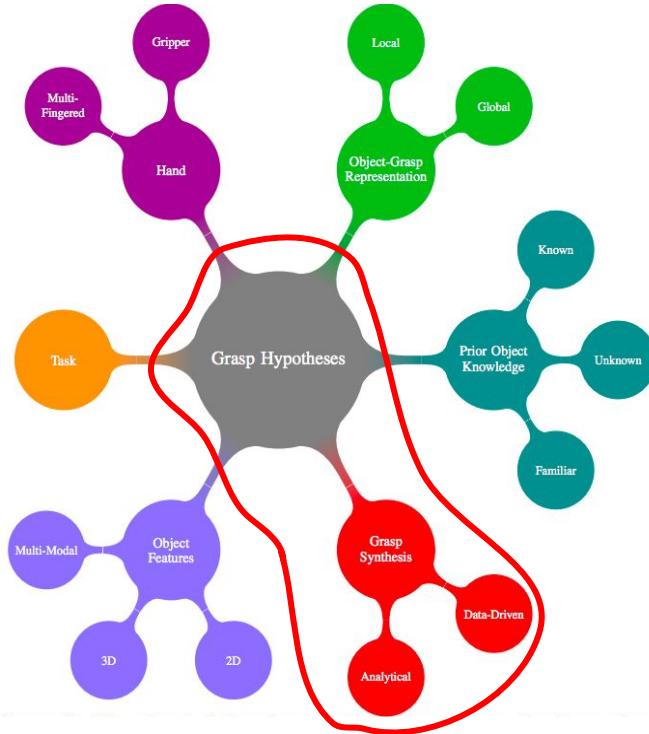


- **Data-Driven** → relies on sampling grasp candidates for an object and ranking them according to a specific metric using existing grasp experience

(+) Much larger range of potential applications

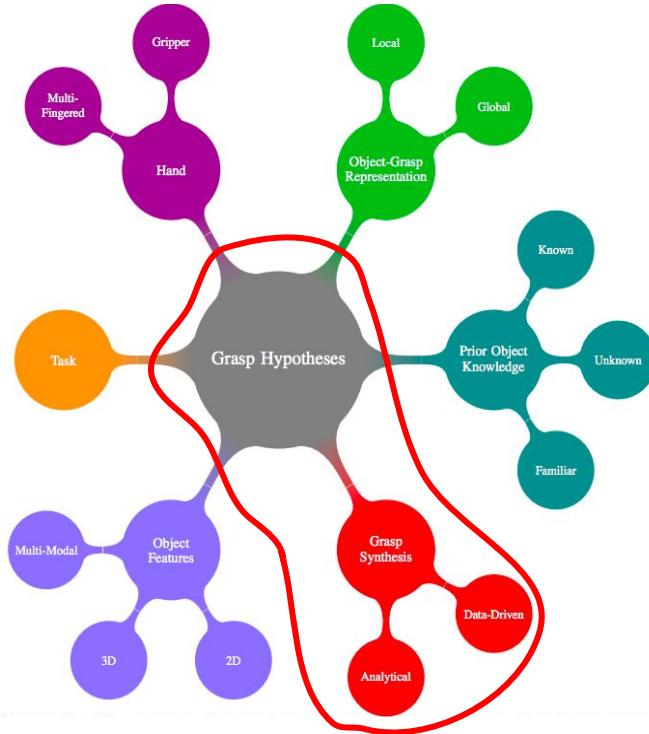
(-) Challenges of data collection + representation

Grasp Synthesis: Analytical vs. Data-Driven



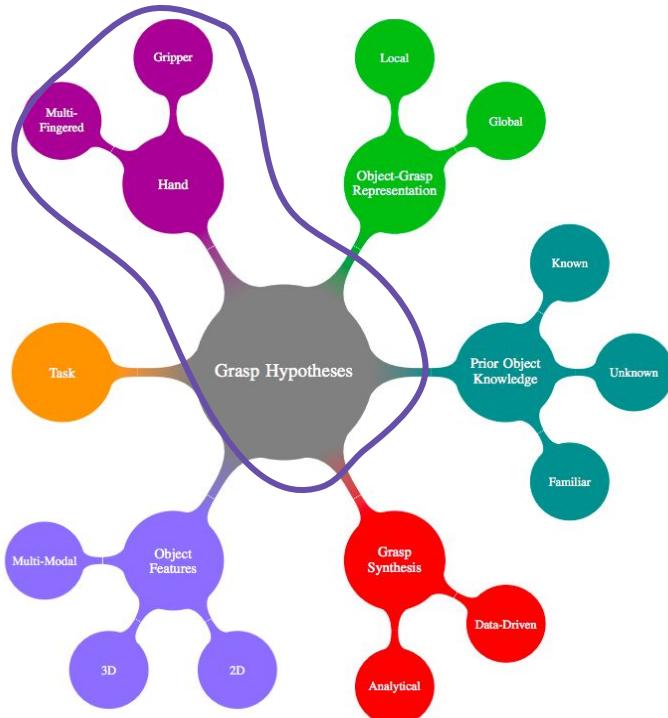
- **Data-Driven** → relies on sampling grasp candidates for an object and ranking them according to a specific metric using existing grasp experience
 - (+) Much larger range of potential applications
 - (-) Challenges of data collection + representation

Grasp Synthesis: Analytical vs. Data-Driven



- **Data-Driven** → relies on sampling grasp candidates for an object and ranking them according to a specific metric using existing grasp experience
 - (+) Much larger range of potential applications
 - (-) Challenges of data collection + representation

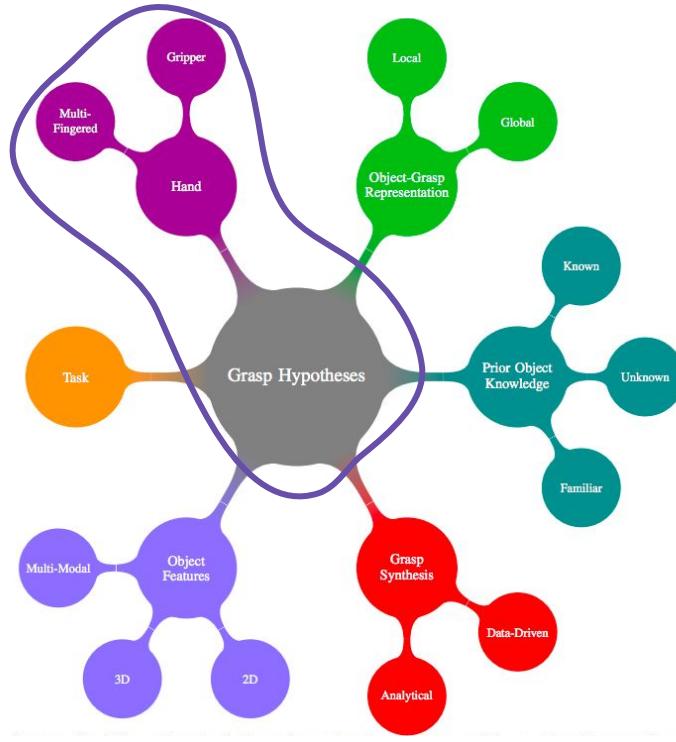
Means of Grasping: Gripper vs. Multi-Fingered



- Gripper → “Learning Hand-Eye Coordination for Robotic Grasping”



Means of Grasping: Gripper vs. Multi-Fingered



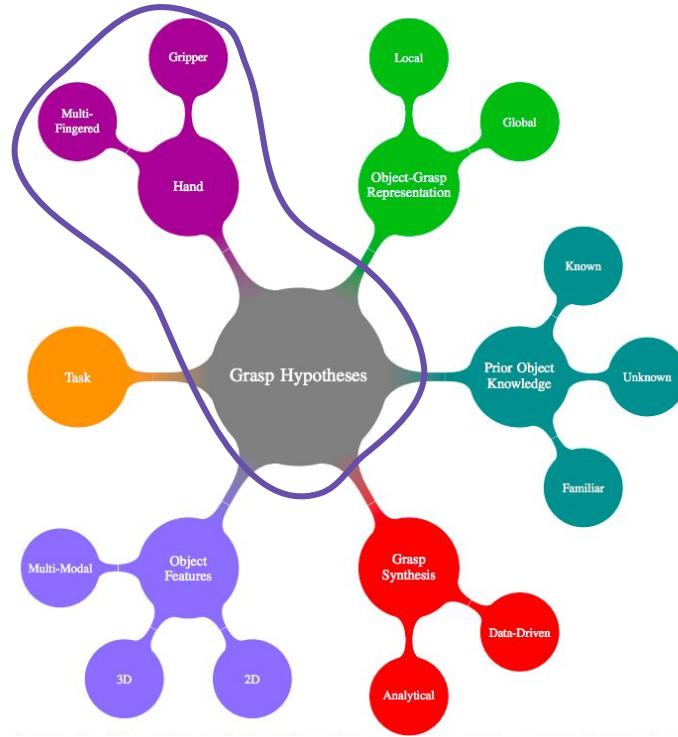
- **Gripper** → “Learning Hand-Eye Coordination for Robotic Grasping”



- **Multi-Fingered** → “Learning Dexterous In-Hand Manipulation”



Means of Grasping: Gripper vs. Multi-Fingered



- **Gripper** → “Learning Hand-Eye Coordination for Robotic Grasping”

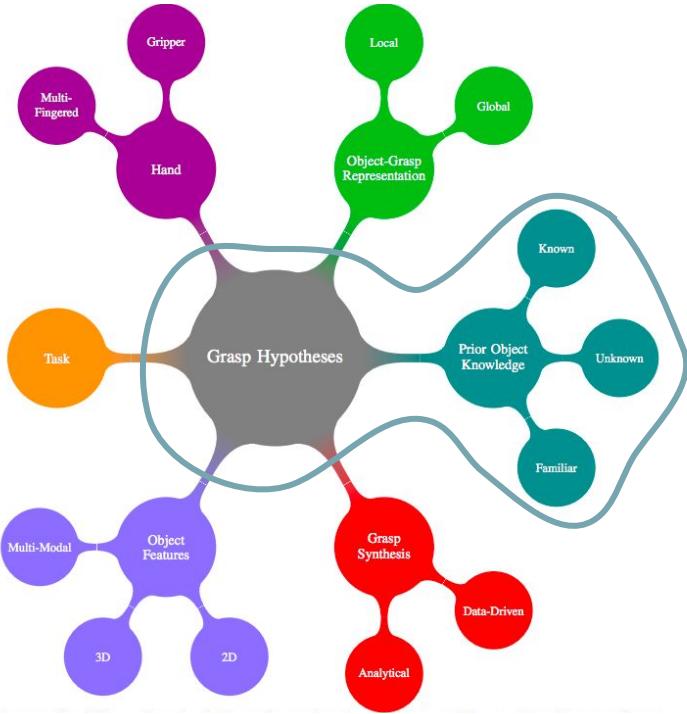


- **Multi-Fingered** → “Learning Dexterous In-Hand Manipulation”



Main Idea:
Higher DoF = Higher Complexity = Higher Possibilities

Prior Object Knowledge: Known vs. Familiar vs. Unknown



- **Known** → once an object is recognized, estimate its pose, and retrieve suitable grasp from existing experience database

(+) Good for specific tasks where you would expect to only see a set of objects

(-) Does not allow learning from new objects + environments

Prior Object Knowledge: Known vs. Familiar vs. Unknown

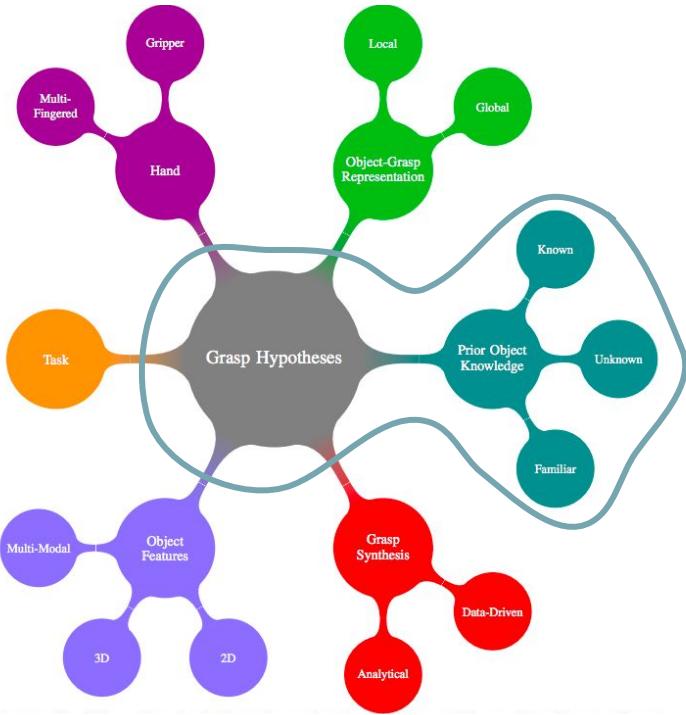


- **Known** → once an object is recognized, estimate its pose, and retrieve suitable grasp from existing experience database

(+) Good for specific tasks where you would expect to only see a set of objects

(-) Does not allow learning from new objects + environments

Prior Object Knowledge: Known vs. Familiar vs. Unknown

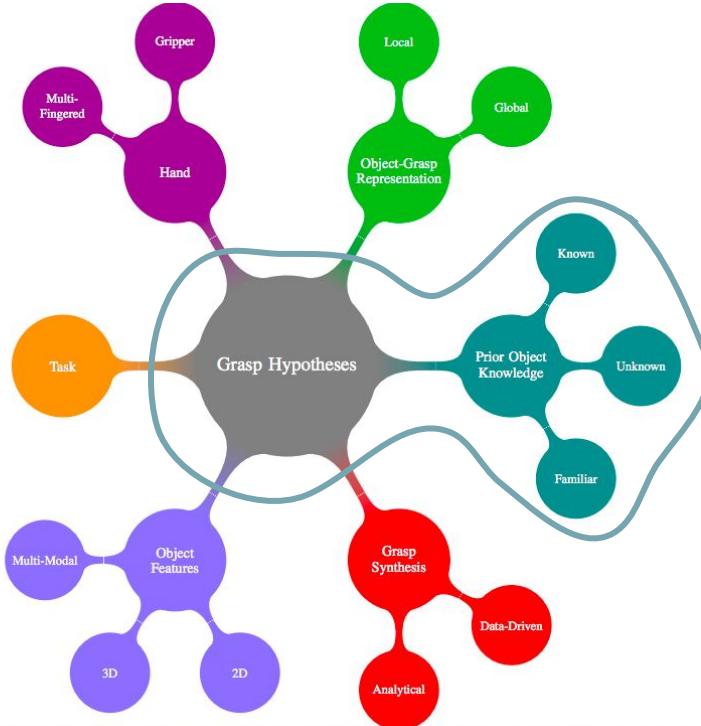


- **Known** → once an object is recognized, estimate its pose, and retrieve suitable grasp from existing experience database

(+) Good for specific tasks where you would expect to only see a set of objects

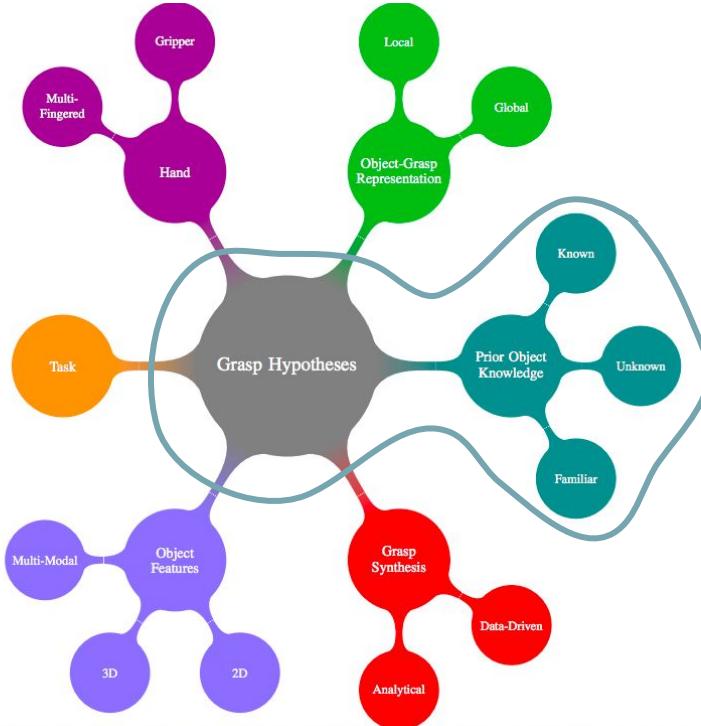
(-) Does not allow learning from new objects + environments

Prior Object Knowledge: Known vs. Familiar vs. Unknown



- Familiar → Plan grasp of object by identifying common characteristics with objects in the database
 - Low Level: texture, shape
 - High Level: functionality (pouring, rolling)
 - Develop a “similarity metric” to compare object + make grasps transferable

Prior Object Knowledge: Known vs. Familiar vs. Unknown



- Familiar → Plan grasp of object by identifying common characteristics with objects in the database
 - Low Level: texture, shape
 - High Level: functionality (pouring, rolling)
 - Develop a “similarity metric” to compare object + make grasps transferable

Similar Object Approaches

Using ANN to distinguish between graspable
and non-graspable parts of object
(El-Koury and Sahbani, 2008)

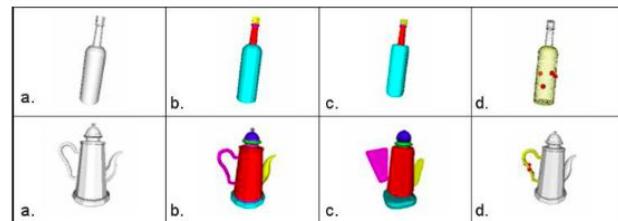


Figure 10: a) Object model. b) Part segmentation. c) SQ approximation. d) Graspable part and contact points [89].

F

Similar Object Approaches

Using ANN to distinguish between graspable and non-graspable parts of object
(El-Koury and Sahbani, 2008)

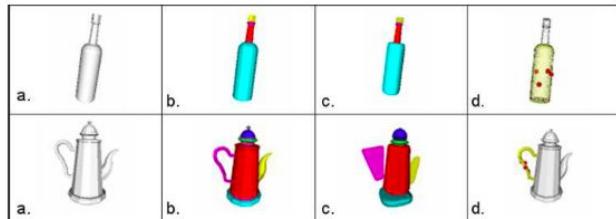


Figure 10: a) Object model. b) Part segmentation. c) SQ approximation. d) Graspable part and contact points [89].

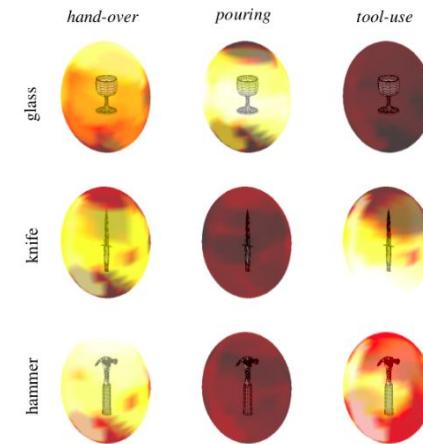
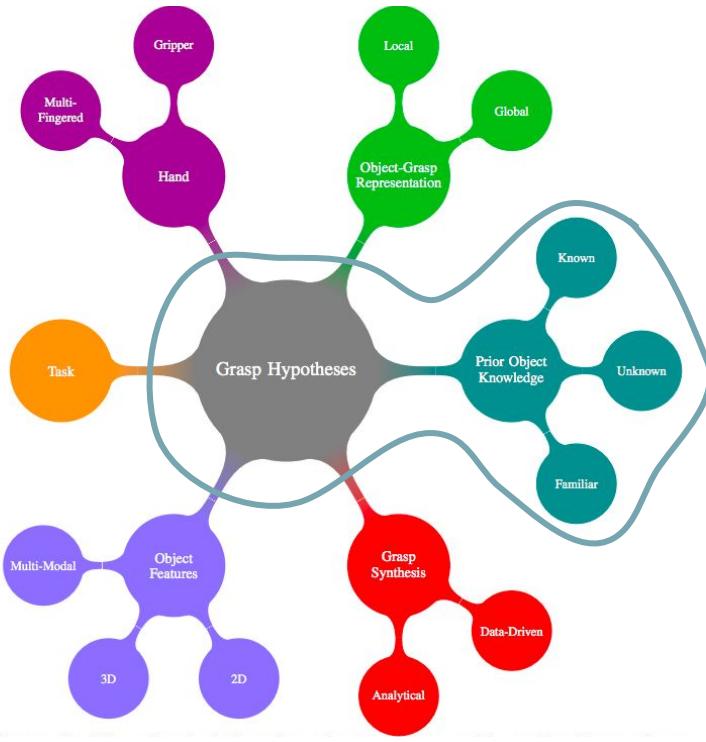


Figure 20: Ranking of approach vectors for different objects given a specific task. The brighter an area the higher the rank. The darker an area, the lower the rank [87].

Generating model for grasping using Bayesian learning
(Song et al., 2011)

Prior Object Knowledge: Known vs. Familiar vs. Unknown



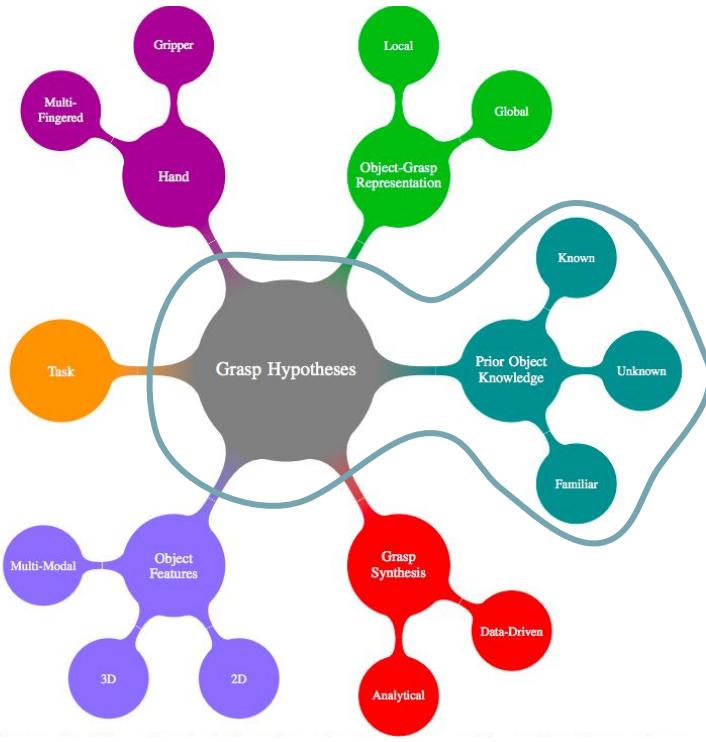
- **Unknown** → Assume no access to model of OR grasp experience with the object.

Method: Combine sensory data with heuristics to analyze candidate grasps

(+) Allows most applicability to real world scenarios

(-) Many challenges in trying to estimate object / grasps using sparse, noisy, or incomplete sensor data

Prior Object Knowledge: Known vs. Familiar vs. Unknown



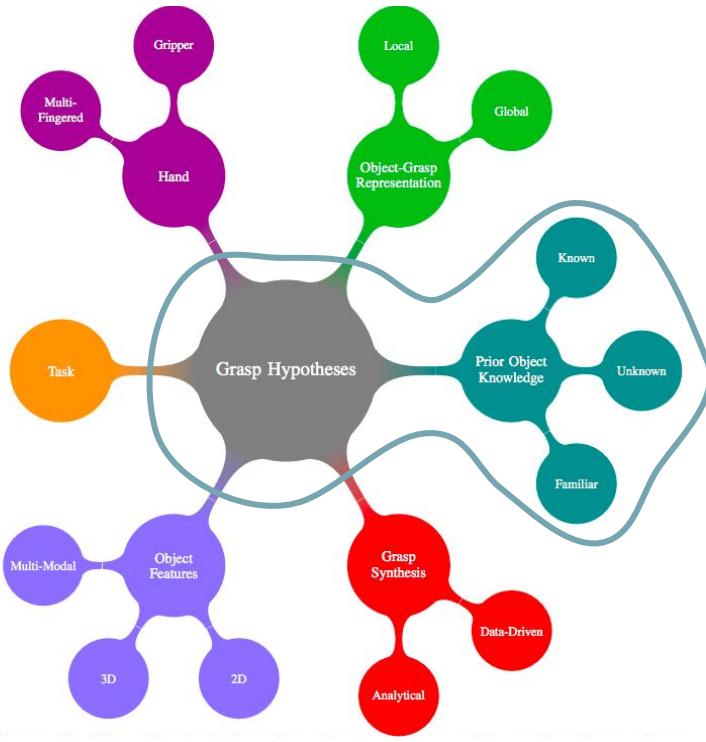
- **Unknown** → Assume no access to model of OR grasp experience with the object.

Method: Combine sensory data with heuristics to analyze candidate grasps

(+) Allows most applicability to real world scenarios

(-) Many challenges in trying to estimate object / grasps using sparse, noisy, or incomplete sensor data

Prior Object Knowledge: Known vs. Familiar vs. Unknown



- **Unknown** → Assume no access to model of OR grasp experience with the object.

Method: Combine sensory data with heuristics to analyze candidate grasps

- (+) Allows most applicability to real world scenarios
- (-) Many challenges in trying to estimate object / grasps using sparse, noisy, or incomplete sensor data

Unknown Object Approaches

Approximating unknown object shape using symmetry
(Bohg et al., 2011)

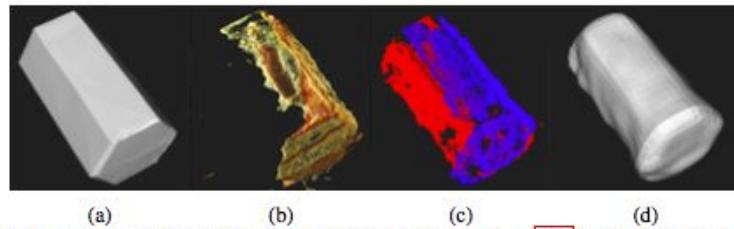


Figure 22: Estimated full object shape by assuming symmetry. [22a] Ground Truth Mesh.
[22b] Original Point Cloud. [22c] Mirrored Cloud with Original Points in Blue and
Additional Points in Red. [22d] Reconstructed Mesh [120].

Unknown Object Approaches

Approximating unknown object shape using symmetry
(Bohg et al., 2011)

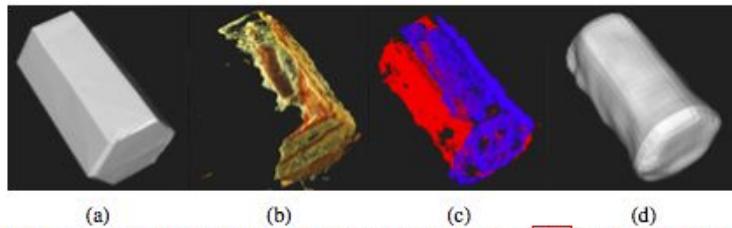
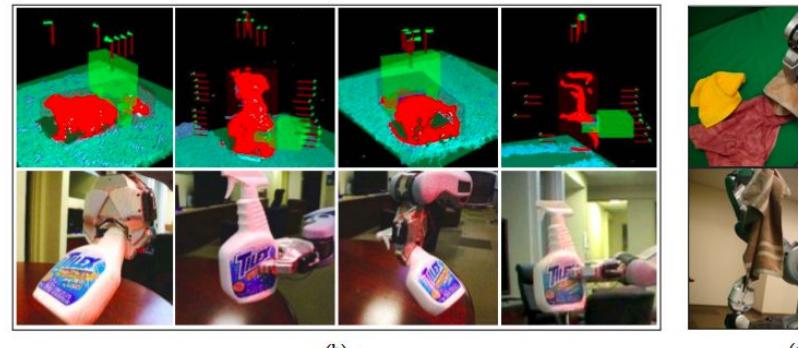
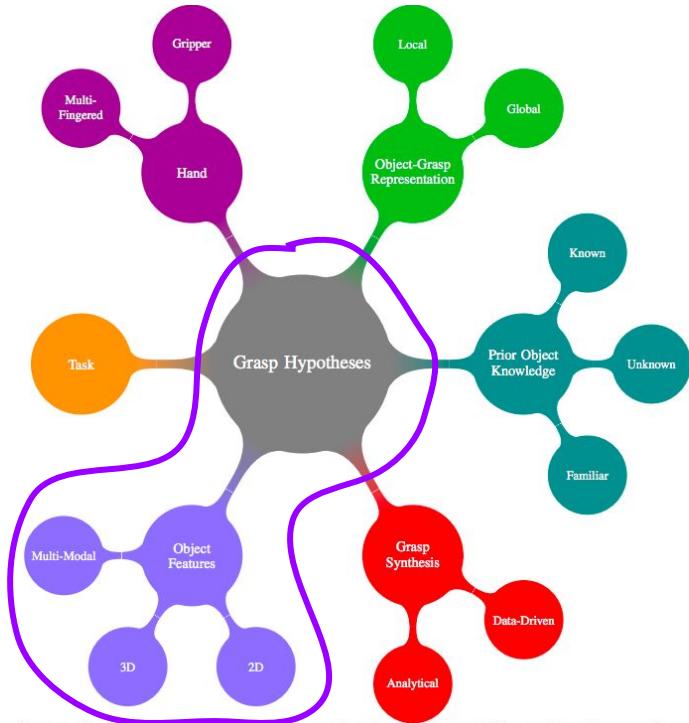


Figure 22: Estimated full object shape by assuming symmetry. [22a] Ground Truth Mesh.
[22b] Original Point Cloud. [22c] Mirrored Cloud with Original Points in Blue and
Additional Points in Red. [22d] Reconstructed Mesh [120].

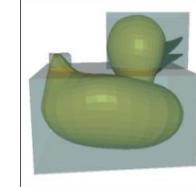


Generating grasp hypotheses from segmented point cloud
+ ranking by considering specific features of the grasp
(Hsiao et al., 2010)

Object Features: 3D vs. 2D vs. Multi-Modal



- 3D



Box Decomposition
(Hubner et al., 2008)

- 2D

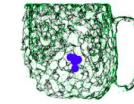
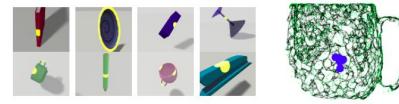
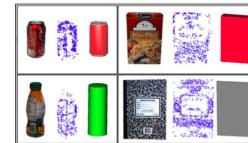


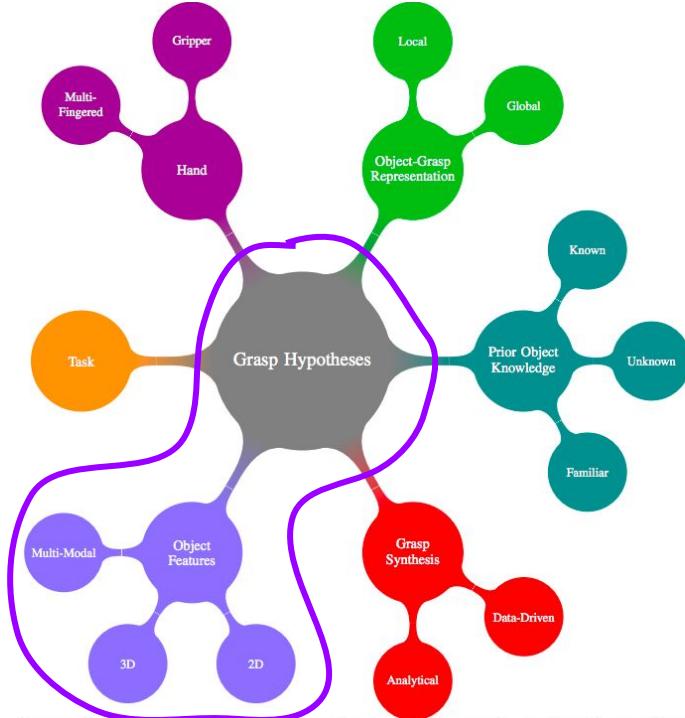
Figure 12: Labeled training data. (a) One example for each of the eight object classes in training data in [28] along with their grasp labels (in yellow). (b) Positive (red) and negative examples (blue) for grasping points [94].

- Multi-Modal → Combine 3D and 2D

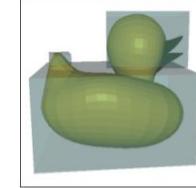


Multiview Object Representation
(Detry et al, 2009)

Object Features: 3D vs. 2D vs. Multi-Modal



- 3D



Box Decomposition
(Hubner et al., 2008)

- 2D

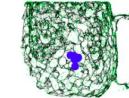
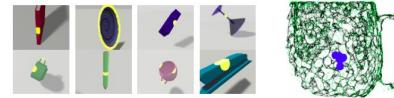
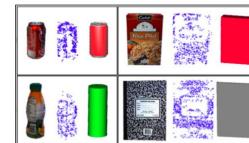


Figure 12: Labeled training data. (a) One example for each of the eight object classes in training data in [28] along with their grasp labels (in yellow). (b) Positive (red) and negative examples (blue) for grasping points [94].

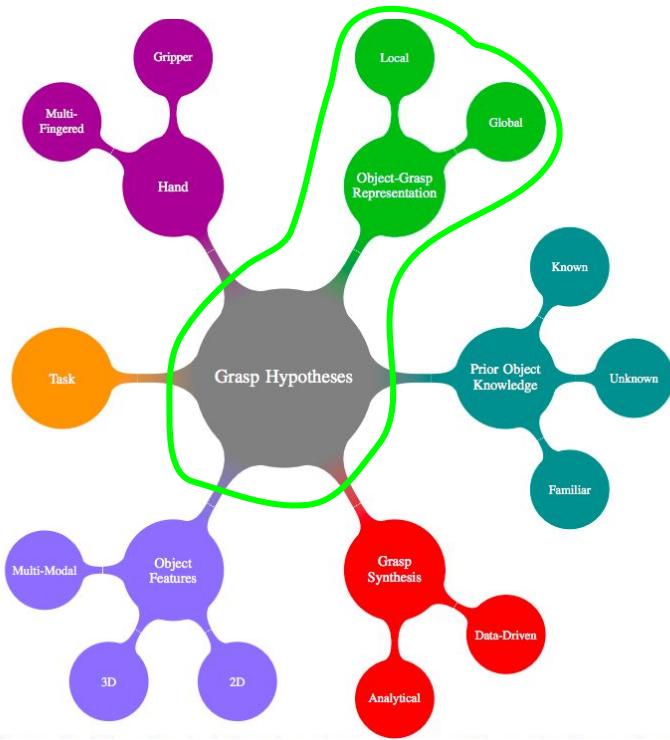
- Multi-Modal → Combine 3D and 2D



Multiview Object Representation
(Detry et al, 2009)

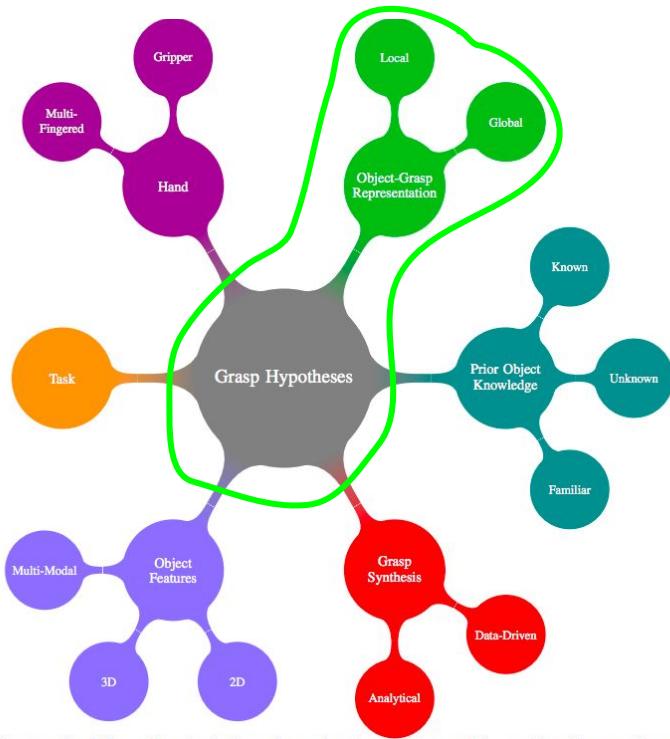
Key Concern: Costs of Complexity

Object Grasp Representation: Local vs. Global



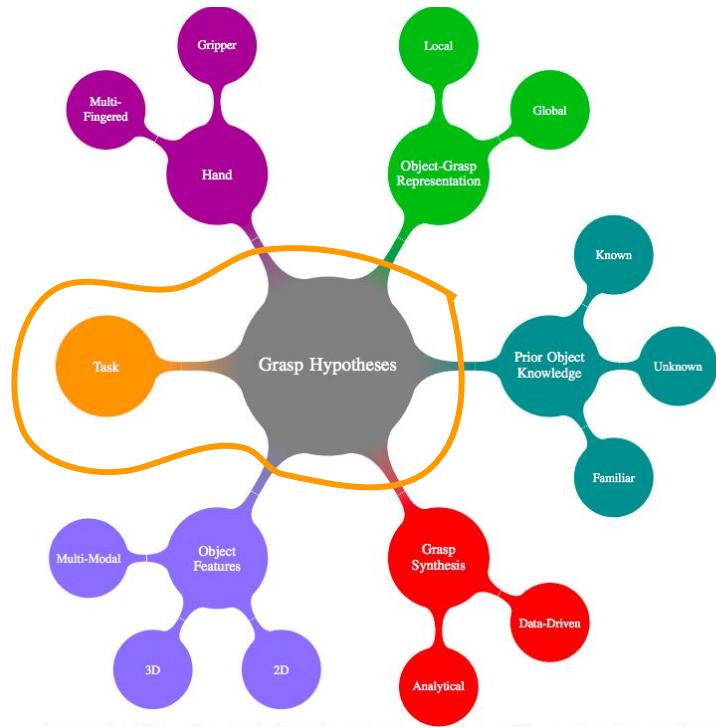
- **Local** → examining low level features (ex. Contour descriptors)
- **Global** → examining shape of object as a whole (ex. Object outline)

Object Grasp Representation: Local vs. Global



- **Local** → examining low level features (ex. Contour descriptors)
- **Global** → examining shape of object as a whole (ex. Object outline)

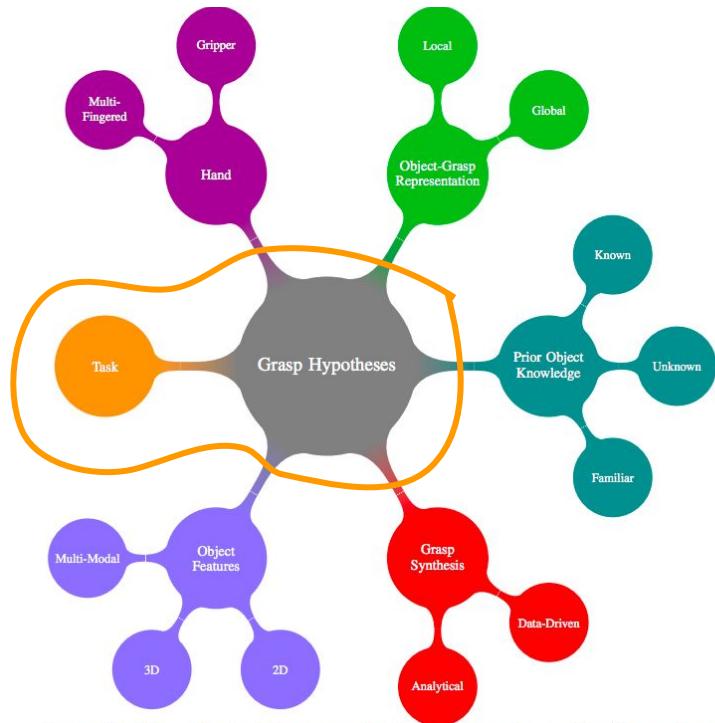
Task: What do we want to do?



Main Idea: Understanding our task is central to making choices about all these other categories related to grasping!

- Grasp database storage constraints?
- Objects it will encounter?
- Sensors on object?
- Etc.

Task: What do we want to do?



Main Idea: Understanding our task is central to making choices about all these other categories related to grasping!

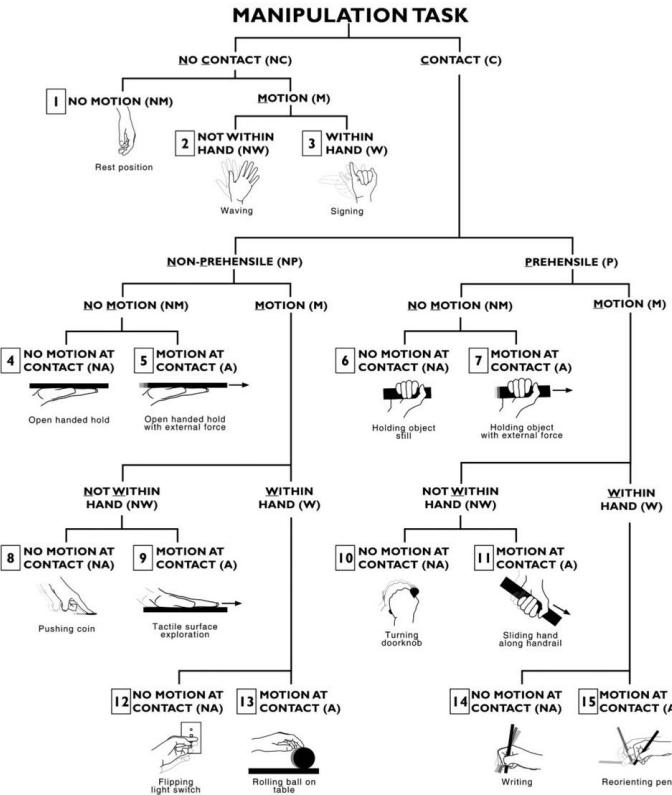
- Grasp database storage constraints?
- Objects it will encounter?
- Sensors on object?
- Etc.

Manipulation

Defining Manipulation

Manipulation: Refers to an agent's control of its environment through selective contact (Mason, 2018)

Fun Fact: Human's fabrication of hand tools is seen a key influence in our development of fine manipulation skills (Mason, 2018)



Bullock et al. 2013

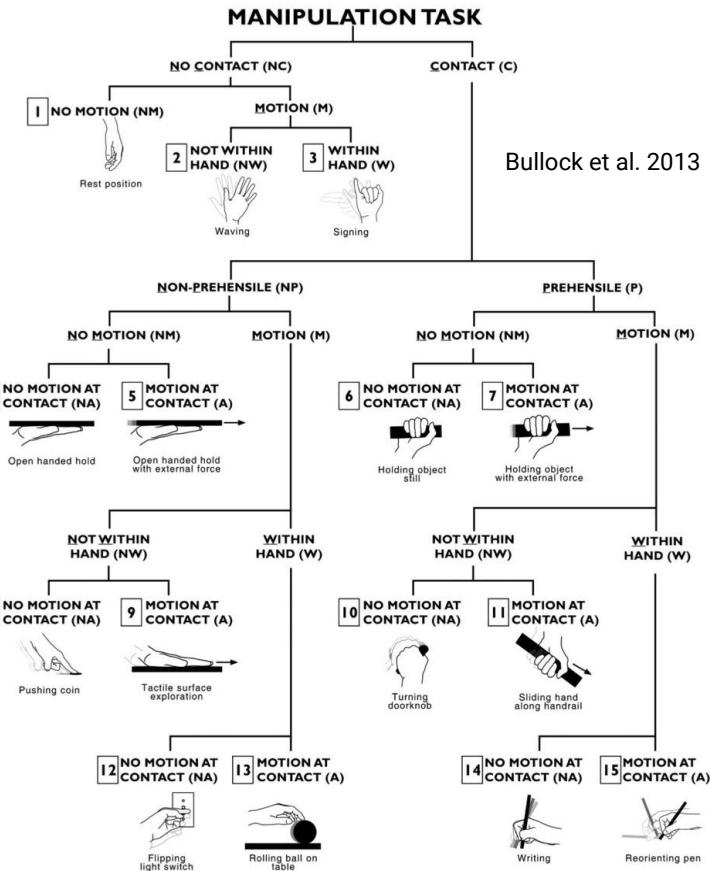
Types of Manipulation

1. Pick-and-Place Manipulation
2. In-Hand Manipulation



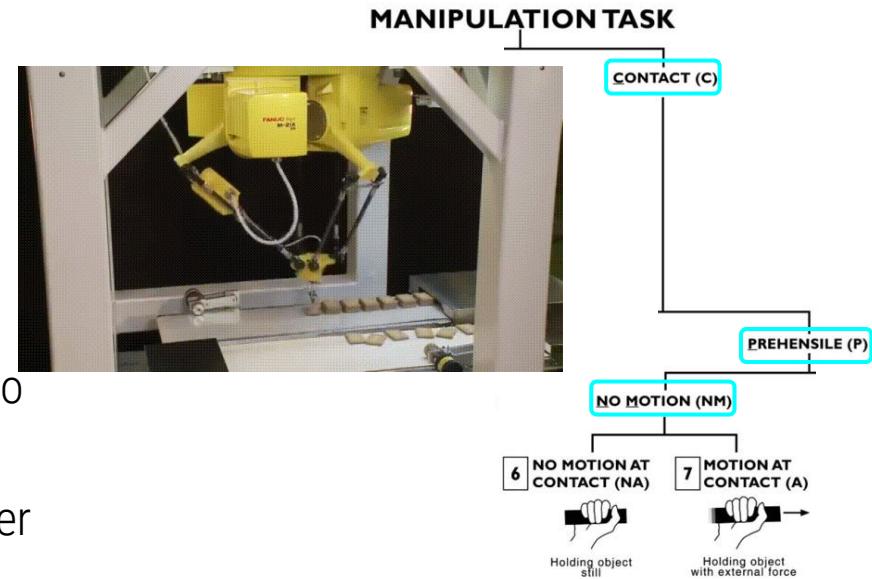
Pick-and-Place Manipulation

- **Definition:** Tasks that involve moving a sequence of objects one at a time from one place to another (Mason, 2018)



Pick-and-Place Manipulation

- **Definition:** Tasks that involve moving a sequence of objects one at a time from one place to another (Mason, 2018)
1. Structured Pick-and-Place
 - Structured Environment = Engineered to simplify a task -- arranging objects in a predictable position + eliminating clutter



Pick-and-Place Manipulation

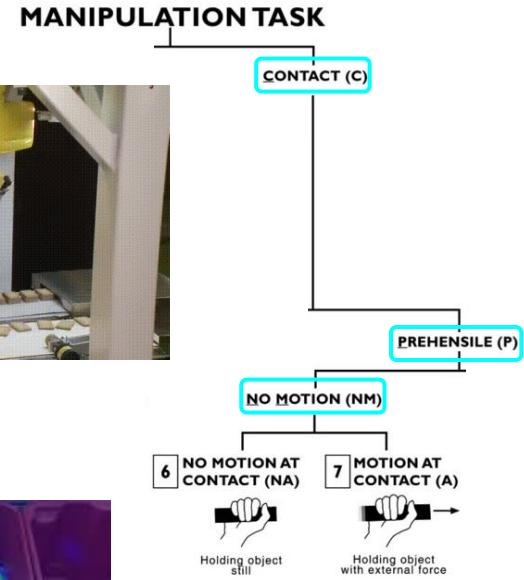
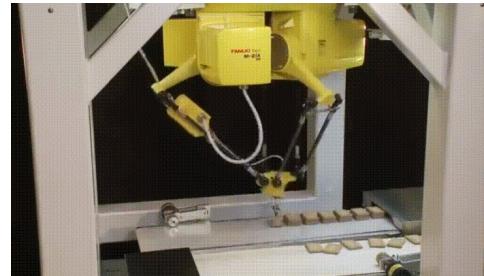
- **Definition:** Tasks that involve moving a sequence of objects one at a time from one place to another (Mason, 2018)

1. Structured Pick-and-Place

- Structured Environment = Engineered to simplify a task -- arranging objects in a predictable position + eliminating clutter

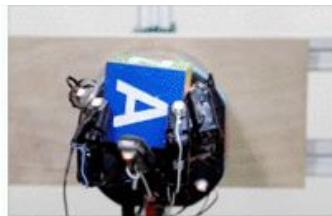
2. Unstructured Pick-and-Place

- Path Planning
- General-Purpose Grippers
- Grasp + Placement Pose Planning

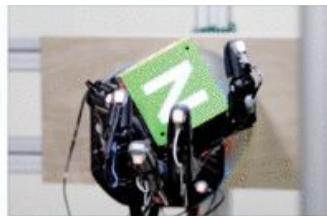


In-Hand Manipulation

- Dexterous In-Hand Manipulation: the skillful turning and shifting objects in the hand through the motion of the palm and fingers



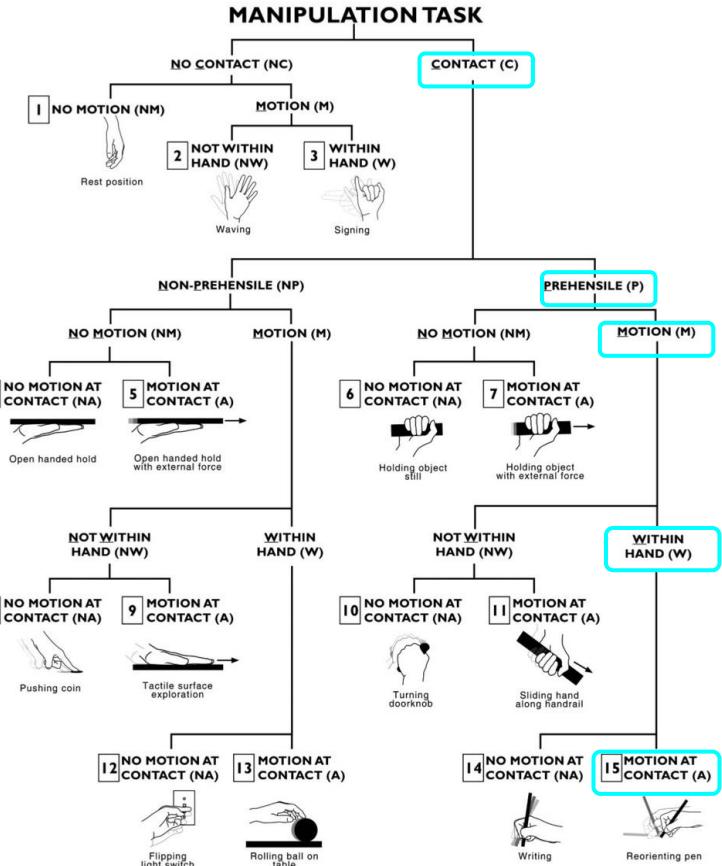
FINGER PIVOTING



SLIDING



FINGER GAITING



Grasping + Manipulation: Relating to the Two Papers

Overview of Two Papers Presented

1. Learning Hand-Eye Coordination for Robotic Grasping

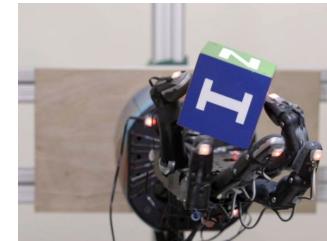
- Published by **Google** in **2016**
- They trained a robot arm with a **2 fingered gripper** to pick up objects using only images from a single camera



Learning Hand-Eye Coordination for Robotic Grasping:
Experimental setup

2. Learning Dexterous In-Hand Manipulation

- Published by **OpenAI** in **2018**
- Trained a **Shadow Dexterous Hand** robot to manipulate a block to a desired configuration



Learning Dexterous In-Hand Manipulation: Shadow Hand
holding cubic block

Two Key Differences:

1. Defining Manipulation
2. Training Method

1. Defining Manipulation: Key Challenge

The Regrasp

- **Pick and Place** → because the object is held rigidly, in order to regrasp, the object has to be put down and picked up with a new grasp
- **In-Hand** → Uses increased degrees of freedom to move object while maintaining fixed or rolling contact



1. Defining Manipulation: Key Difference

- The Google Paper → “Manipulation” = a single grasp/release per object
 - Unstructured Pick-And-Place



1. Defining Manipulation: Key Difference

- The Google Paper → “Manipulation” = a single grasp/release per object
 - Unstructured Pick-And-Place
- The Open AI Paper → “Manipulation” = changing grasps to rotate object
 - In-Hand Manipulation



2. Training Method: Key Challenge

The **Sim-to-Real Transfer** is a key consideration in deciding how robot will be trained

- Training on Simulation → “Reality gap”, **BUT**
allows massive amounts of training in limited time
- Training on Robot → No “Reality gap”, **BUT**
requires lots of time and expensive robotics hardware

2. Training Method: Key Challenge

The **Sim-to-Real Transfer** is a key consideration in deciding how robot will be trained

- **Training on Simulation** → “Reality gap”, **BUT**
allows massive amounts of training in limited time
- **Training on Robot** → No “Reality gap”, **BUT**
requires lots of time and expensive robotics hardware

EXPECTATION:



REALITY:



2. Training Method: Key Challenge

The **Sim-to-Real Transfer** is a key consideration in deciding how robot will be trained

- **Training on Simulation** → “Reality gap”, BUT allows massive amounts of training in limited time
- **Training on Robot** → No “Reality gap”, BUT requires lots of time and expensive robotics hardware



2. Training Method: Key Difference

- The Google Paper → Trained exclusively on data from 800,000 grasping attempts in the real world



- The OpenAI Paper → Trained on millions of simulator runs, then deployed on robot

2. Training Method: Key Difference

- The Google Paper → Trained exclusively on data from 800,000 grasping attempts in the real world



- The OpenAI Paper → Trained on millions of simulator runs, then deployed on robot



Common Thematic Elements

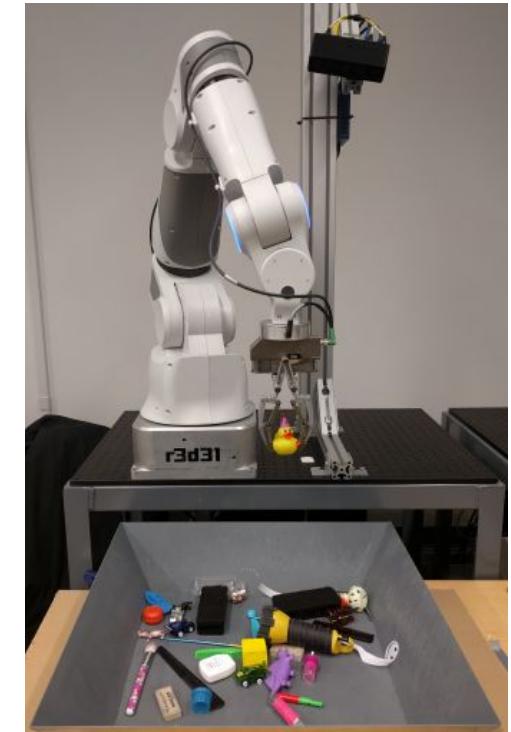
- Reinforcement Learning
 - Google used RL in the real world; OpenAI used it in the simulator
 - OpenAI used Generalized Advantage Estimator (GAE) and Proximal Policy Optimization (PPO)
 - Google trained their NN as a supervised learning problem (predict grasping success)
- Generalizability
 - OpenAI used domain randomization to improve generalizability
 - Google used a diverse range of object sizes, shapes and textures
- sim2real (or other transfer learning), sim2sim
 - Sim2real gap was significant in OpenAI, but not so big the robot couldn't perform well
 - Sim2sim arose when OpenAI trained robot to use vision only without ground truth position data

Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection

Levine et al. (Google 2016)

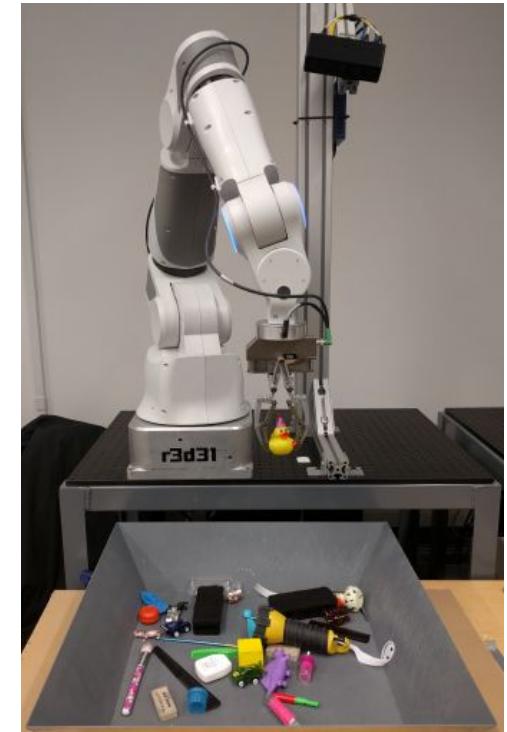
Main Contributions

- Physical system: 7 DOF robot arm, 2 finger gripper, and an uncalibrated monocular camera
1. Novel **CNN architecture** that predicts grasping success
 2. **Continuous visual servoing function** controls the robot
 3. **Large-scale data collection** framework for robotic grasps



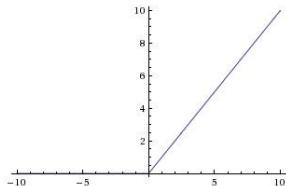
Main Contributions

- Physical system: 7 DOF robot arm, 2 finger gripper, and an uncalibrated monocular camera
- Novel **CNN architecture** that predicts grasping success
 - Continuous visual servoing function controls the robot
 - Large-scale data collection framework for robotic grasps

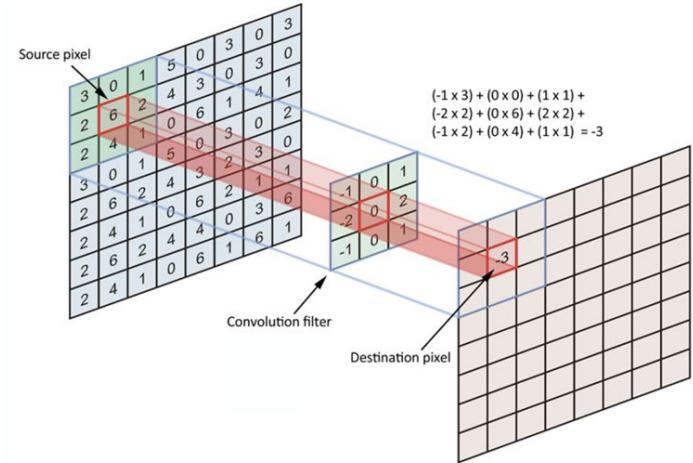


Convolutional Neural Network (CNN) Refresher

- Convolutional Layers
 - Define features in terms of a local filter
 - Have a size and stride length
- Max Pooling Layers
 - Summarize activity of a feature in an area
- ReLU activation
 - Activation function $a(\mathbf{x}) = \max(\mathbf{x}, 0)$



Images Credit: CS 109b Lecture Notes, Spring 2019



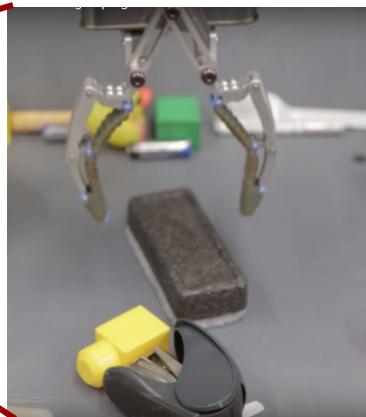
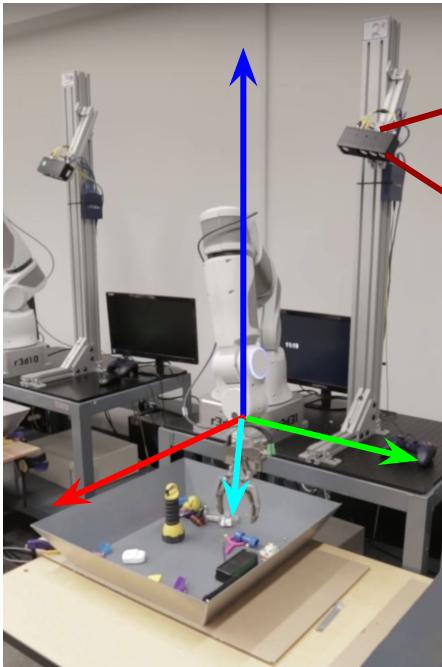
Forward mode, 3x3 stride 1

2	4	8	3	6
9	3	4	2	5
5	4	6	3	1
2	3	1	3	4
2	7	4	5	7



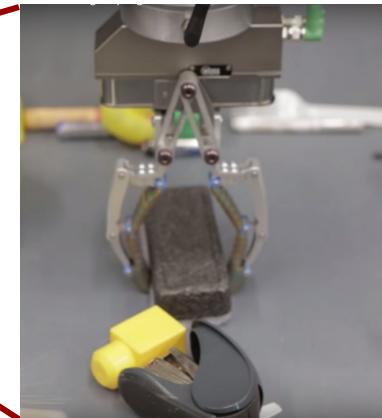
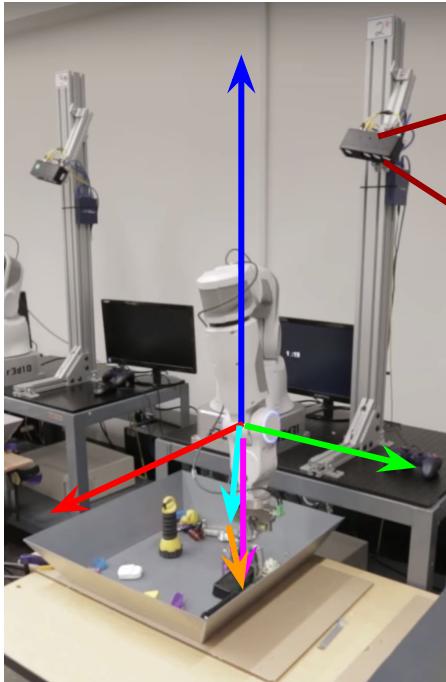
9		

Grasp Sample Setup: Current Time Step, t

 I_t^i

Grasp attempt i

Grasp Sample Setup: Final Time Step, T



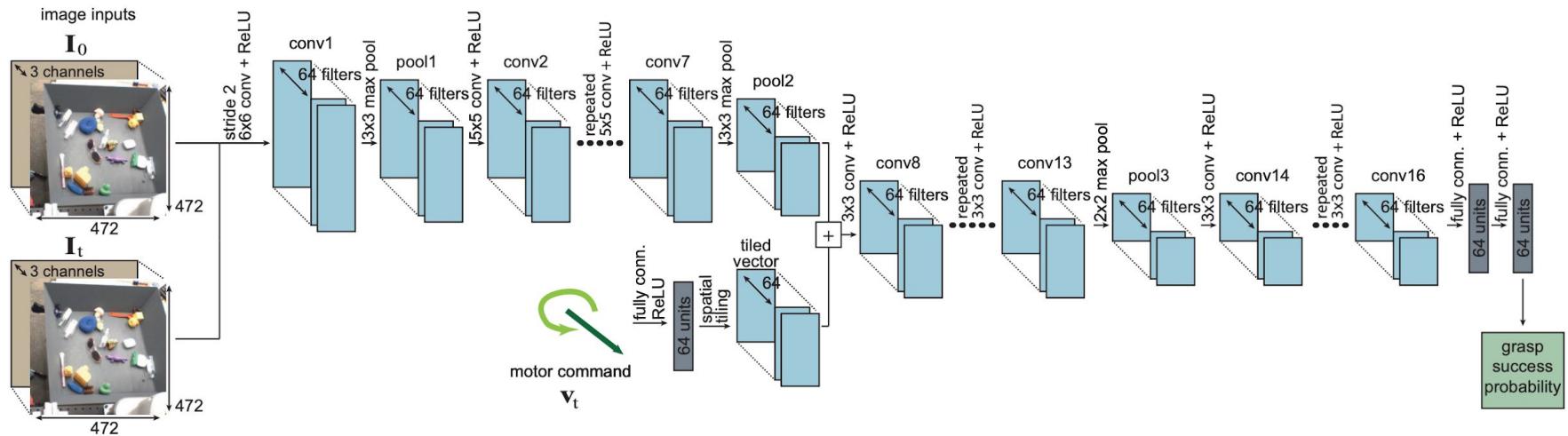
Current pose, \mathbf{p}_t^i

Final pose, \mathbf{p}_T^i

Task-space motion, \mathbf{v}_t^i

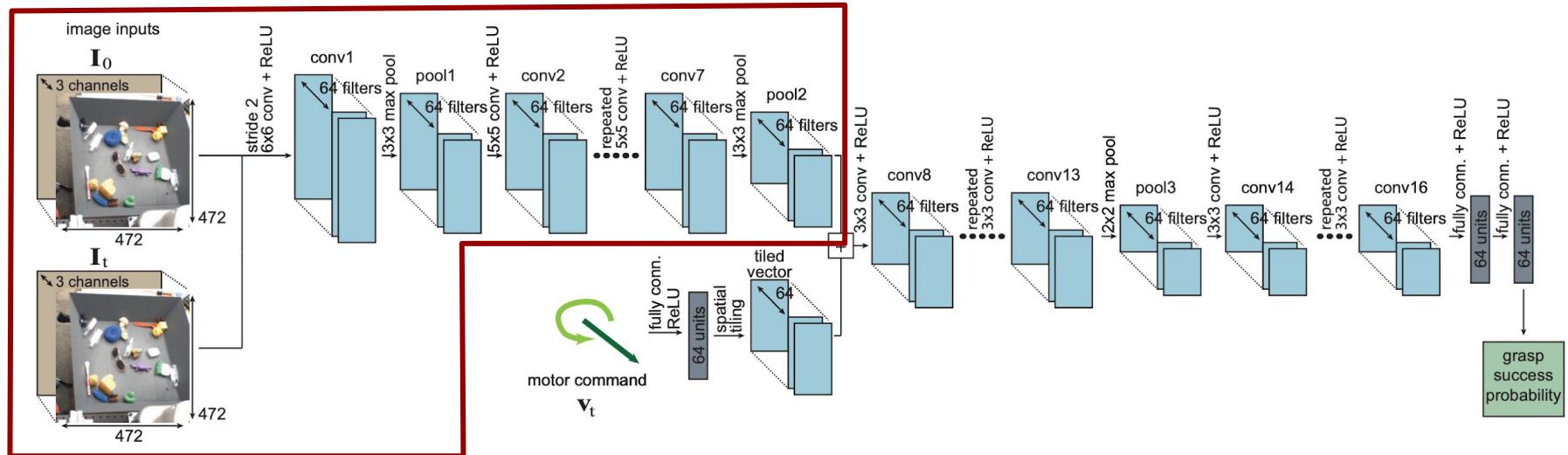
Label, \mathbf{l}_t^i
 $(\mathbf{l}_t^i, \mathbf{p}_T^i - \mathbf{p}_t^i, \mathbf{l}_t^i)$

Neural Network Architecture for $g(\mathbf{l}_t, \mathbf{v}_t)$



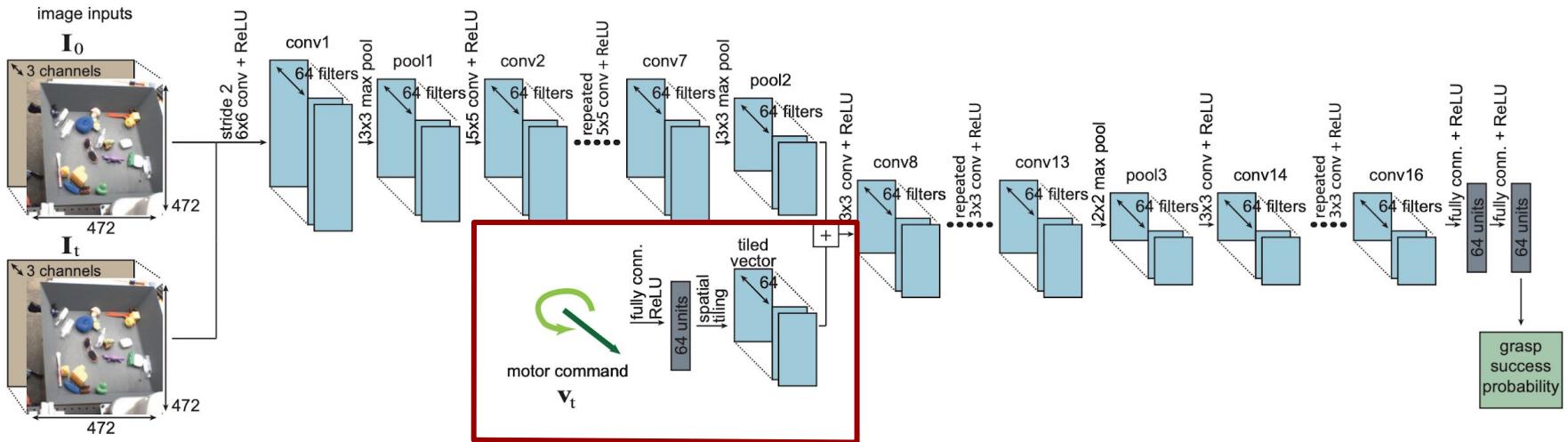
Network predicts the probability of success of a given motor command,
independent of the exact camera pose

Neural Network Architecture for $g(\mathbf{l}_t, \mathbf{v}_t)$



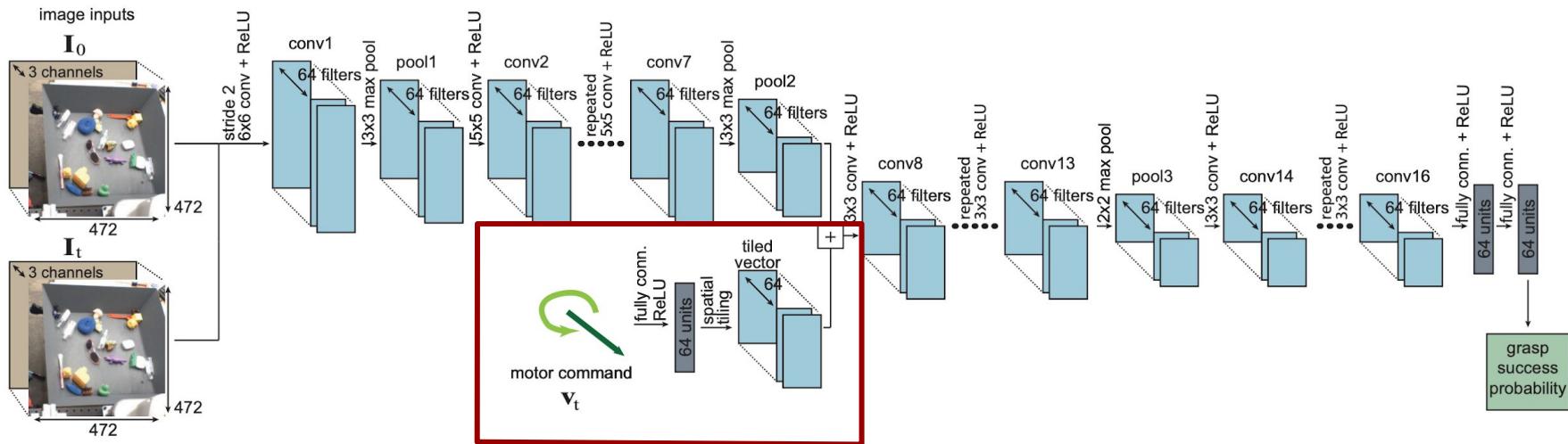
Input \mathbf{l}_t and \mathbf{l}_0
5 convolutional layers with batch normalization, followed by max pooling

Neural Network Architecture for $g(\mathbf{l}_t, \mathbf{v}_t)$



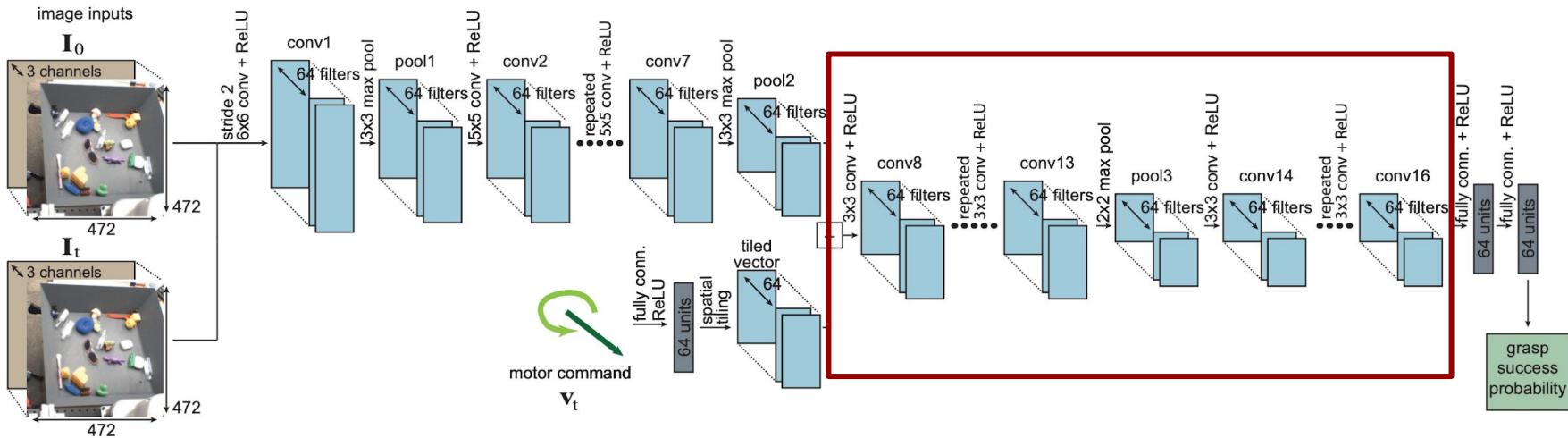
The motor command $\mathbf{v}_t = \mathbf{p}_T - \mathbf{p}_t$ is sent to 64 FC units
and then concatenated with visual activations

Neural Network Architecture for $g(\mathbf{l}_t, \mathbf{v}_t)$



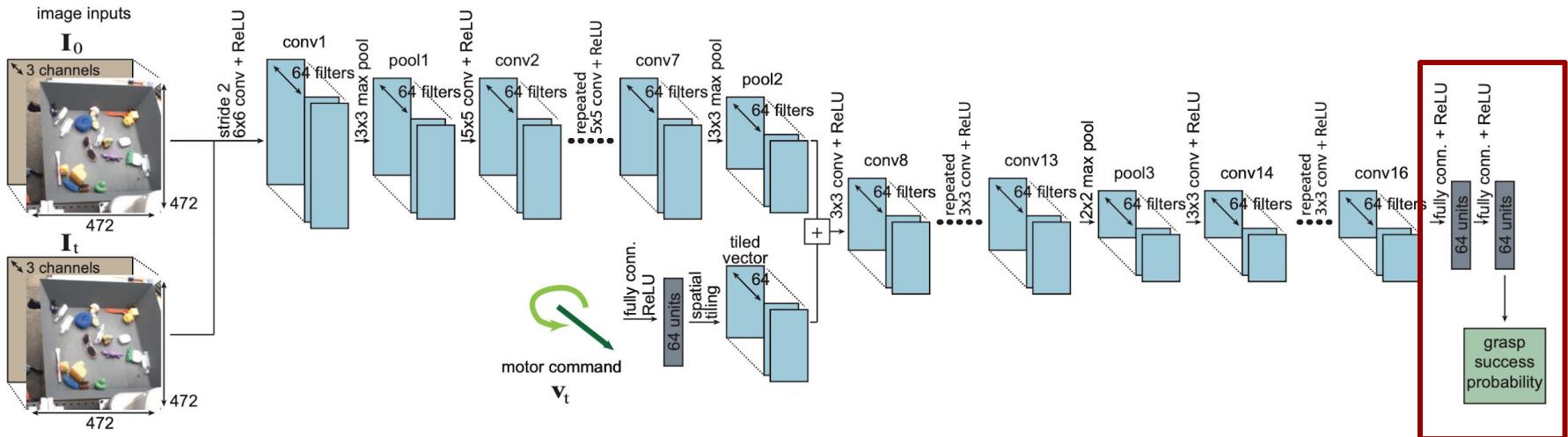
Is it just standard CNN architecture to add in the vector after the image has been processed for a few steps, or was there some underlying reason why the authors chose to do that?

Neural Network Architecture for $g(\mathbf{l}_t, \mathbf{v}_t)$



The combined activations are fed through another 9 CNN layers

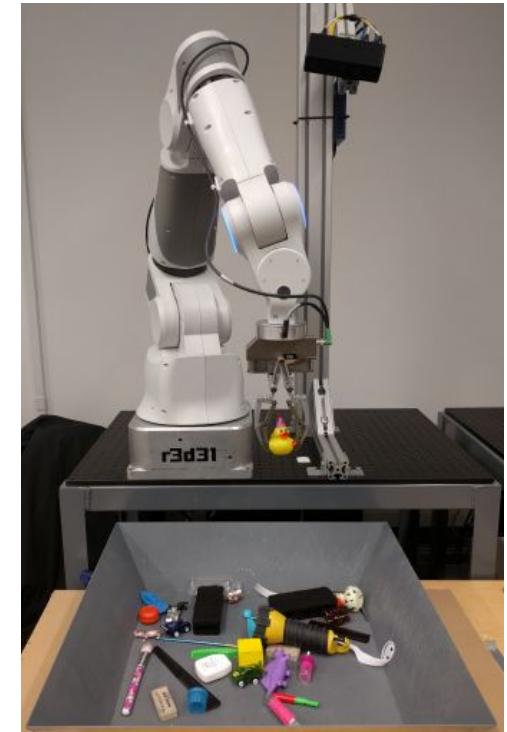
Neural Network Architecture for $g(\mathbf{l}_t, \mathbf{v}_t)$



The output features are fed into 2 fully connected layers with sigmoid activation to predict grasp success probability

Main Contributions

- Physical system: 7 DOF robot arm, 2 finger gripper, and an uncalibrated monocular camera
1. Novel CNN architecture that predicts grasping success
 2. **Continuous visual servoing function** controls the robot
 3. Large-scale data collection framework for robotic grasps

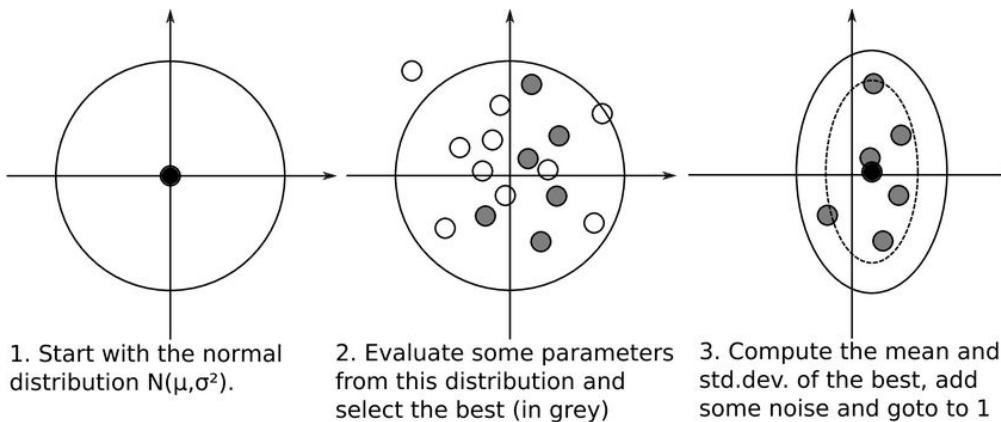


Servoing Mechanism $f(\mathbf{I}_t)$ - Algorithm

1. Given current image \mathbf{I}_t and network g .
2. Infer \mathbf{v}_t^* using g and CEM
3. Evaluate $p = g(\mathbf{I}_t, 0) / g(\mathbf{I}_t, \mathbf{v}_t^*)$
4. if $p > 0.9$ then
 - a. Output 0, close gripper.
5. else if $p \leq 0.5$, then
 - a. Modify \mathbf{v}_t^* to raise gripper height and execute \mathbf{v}_t^*
6. else
 - a. Execute \mathbf{v}_t^*
7. end if

Cross-Entropy Method for Optimization on \mathbf{v}_t

- Samples a batch of N values at each iteration
- Fits a Gaussian distribution to $M < N$ of these samples from this Gaussian
- Used $N = 64$ and $M = 6$; performed three iterations of CEM to determine the best available command \mathbf{v}_t^* to evaluate $f(\mathbf{l}_t)$, the servoing mechanism



Adding Heuristics for Raising and Lowering the Gripper

1. Given current image \mathbf{I}_t and network g .
2. Infer \mathbf{v}_t^* using g and CEM
3. Evaluate $p = g(\mathbf{I}_t, 0) / g(\mathbf{I}_t, \mathbf{v}_t^*)$
4. if $p > 0.9$ then
 - a. Output 0, close gripper.
5. else if $p \leq 0.5$, then
 - a. Modify \mathbf{v}_t^* to raise gripper height and execute \mathbf{v}_t^*
6. else
 - a. Execute \mathbf{v}_t^*
7. end if

Main Contributions

- Physical system: 7 DOF robot arm, 2 finger gripper, and an uncalibrated monocular camera
- Novel **CNN architecture** that predicts grasping success
 - Continuous visual servoing function** controls the robot
 - Large-scale data collection framework for robotic grasps
-  Parallels with RL

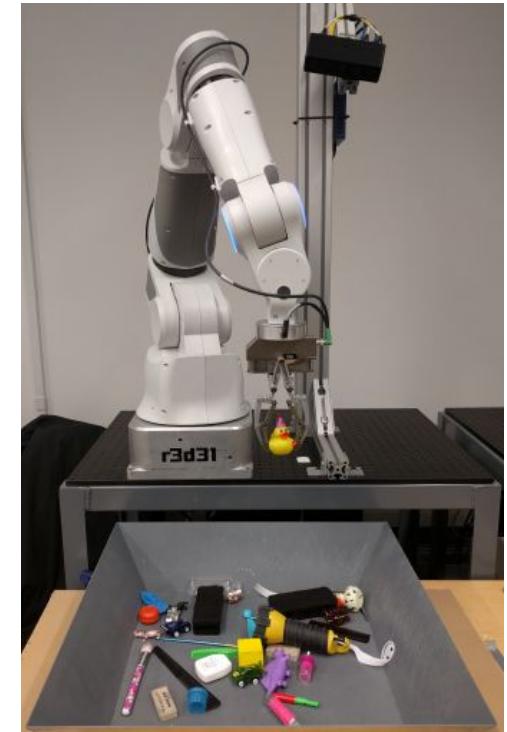
Parallels with Reinforcement Learning

The g network is similar to a Q-function in RL, but not quite the same

- If $T = 2$, then g really is a Q function (action-value function)
 - Policy is the servoing mechanism $f(\mathbf{l}_t)$
 - Reward function is 1 when the grasp succeeds and 0 otherwise
- When $T < 2$, then g is **approximately** a Q function, if the path taken from start to end position isn't important and only the final position matters
 - This approximation isn't 100% correct (can bump into other objects along the way) but it's close enough for the training to work

Main Contributions

- Physical system: 7 DOF robot arm, 2 finger gripper, and an uncalibrated monocular camera
1. Novel CNN architecture that predicts grasping success
 2. Continuous visual servoing function controls the robot
 3. Large-scale data collection framework for robotic grasps



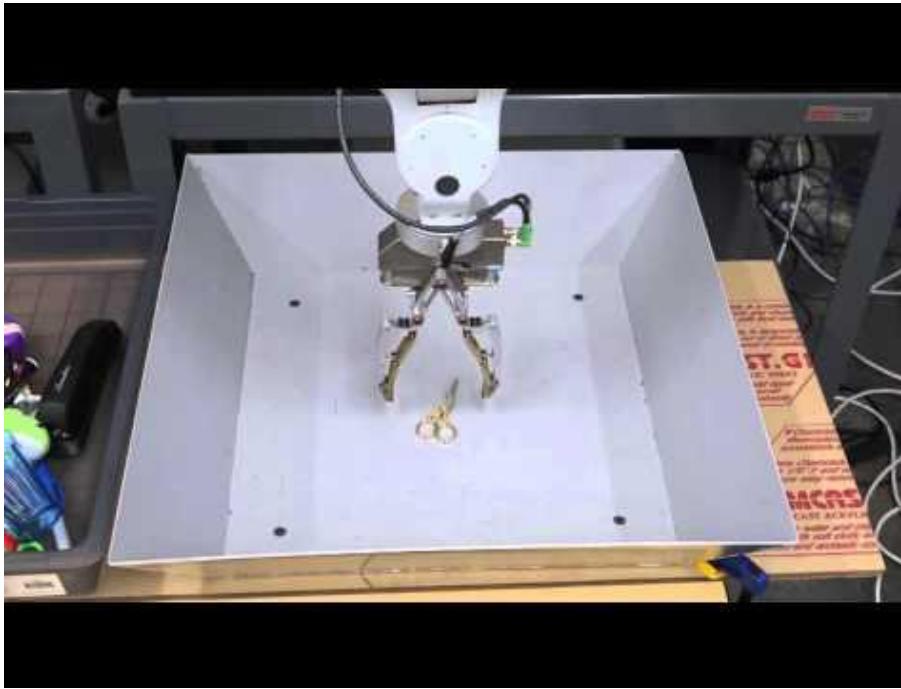
Data Collection

- All data was collected by robots in the real world. **No simulators!**
- The authors used **6-14** robot arms running in parallel over ~2 months and made a total of about **800,000 grasping attempts**
- The first **400,000 attempts** were made with **random control** and $T = 2$
 - Success rate of 10-30%
- After the first batch was in, they **trained the CNN** and used $f(\mathbf{I}_t)$ for control
- **Re-trained the network 4 times**, gradually **increasing T** from 2 to 10
- Grasp success was evaluated **without human annotation**
 - If the gripper was more than 1 cm apart after it closed, it was a success
 - Took a picture, opened the grippers, and took another picture. If the image changed, they inferred that they dropped a previously grasped object

Experimental Results

- The network was trained on a **diverse set of household and office objects**
 - These had varying size, shape, hardness, and texture
- The robot was tested on **novel objects** (unseen in training) with broadly similar attributes
- The most important test was success rate in moving objects from one bin to another (grasping without replacement)
 - The authors never attempted to train the system to grasp any specific object
- The robot achieved strong performance, with **80% success** on this task
 - Baselines: **human engineered system** 65%, **open loop control** 57%, **random** 31%
- Performance increased substantially with extra training data
 - Failure rate on grasping 30/120 objects was 17.5% after full training, 25.0% after training on half the data

Video



Novelty of These Results

- These results appeared in 2016
- First time continuous visual servoing with uncalibrated monocular cameras was used successfully for grasping
- Developed a novel CNN architecture that performed well on the task of predicting grasping success from a single camera image
- Gathered and published a data set of unprecedented size: 800,000 grasping attempts on a physical robot
 - Typical real world data set sizes were e.g. 3D scans of 10,000 objects
 - Other large data sets were only developed on simulators
- The authors achieved state of the art performance on the end task of gripping a diverse range of objects in a bin

Testing Transfer Between Robots

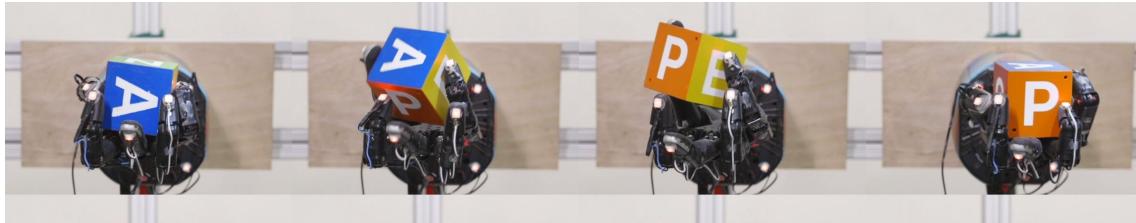
- Evaluating whether grasping data collected by one type of robot could be used to improve the grasping proficiency of a different robot
- 900,000 additional grasp attempts using a different robotic manipulator with a substantially larger variety of objects

Hardware Setup of Transfer Testing

Learning Dexterous In-Hand Manipulation

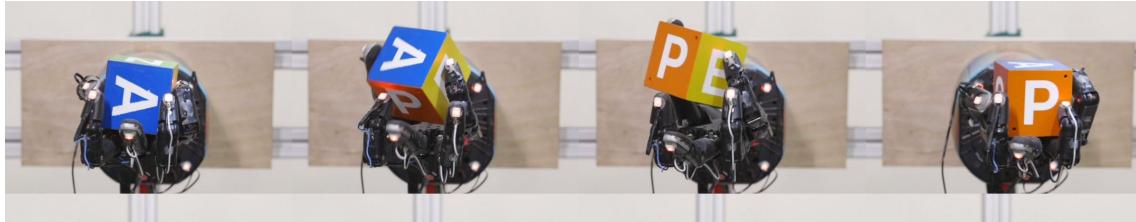
Overview

- Authors used RL to train a humanoid hand robot to manipulate a block
 - Manipulation Task: reorient a block using ground truth from a mocap system or vision only
- Training was carried out in a simulated environment
 - Fidelity of the simulation was only moderate
 - To overcome the reality gap, authors used domain randomization technique
 - Training one configuration took about 50 hours, equivalent to ~100 years experience!
- Policies learned on simulator were run on a real shadow hand robot
 - While performance in real life wasn't as strong as in the simulator, it was still state of the art



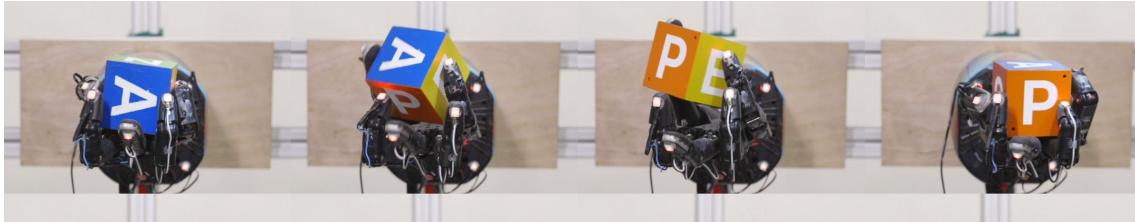
Overview

- Authors used RL to train a humanoid hand robot to manipulate a block
 - Manipulation Task: reorient a block using ground truth from a mocap system or vision only
- Training was carried out in a simulated environment
 - Fidelity of the simulation was only moderate
 - To overcome the reality gap, authors used domain randomization technique
 - Training one configuration took about 50 hours, equivalent to ~100 years experience!
- Policies learned on simulator were run on a real shadow hand robot
 - While performance in real life wasn't as strong as in the simulator, it was still state of the art



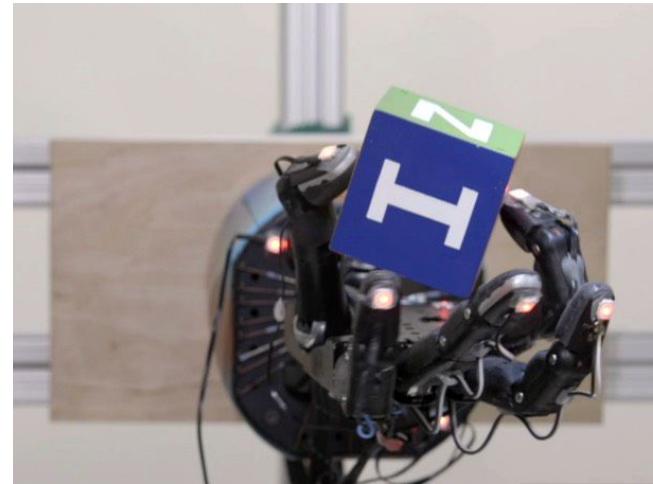
Overview

- Authors used RL to train a humanoid hand robot to manipulate a block
 - Manipulation Task: reorient a block using ground truth from a mocap system or vision only
- Training was carried out in a simulated environment
 - Fidelity of the simulation was only moderate
 - To overcome the reality gap, authors used domain randomization technique
 - Training one configuration took about 50 hours, equivalent to ~100 years experience!
- Policies learned on simulator were run on a real shadow hand robot
 - While performance in real life wasn't as strong as in the simulator, it was still state of the art



Task Overview

- Task: rotate a cubic block to the desired orientation without dropping it
- State data: hand and object positions
 - “Ground Truth” from 16 PhaseSpace trackers (use LEDs on cube and fingertips)
 - Also estimated from cameras via separate vision model



Hardware Overview

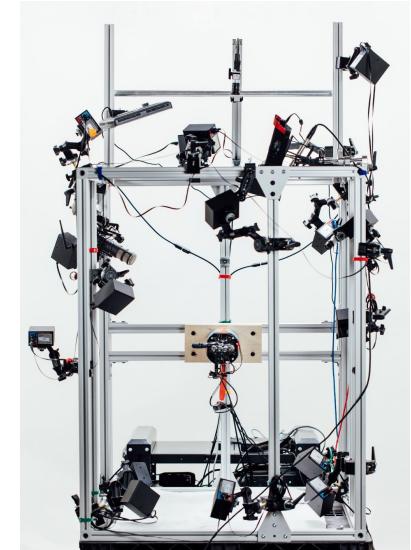
- Shadow Dexterous Hand robot
 - Most advanced humanoid hand robot on the market, released 2005
 - 24 degrees of freedom; 20 controls with tendon pairs, DC motors
 - Hand includes tactile sensors, but these were NOT used!
 - Cost is not public, discussion thread ~\$100k in 2005
- Motion capture system: 16 PhaseSpace trackers
 - LED tags attached to 5 fingertips and block corners
 - 3 cameras used for alternate vision based control
- Training hardware: insane computational resources!
 - 6144 CPU cores; est. cost ~\$720k
 - 8 NVIDIA V100 GPU; est. cost \$90k



Real World Environment: the "cage" houses 1 robot hand, 16 PhaseSpace tracking cameras, and 3 Basler RGB cameras.

Hardware Overview

- Shadow Dexterous Hand robot
 - Most advanced humanoid hand robot on the market, released 2005
 - 24 degrees of freedom; 20 controls with tendon pairs, DC motors
 - Hand includes tactile sensors, but these were NOT used!
 - Cost is not public, discussion thread ~\$100k in 2005
- Motion capture system: 16 PhaseSpace trackers
 - LED tags attached to 5 fingertips and block corners
 - 3 cameras used for alternate vision based control
- Training hardware: insane computational resources!
 - 6144 CPU cores; est. cost ~\$720k
 - 8 NVIDIA V100 GPU; est. cost \$90k



Real World Environment: the "cage" houses 1 robot hand, 16 PhaseSpace tracking cameras, and 3 Basler RGB cameras.

Hardware Overview

- Shadow Dexterous Hand robot
 - Most advanced humanoid hand robot on the market, released 2005
 - 24 degrees of freedom; 20 controls with tendon pairs, DC motors
 - Hand includes tactile sensors, but these were NOT used!
 - Cost is not public, discussion thread ~\$100k in 2005
- Motion capture system: 16 PhaseSpace trackers
 - LED tags attached to 5 fingertips and block corners
 - 3 cameras used for alternate vision based control
- Training hardware: insane computational resources!
 - 6144 CPU cores; est. cost ~\$720k
 - 8 NVIDIA V100 GPU; est. cost \$90k



Real World Environment: the "cage" houses 1 robot hand, 16 PhaseSpace tracking cameras, and 3 Basler RGB cameras.

Simulation Overview

- Simulation uses model of S.D.H. built in OpenAI Gym
- Physics engine provided by MuJoCo
 - Tendons are ignored; simplified as a single torque
 - Rigid body simulator; doesn't handle deformable or semi-rigid finger tips, which are critical for modeling contact points
- Scene rendering with Unity video game engine



Rendering of a simulated environment.

Simulation Overview

- Simulation uses model of S.D.H. built in OpenAI Gym
- Physics engine provided by MuJoCo
 - Tendons are ignored; simplified as a single torque
 - Rigid body simulator; doesn't handle deformable or semi-rigid finger tips, which are critical for modeling contact points
- Scene rendering with Unity video game engine



Rendering of a simulated environment.

Simulation Overview

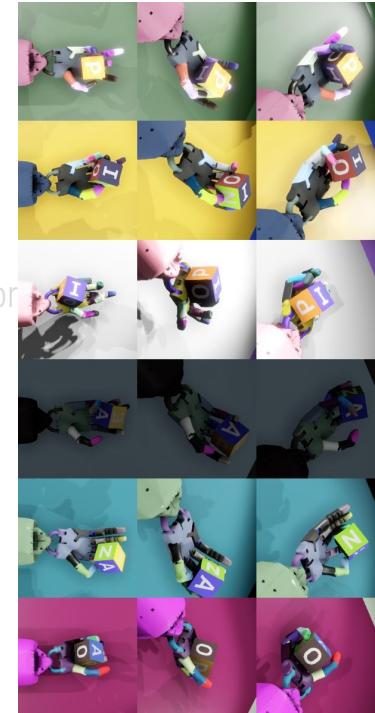
- Simulation uses model of S.D.H. built in OpenAI Gym
- Physics engine provided by MuJoCo
 - Tendons are ignored; simplified as a single torque
 - Rigid body simulator; doesn't handle deformable or semi-rigid finger tips, which are critical for modeling contact points
- Scene rendering with Unity video game engine



Rendering of a simulated environment.

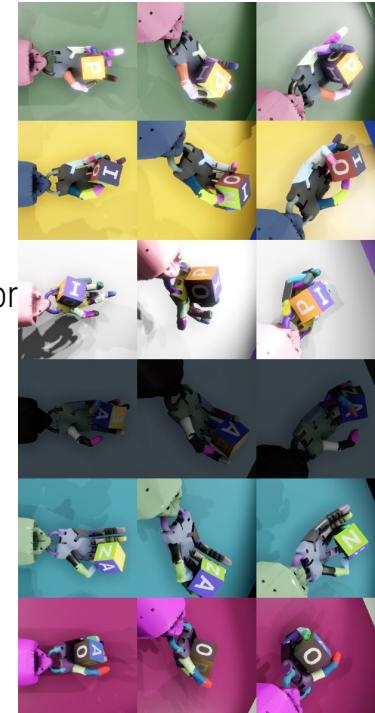
Domain Randomization and Transferable Simulations

- One of the biggest challenges is overcoming the “reality gap”
- Vary simulation attributes to learn a robust policy that generalizes well and recovers from small errors
 - Physics parameters
 - Dimensions, object / robot masses, friction coefficients, joint damping, actuator force gains, joint limits, gravity vector
 - Unmodeled effects
 - Small random perturbing forces; actuation delays; joint whiplash
 - Visual appearance
 - Color, background color, luminosity, texture are all randomized
- All of these randomizations help to **limit overfitting**



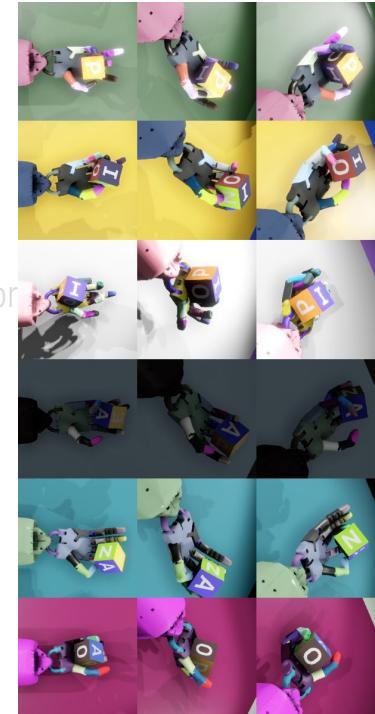
Domain Randomization and Transferable Simulations

- One of the biggest challenges is overcoming the “reality gap”
- Vary simulation attributes to learn a robust policy that generalizes well and recovers from small errors
 - Physics parameters
 - Dimensions, object / robot masses, friction coefficients, joint damping, actuator force gains, joint limits, gravity vector
 - Unmodeled effects
 - Small random perturbing forces; actuation delays; joint whiplash
 - Visual appearance
 - Color, background color, luminosity, texture are all randomized
- All of these randomizations help to **limit overfitting**



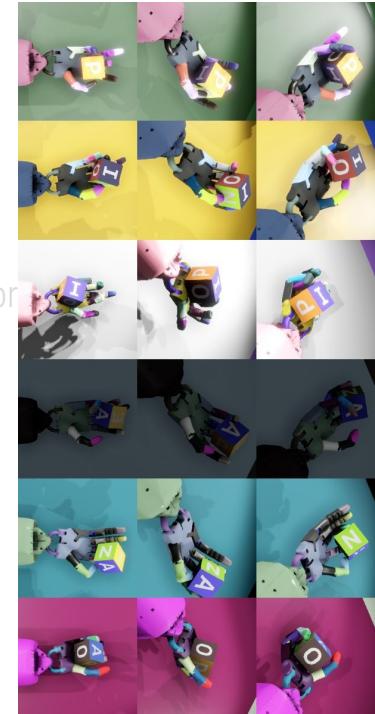
Domain Randomization and Transferable Simulations

- One of the biggest challenges is overcoming the “reality gap”
- Vary simulation attributes to learn a robust policy that generalizes well and recovers from small errors
 - Physics parameters
 - Dimensions, object / robot masses, friction coefficients, joint damping, actuator force gains, joint limits, gravity vector
 - Unmodeled effects
 - Small random perturbing forces; actuation delays; joint whiplash
 - Visual appearance
 - Color, background color, luminosity, texture are all randomized
- All of these randomizations help to **limit overfitting**



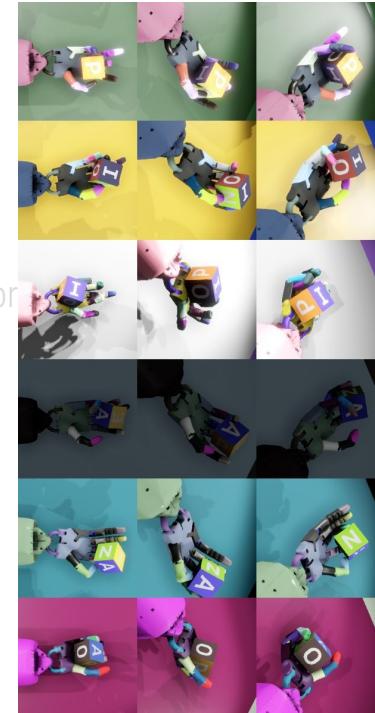
Domain Randomization and Transferable Simulations

- One of the biggest challenges is overcoming the “reality gap”
- Vary simulation attributes to learn a robust policy that generalizes well and recovers from small errors
 - Physics parameters
 - Dimensions, object / robot masses, friction coefficients, joint damping, actuator force gains, joint limits, gravity vector
 - Unmodeled effects
 - Small random perturbing forces; actuation delays; joint whiplash
 - Visual appearance
 - Color, background color, luminosity, texture are all randomized
- All of these randomizations help to **limit overfitting**



Domain Randomization and Transferable Simulations

- One of the biggest challenges is overcoming the “reality gap”
- Vary simulation attributes to learn a robust policy that generalizes well and recovers from small errors
 - Physics parameters
 - Dimensions, object / robot masses, friction coefficients, joint damping, actuator force gains, joint limits, gravity vector
 - Unmodeled effects
 - Small random perturbing forces; actuation delays; joint whiplash
 - Visual appearance
 - Color, background color, luminosity, texture are all randomized
- All of these randomizations help to **limit overfitting**



Reinforcement Learning 101

- Mapping key concepts back to RL 101
 - Environment: hand and block as simulated by MuJoCo
 - State: 85 DOF with hand position, block position and orientation; and velocities
 - Action: Torque applied to each joint
 - Reward: Proximity to goal orientation; reaching goal; not dropping block
- Policy π , Value function V are both neural networks (Deep RL)
- Actor Critic: simultaneously train π and V
 - Alternative to older value iteration and policy iteration approaches
- PPO: Proximal Policy Optimization - popular on-policy method using deep nets
- Why not DQN or DDPG?
 - DQN only learns the policy; less powerful here because it helps to also learn state function
 - DDPG is off-policy; this problem naturally lends itself to on-policy (learn by trying task in simulator)

Reinforcement Learning 101

- Mapping key concepts back to RL 101
 - Environment: hand and block as simulated by MuJoCo
 - State: 85 DOF with hand position, block position and orientation; and velocities
 - Action: Torque applied to each joint
 - Reward: Proximity to goal orientation; reaching goal; not dropping block
- Policy π , Value function V are both neural networks (Deep RL)
- Actor Critic: simultaneously train π and V
 - Alternative to older value iteration and policy iteration approaches
- PPO: Proximal Policy Optimization - popular on-policy method using deep nets
- Why not DQN or DDPG?
 - DQN only learns the policy; less powerful here because it helps to also learn state function
 - DDPG is off-policy; this problem naturally lends itself to on-policy (learn by trying task in simulator)

Reinforcement Learning 101

- Mapping key concepts back to RL 101
 - Environment: hand and block as simulated by MuJoCo
 - State: 85 DOF with hand position, block position and orientation; and velocities
 - Action: Torque applied to each joint
 - Reward: Proximity to goal orientation; reaching goal; not dropping block
- Policy π , Value function V are both neural networks (Deep RL)
- Actor Critic: simultaneously train π and V
 - Alternative to older value iteration and policy iteration approaches
- PPO: Proximal Policy Optimization - popular on-policy method using deep nets
- Why not DQN or DDPG?
 - DQN only learns the policy; less powerful here because it helps to also learn state function
 - DDPG is off-policy; this problem naturally lends itself to on-policy (learn by trying task in simulator)

Reinforcement Learning 101

- Mapping key concepts back to RL 101
 - Environment: hand and block as simulated by MuJoCo
 - State: 85 DOF with hand position, block position and orientation; and velocities
 - Action: Torque applied to each joint
 - Reward: Proximity to goal orientation; reaching goal; not dropping block
- Policy π , Value function V are both neural networks (Deep RL)
- Actor Critic: simultaneously train π and V
 - Alternative to older value iteration and policy iteration approaches
- PPO: Proximal Policy Optimization - popular on-policy method using deep nets
- Why not DQN or DDPG?
 - DQN only learns the policy; less powerful here because it helps to also learn state function
 - DDPG is off-policy; this problem naturally lends itself to on-policy (learn by trying task in simulator)

Reinforcement Learning 101

- Mapping key concepts back to RL 101
 - Environment: hand and block as simulated by MuJoCo
 - State: 85 DOF with hand position, block position and orientation; and velocities
 - Action: Torque applied to each joint
 - Reward: Proximity to goal orientation; reaching goal; not dropping block
- Policy π , Value function V are both neural networks (Deep RL)
- Actor Critic: simultaneously train π and V
 - Alternative to older value iteration and policy iteration approaches
- PPO: Proximal Policy Optimization - popular on-policy method using deep nets
- Why not DQN or DDPG?
 - DQN only learns the policy; less powerful here because it helps to also learn state function
 - DDPG is off-policy; this problem naturally lends itself to on-policy (learn by trying task in simulator)

RL Techniques Used: GAE and PPO

Generalized Advantage Estimator (GAE)

$$\hat{V}_t^{(k)} = \sum_{i=t}^{t+k-1} \gamma^{i-t} r_i + \gamma^k V(s_{t+k}) \approx V^\pi(s_t, a_t)$$

$$\hat{V}_t^{\text{GAE}} = (1 - \lambda) \sum_{k>0} \lambda^{k-1} \hat{V}_t^{(k)} \approx V^\pi(s_t, a_t)$$

$$\hat{A}_t^{\text{GAE}} = \hat{V}_t^{\text{GAE}} - V(s_t) \approx A^\pi(s_t, a_t)$$

Training is more stable for A^{GAE} than for the value function V .

Proximal Policy Optimization (PPO)

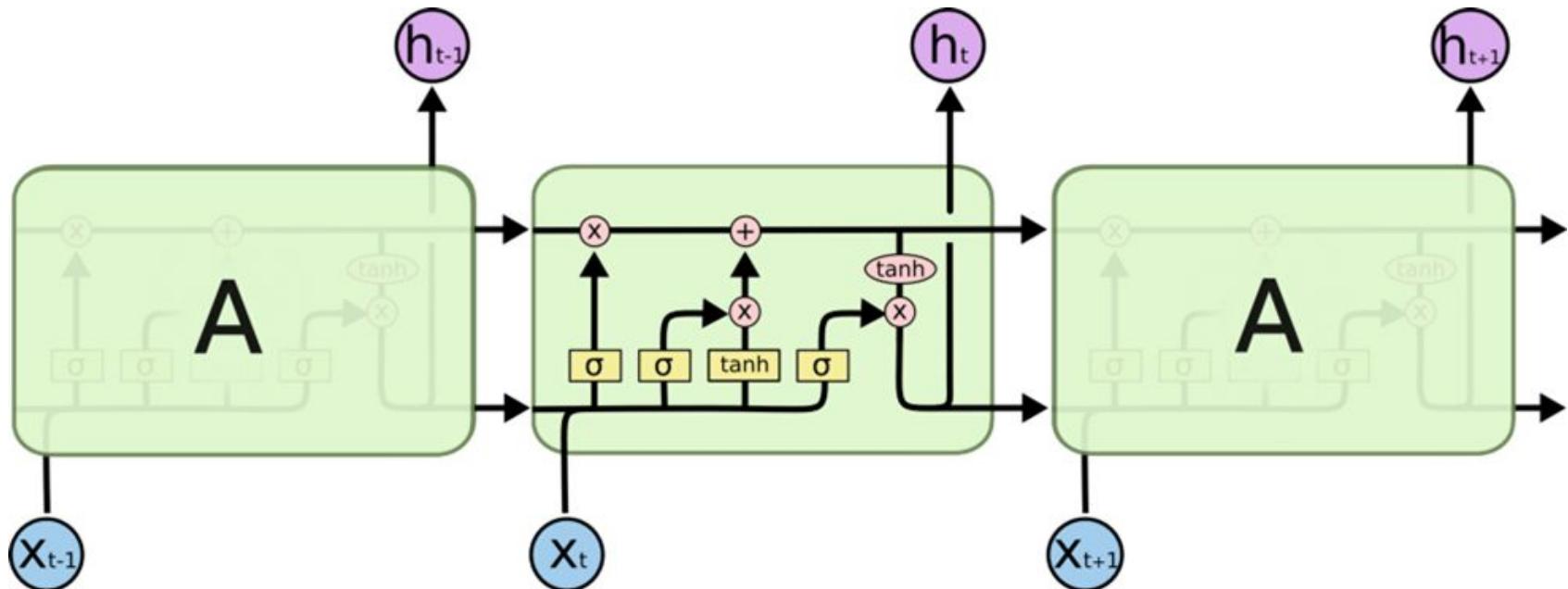
$$L_{\text{PPO}} = \mathbb{E} \min \left(\frac{\pi(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)} \hat{A}_t^{\text{GAE}}, \text{clip} \left(\frac{\pi(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t^{\text{GAE}} \right)$$

$$\frac{\pi(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}$$

Is the ratio of taking the action between the old and new policies; ϵ is a hyperparameter, often 0.2

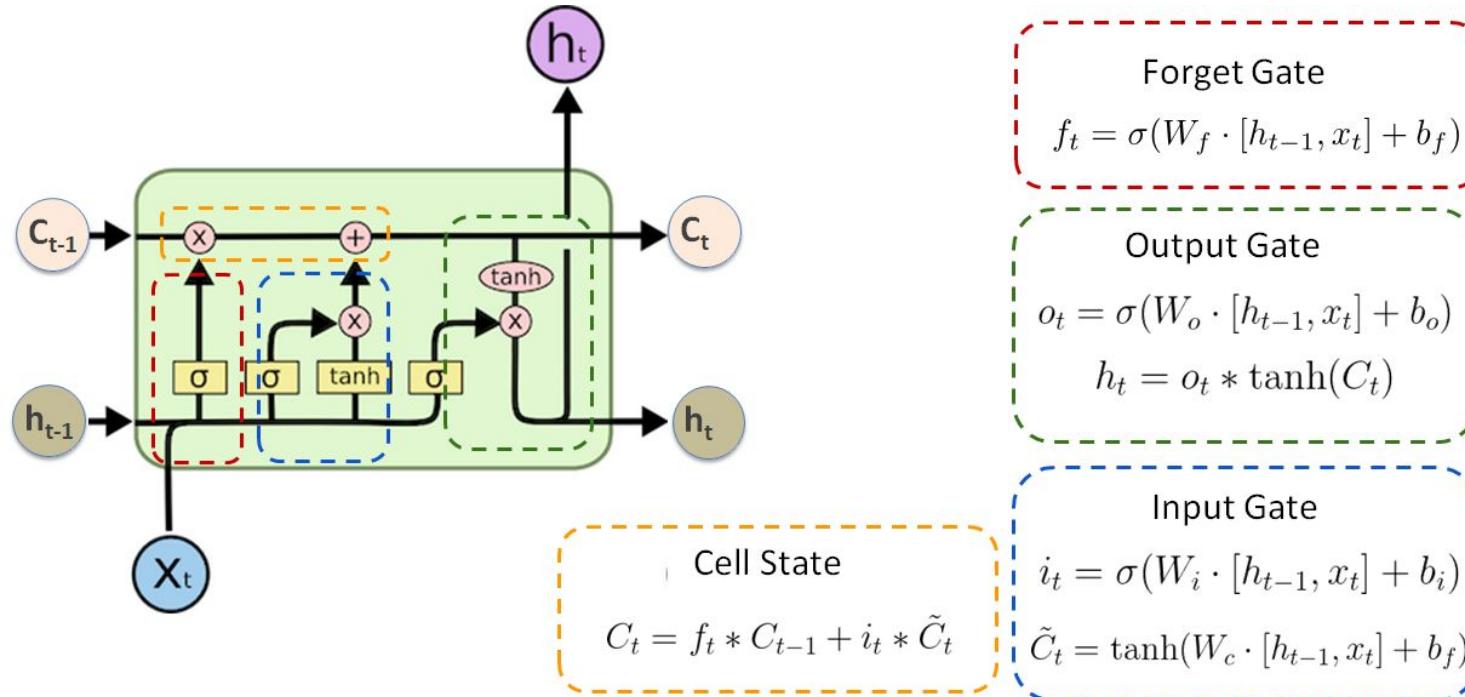
PPO encourages the policy to shift actions with better than average payoffs, but the clipping discourages changes that are too large ("proximal")

LSTM Overview



Images Credit: CS 109b Lecture Notes, Spring 2019

LSTM: Detail of One Cell



Images Credit: CS 109b Lecture Notes, Spring 2019

Learning Control Policies from State

- Policy is a recurrent neural network (RNN) with memory (LSTM)
 - As this network process inputs, it updates its internal state (memory)
 - Trained using Proximal Policy Optimization (PPO) with policy and value networks
 - Asymmetric actor-critic method; critic has more inputs than actor
- State Variables for Policy & Value Networks
 - Position of 5 fingertips (15 dim); position of object (3D); relative orientation to goal (4D quaternion); total 22 dimensions
 - Value net also has orientations for object and target; joint angles; and velocities; total 85 dim
- Actions: Discretized Joint Angles (11 Bins)
- Rewards: Angular Distance, Bonus for Success, Penalty for Drop

Learning Control Policies from State

- Policy is a recurrent neural network (RNN) with memory (LSTM)
 - As this network processes inputs, it updates its internal state (memory)
 - Trained using Proximal Policy Optimization (PPO) with policy and value networks
 - Asymmetric actor-critic method; critic has more inputs than actor
- State Variables for Policy & Value Networks
 - Position of 5 fingertips (15 dim); position of object (3D); relative orientation to goal (4D quaternion); total 22 dimensions
 - Value net also has orientations for object and target; joint angles; and velocities; total 85 dim
- Actions: Discretized Joint Angles (11 Bins)
- Rewards: Angular Distance, Bonus for Success, Penalty for Drop

Learning Control Policies from State

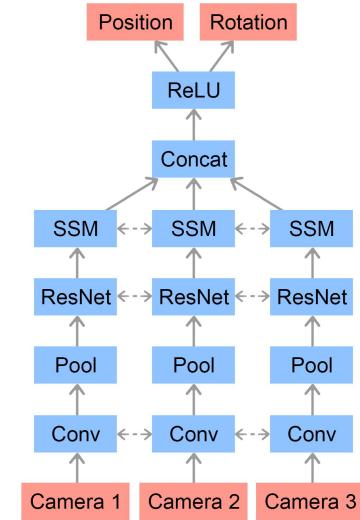
- Policy is a recurrent neural network (RNN) with memory (LSTM)
 - As this network process inputs, it updates its internal state (memory)
 - Trained using Proximal Policy Optimization (PPO) with policy and value networks
 - Asymmetric actor-critic method; critic has more inputs than actor
- State Variables for Policy & Value Networks
 - Position of 5 fingertips (15 dim); position of object (3D); relative orientation to goal (4D quaternion); total 22 dimensions
 - Value net also has orientations for object and target; joint angles; and velocities; total 85 dim
- Actions: Discretized Joint Angles (11 Bins)
- Rewards: Angular Distance, Bonus for Success, Penalty for Drop

Learning Control Policies from State

- Policy is a recurrent neural network (RNN) with memory (LSTM)
 - As this network processes inputs, it updates its internal state (memory)
 - Trained using Proximal Policy Optimization (PPO) with policy and value networks
 - Asymmetric actor-critic method; critic has more inputs than actor
- State Variables for Policy & Value Networks
 - Position of 5 fingertips (15 dim); position of object (3D); relative orientation to goal (4D quaternion); total 22 dimensions
 - Value net also has orientations for object and target; joint angles; and velocities; total 85 dim
- Actions: Discretized Joint Angles (11 Bins)
- Rewards: Angular Distance, Bonus for Success, Penalty for Drop

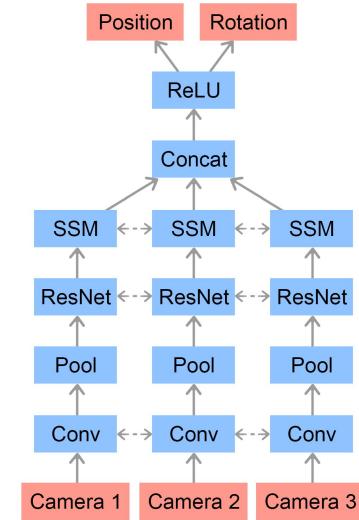
Estimating State with Vision

- The simulator is trained based on position data
- The robot can use either “ground truth” position data from the mocap system, or from RGB camera data
- They trained an auxiliary network to learn position from cameras
- Results weren’t as good as with the mocap, but still respectable



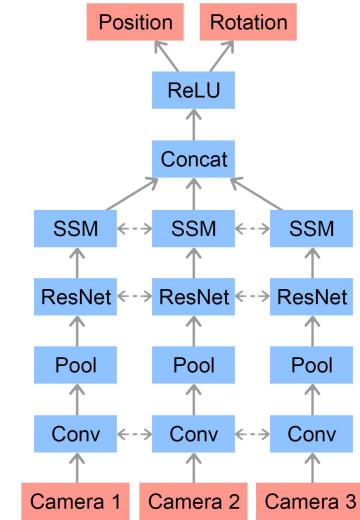
Estimating State with Vision

- The simulator is trained based on position data
- The robot can use either “ground truth” position data from the mocap system, or from RGB camera data
- They trained an auxiliary network to learn position from cameras
- Results weren’t as good as with the mocap, but still respectable



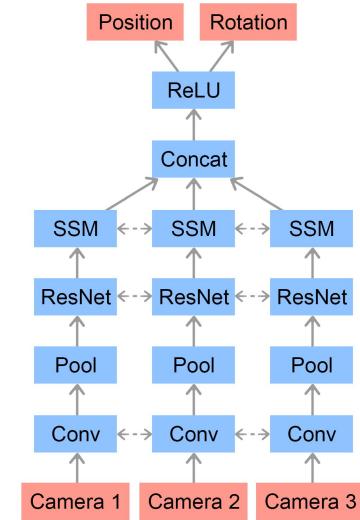
Estimating State with Vision

- The simulator is trained based on position data
- The robot can use either “ground truth” position data from the mocap system, or from RGB camera data
- They trained an auxiliary network to learn position from cameras
- Results weren’t as good as with the mocap, but still respectable



Estimating State with Vision

- The simulator is trained based on position data
- The robot can use either “ground truth” position data from the mocap system, or from RGB camera data
- They trained an auxiliary network to learn position from cameras
- Results weren’t as good as with the mocap, but still respectable



Experimental Results

- The robot learned this task with performance comparable to a person
 - It also learned grasp types similar to those used by people
- Benchmark experiment: 80 attempts, ending after 80 seconds or 1 drop

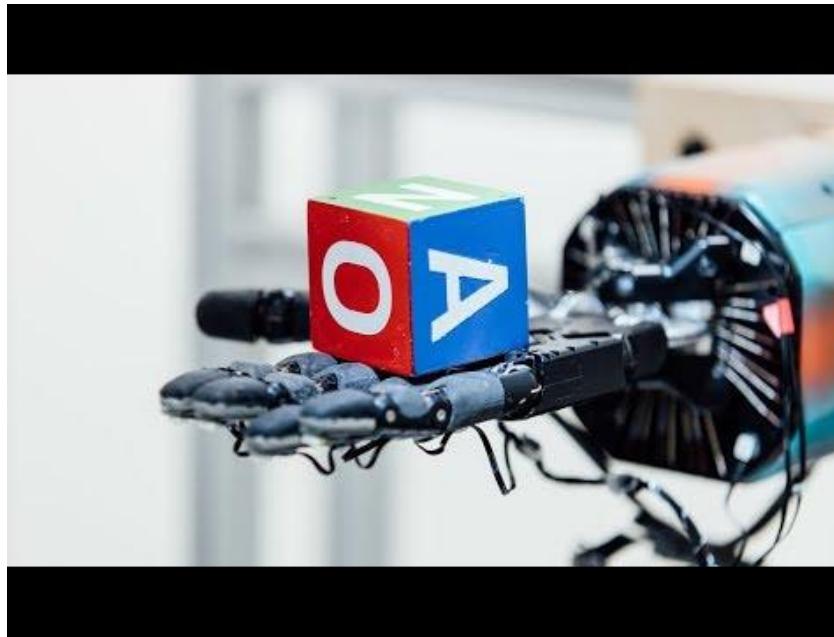
Simulated task	Mean	Median
Block (state)	43.4 ± 13.8	50
Block (state, locked wrist)	44.2 ± 13.4	50
Block (vision)	30.0 ± 10.3	33
Octagonal prism (state)	29.0 ± 19.7	30

Physical task	Mean	Median
Block (state)	18.8 ± 17.1	13
Block (state, locked wrist)	26.4 ± 13.4	28.5
Block (vision)	15.2 ± 14.3	11.5
Octagonal prism (state)	7.8 ± 7.8	5

Stylized Findings

- Robot does better in simulation
- Significant drop to real hand, but performance still respectable
- Modest further drop using camera
- Some transfer learning for octagonal prism task

Video



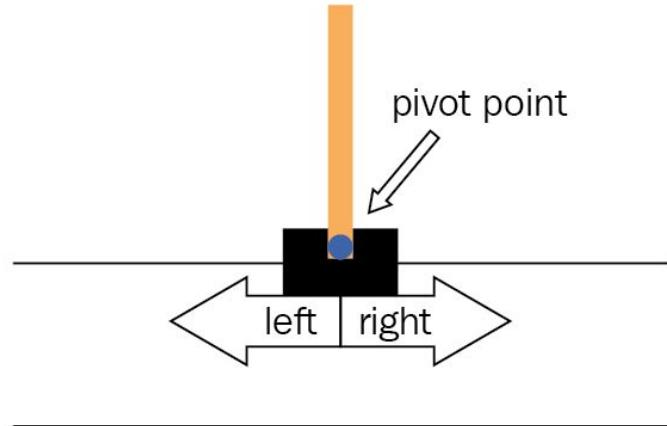
Stupid Human Tricks: Block Manipulation

- Task 1: Try to manipulate the block to different positions using one hand similar to the way the robot did
- Task 2: Try again with your eyes closed after initial glimpse
- Conclusion 1: This task isn't as trivial as picking objects, but it's not hard
- Conclusion 2: By using only vision and not touch, the robot is losing a lot of very useful sensory information. The human way of doing this is much more tactile.

OpenAI Gym: An Accessible RL Environment

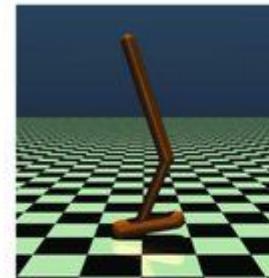
- OpenAI gym is a python environment for RL
 - <https://gym.openai.com/docs/>, <https://github.com/openai/gym>
 - Very user friendly; pip install gym and you are off to the races
- Here is a minimal program demonstrating it

```
import gym
env = gym.make('CartPole-v0')
env.reset()
for _ in range(1000):
    env.render()
    env.step(env.action_space.sample())
# take a random action
env.close()
```

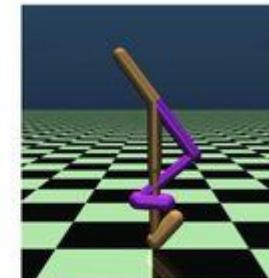


MuJoCo: A Physics Simulator for Robotics

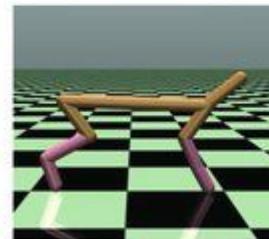
- Multi-Joint dynamics with Contact
 - Developed at University of Washington Movement Control Laboratory
 - Now developed commercially by Roboti LLC
 - <http://mujoco.org/>,
- Physics engine for R&D in robotics, biomechanics, graphics & animation



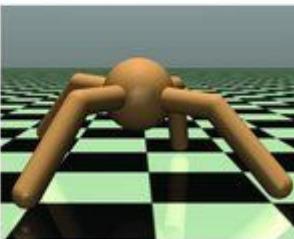
Hopper



Walker2d



Half-Cheetah



Ant

Unity Engine: A Scene Rendering Library

- Cross-platform video game and rendering engine
 - Started in 2005 for Apple Mac OS-X; now supported on > 25 platforms
- Initially for video games, now used in film, automotive, engineering
- Core capabilities: modeling, animation, rendering, simulation
- Scenes include digital “assets” that are rendered
 - These can be purchased or downloaded free, or created from scratch



Example from Unity website: rendering of a BMW.
Unity can create assets from CAD design files!

Green AI and Energy Usage in ML

- Green AI is a recent trend
 - Participants considering energy usage in model training, not just final task performance
- Some ML achievements have come at the cost of huge energy usage
 - OpenAI estimated to have spent \$30 million (!) training its agent to play DOTA II
 - It cost an estimated \$2 million / 656,000 kWh / 323 tons CO₂ to train a Transformer language model (274,000 GPU hours) using Neural Architecture Search (NAS)
- This project was relatively energy efficient
 - Model trained in 50 hours wall time; 400 GPU hours, 3 million CPU hours
 - Energy usage ~169 kWh based on 300W GPU power, 130W CPU power for 16 cores

Green AI and Energy Usage in ML

- Green AI is a recent trend
 - Participants considering energy usage in model training, not just final task performance
- Some ML achievements have come at the cost of huge energy usage
 - OpenAI estimated to have spent \$30 million (!) training its agent to play DOTA II
 - It cost an estimated \$2 million / 656,000 kWh / 323 tons CO₂ to train a Transformer language model (274,000 GPU hours) using Neural Architecture Search (NAS)
- This project was relatively energy efficient
 - Model trained in 50 hours wall time; 400 GPU hours, 3 million CPU hours
 - Energy usage ~169 kWh based on 300W GPU power, 130W CPU power for 16 cores

Green AI and Energy Usage in ML

- Green AI is a recent trend
 - Participants considering energy usage in model training, not just final task performance
- Some ML achievements have come at the cost of huge energy usage
 - OpenAI estimated to have spent \$30 million (!) training its agent to play DOTA II
 - It cost an estimated \$2 million / 656,000 kWh / 323 tons CO₂ to train a Transformer language model (274,000 GPU hours) using Neural Architecture Search (NAS)
- This project was relatively energy efficient
 - Model trained in 50 hours wall time; 400 GPU hours; 300,000 CPU hours
 - Energy usage ~2600 kWh based on 300W GPU power, 130W CPU power for 16 cores

Contributions, Context, Limitations, and Future Work

Hardware Systems

- Mechanical Hands
 - Simplistic
 - Antipodal - 2 parameters: Angle, location, then just close
 - Complicated:
 - Shadow-Hand
 - Tendon-based, very painful to re-thread
 - Similar method in Yale OpenHand
 - How to make them robust?
- Robot arm
 - Collaborative robots: Way safer and lighter
 - No longer need cages
 - IK / path planning can be ignored



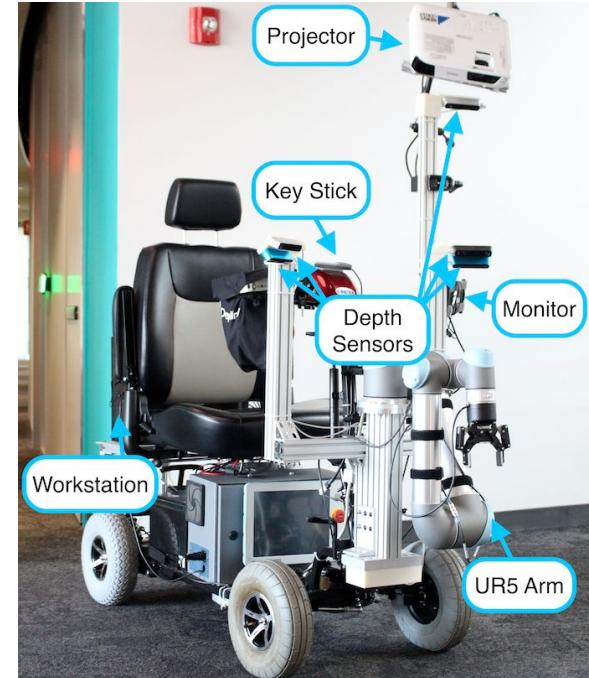
Novelty of These Results -- Limitations & Future Work

- Work Since Then (software)
 - Domain randomization
 - FAT
 - Hierarchical (e.g. dishwasher)
 - Grasping in cluttered / occluded environments
- Future work: dynamic movements?
 - <https://giant.gfycat.com/AcceptablePoorAsiaticlessfreshwatertadpoles.webm>
 - “[World's fastest workers](#)”
 - [Dart throwing](#)
- Grasping around corners / non anti-podal grasp
- Grasping around objects (e.g. shelf)



Questions they raise

- What role can manipulation play in day-to-day life?
 - Immobile 6 DoF arm
 - Extensive sensing required for ShadowHand
- How can we accelerate manipulation research?
- What previously unachievable areas of robotics can machine learning let us imagine?
 - Are the limitations just in hardware now?



Conclusion (both papers) + project ideas

- We can learn robust robotic manipulation strategies in end-to-end methods, given ‘enough data’
 - Robust even to hardware inaccuracies
 - Robotic grasping: bin pick-and-place almost a solved problem since 2016, due to ease of collecting real-life and simulation data
 - In-hand manipulation still open field
- (Project ideas TBD)

Not Stupid Human Trick: Winter Wind Etude

- As a counterpoint to the trivial and easy human manipulation tasks...
- Here is a human taking on a “manipulation task” at the apex of difficulty
- This is a video of Evgeny Kissin playing a notoriously difficult piano piece: Chopin’s Etude Opus 25, Number 11, nicknamed “Winter Wind”
- Whether or not you are a fan of classical piano, consider this for a moment merely as a motion planning problem
- The speed and precision required are astounding!



Stupid Human Tricks I: Picking and Packing

- In-Class demonstration with volunteers
- Task 1: Pick and pack objects from one bin to another with two hands
- Task 2: Use just one hand
- Task 3: Use one hand with only two fingers (thumb and index finger)
- Task 4: Use one hand, all fingers, but blindfolded
- Conclusion: Humans are *way better* at picking and packing than this robot!