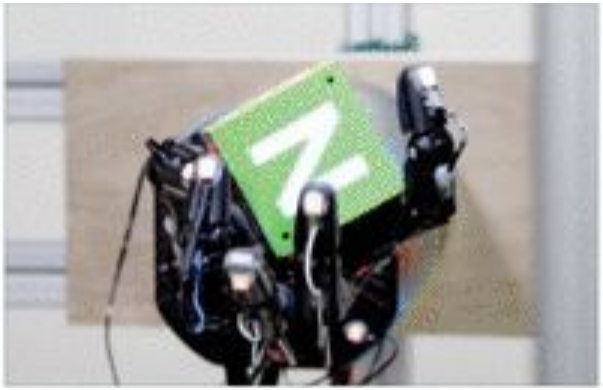
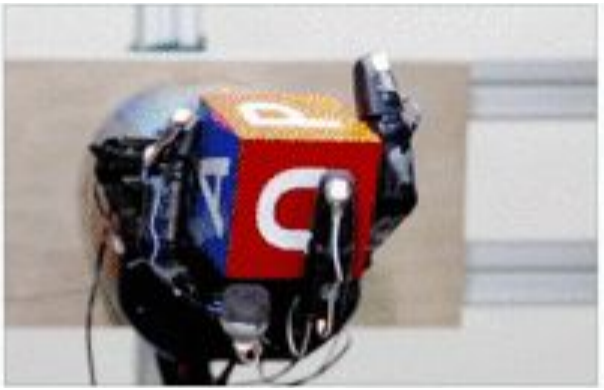
Learning for Manipulation

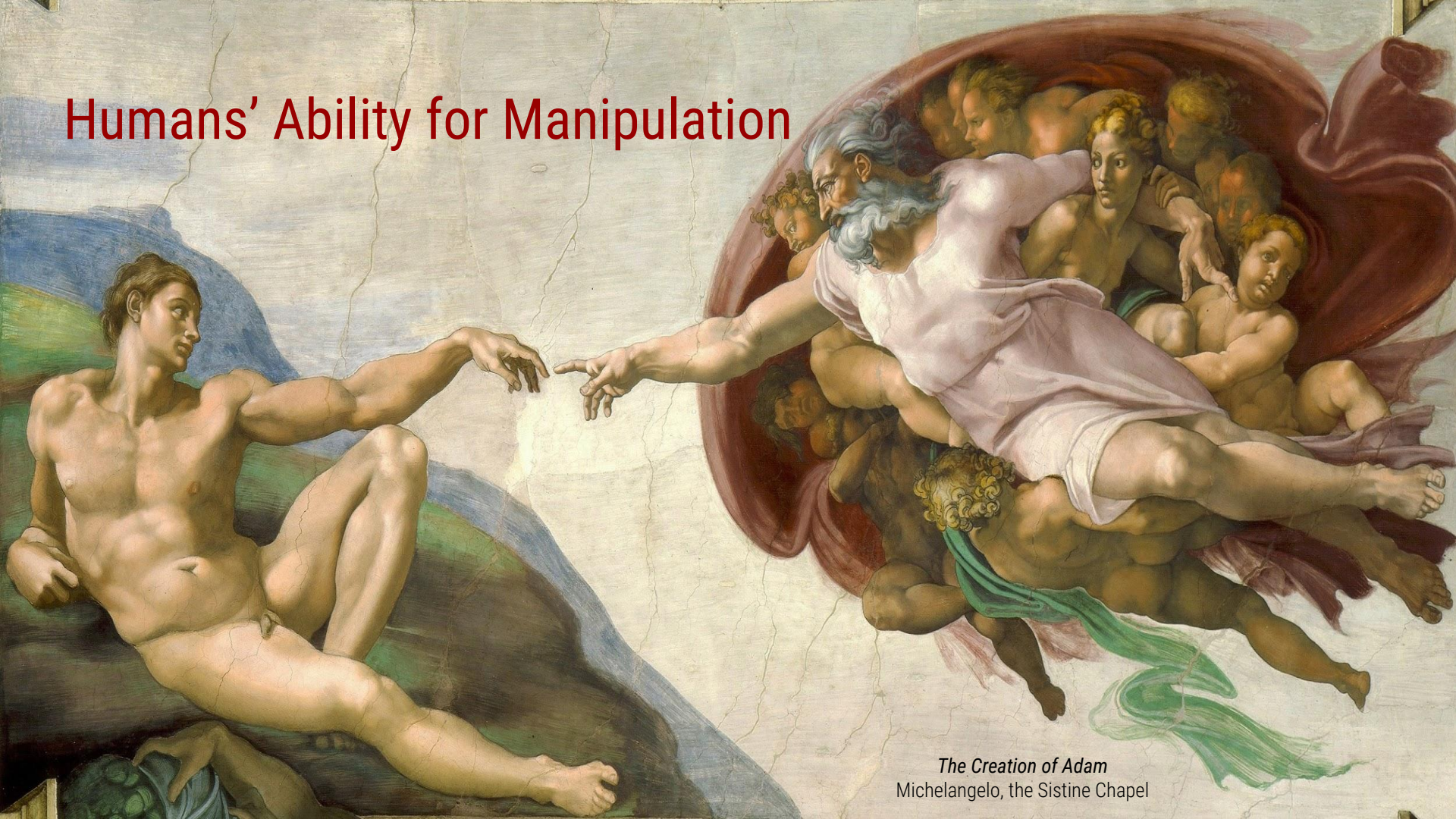
Grace Eysenbach, Nao Ouyang,
Maggie Wang, Michael Emanuel

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

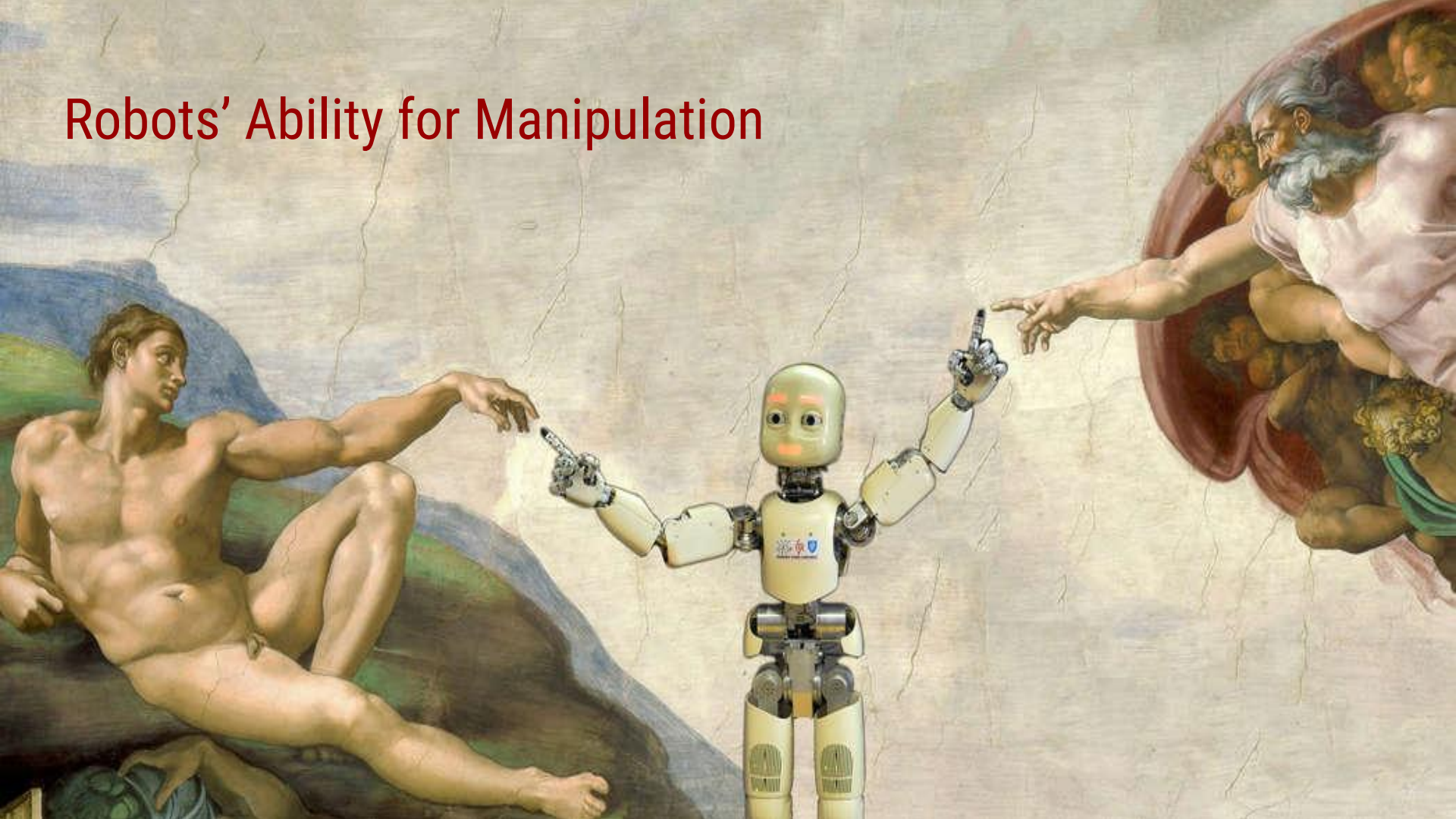
Motivation

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

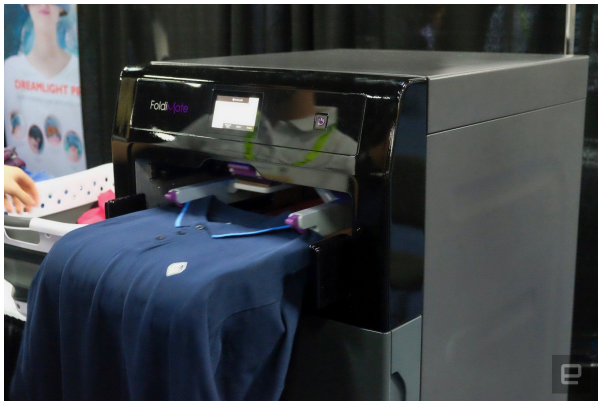
- Why is this so hard? **Huge control space!**

- 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

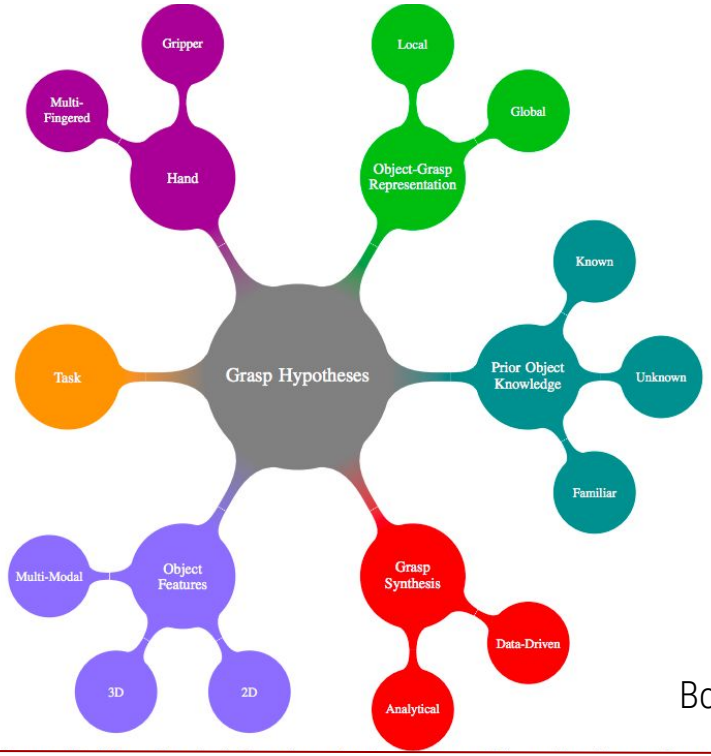
Making Robots Generalizable

- 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

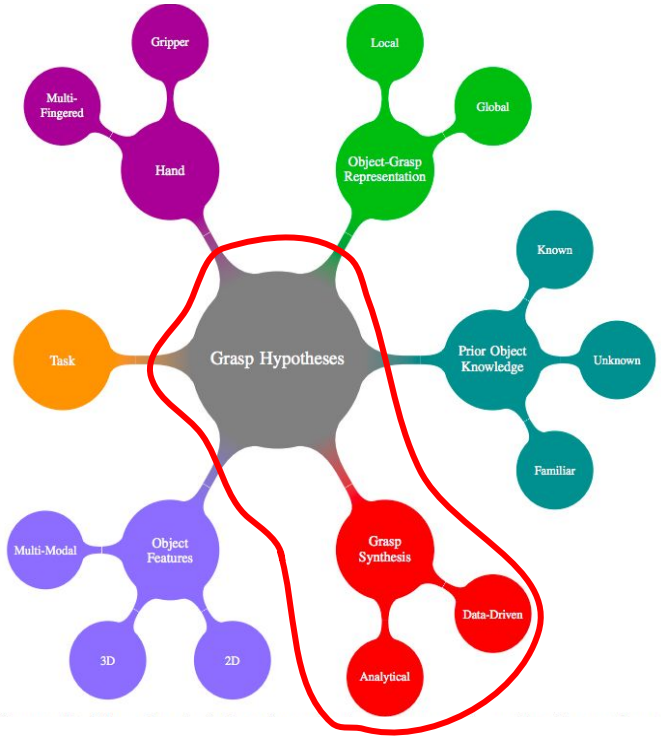
- **Analytical** → Creating kinematic and dynamic models to apply to grasping

  (+) 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



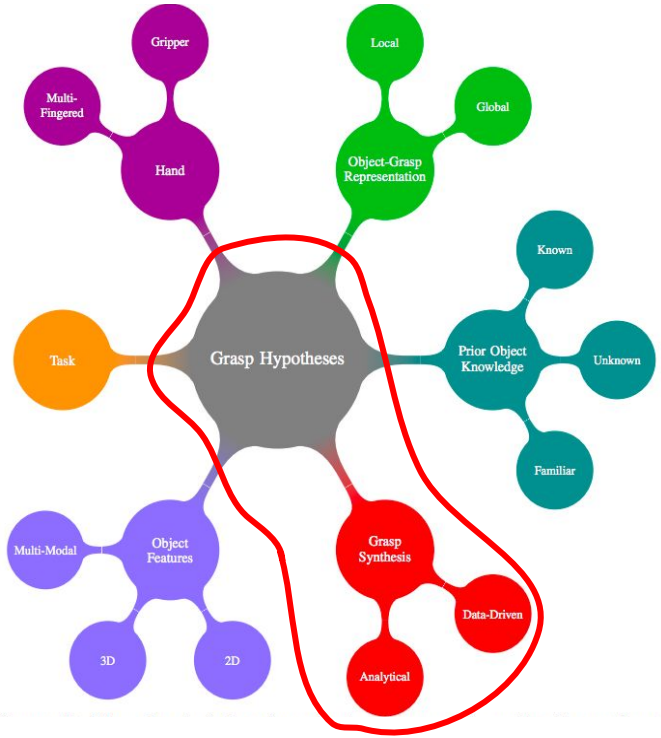
- **Analytical** → Creating kinematic and dynamic models to apply to grasping

  (+) Good for tasks 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

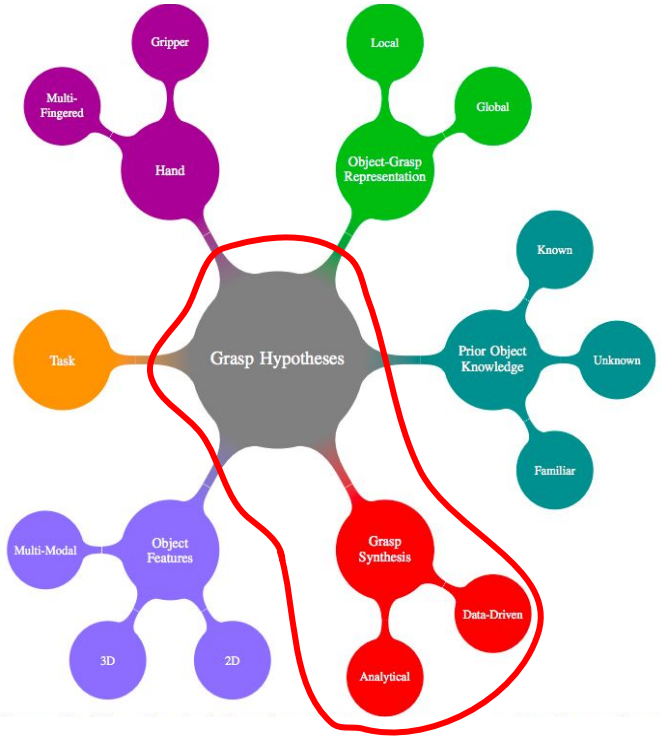
Grasp Synthesis: Analytical vs. Data-Driven

- **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

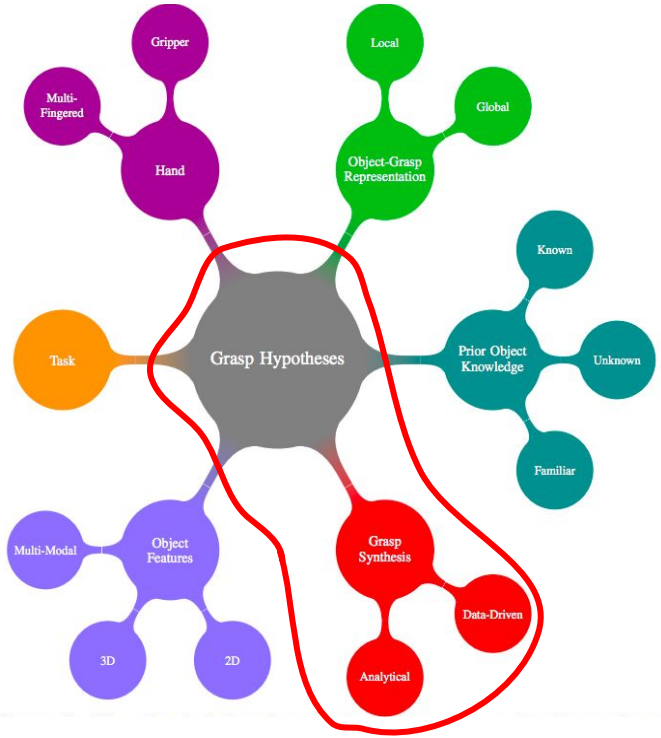


- **Data-Driven** → relies on sampling grasp candidates for an object and ranking them according to a specific metric using existing grasp experience

  (+) Much larger range of potential applications

  (-) Challenges of data collection + representation

Grasp Synthesis: Analytical vs. Data-Driven

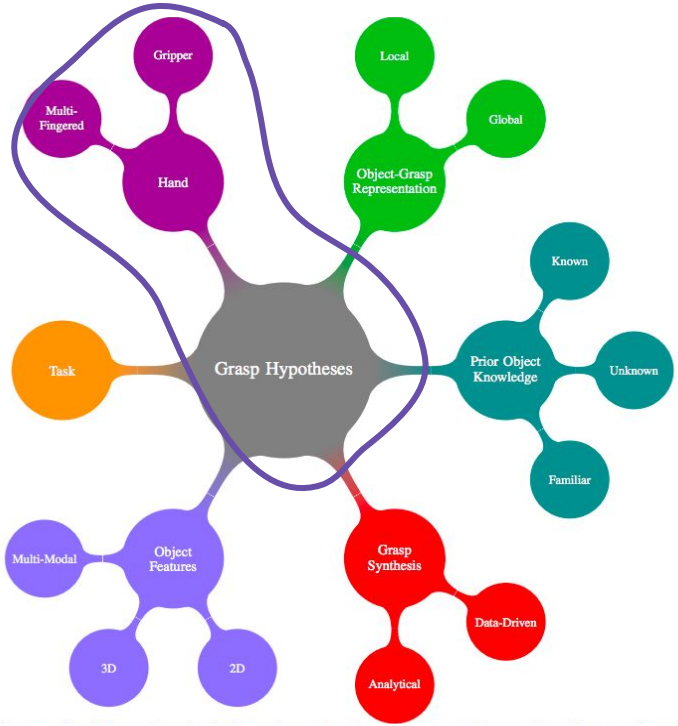


- **Data-Driven** → relies on sampling grasp candidates for an object and ranking them according to a specific metric using existing grasp experience

  (+) Much larger range of potential applications

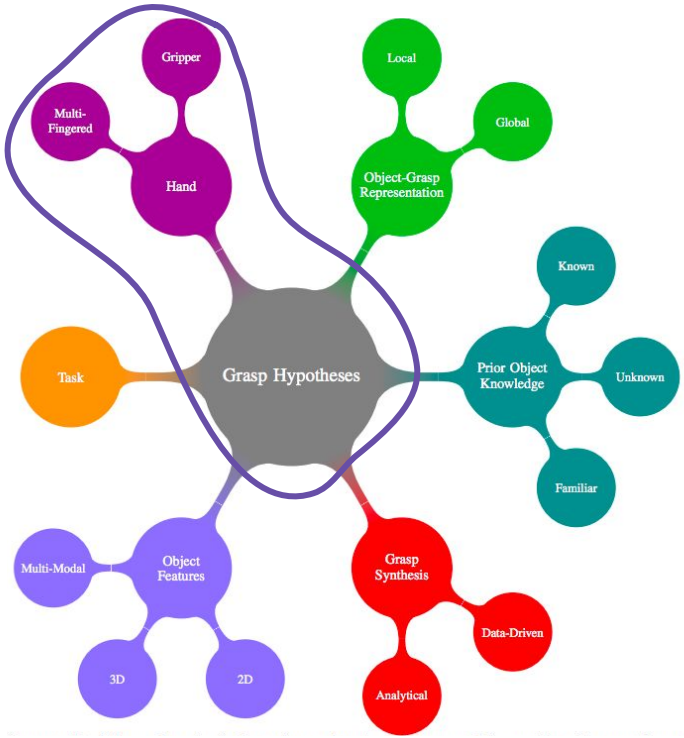
  (-) Challenges of data collection + representation

Means of Grasping: Gripper vs. Multi-Fingered



- **Gripper** → "Learning Hand-Eye Coordination for Robotic Grasping"

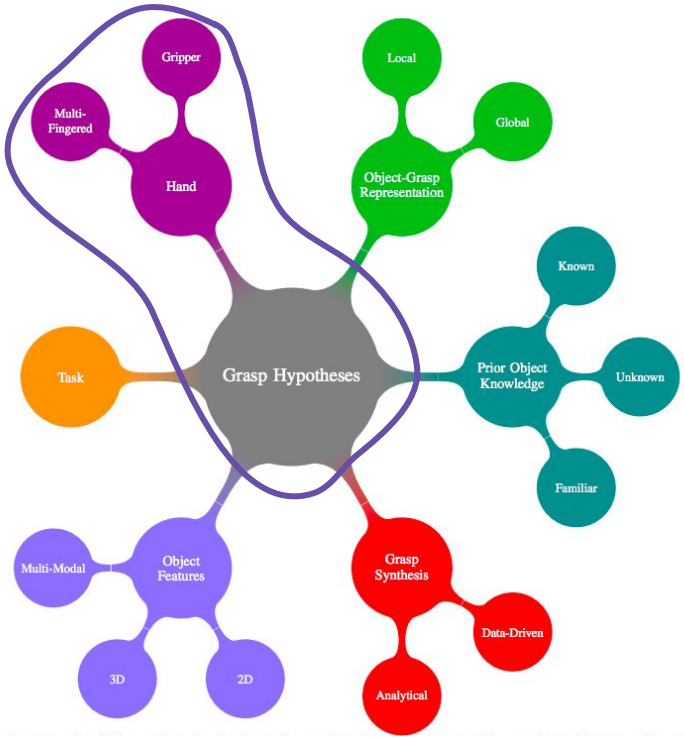
Means of Grasping: Gripper vs. Multi-Fingered



- **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)

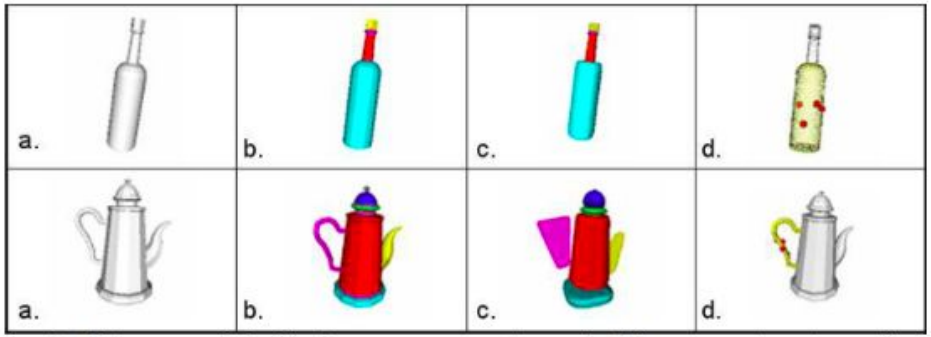


Figure 10: a) Object model. b) Part segmentation. c) SQ approximation. d) Graspable part and contact points [89].

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.

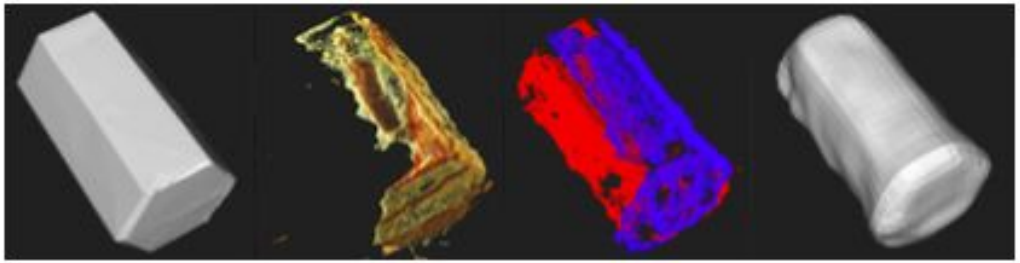
Method: Combine sensory data with heuristics to analyze candidate grasps

(+) 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)

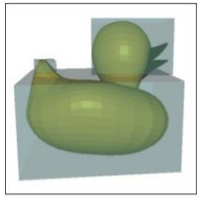


(a) (b) (c) (d)

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].

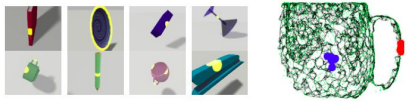
Object Features: 3D vs. 2D vs. Multi-Modal

- 3D



Box Decomposition
(Hubner et al., 2008)

- 2D

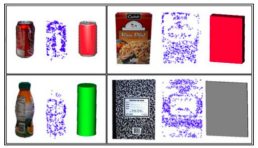


(a) (b)

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

Grasping Using Vision
(Saxena et al., 2008)

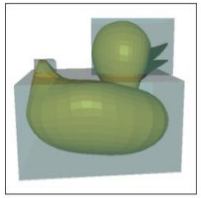
- Multi-Modal → Combine 3D and 2D



Multiview Object Representation
(Detry et al, 2009)

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

- 3D



Box Decomposition (Hubner et al., 2008)

- 2D

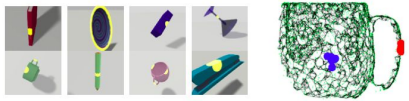
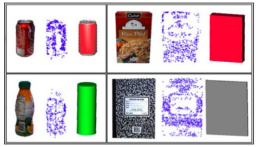


Figure 12: Labeled training data. 12a) One example for each of the eight object classes in training data in [28] along with their grasp labels (in yellow). 12b) Positive (red) and negative examples (blue) for grasping points [94].

Grasping Using Vision (Saxena et al., 2008)

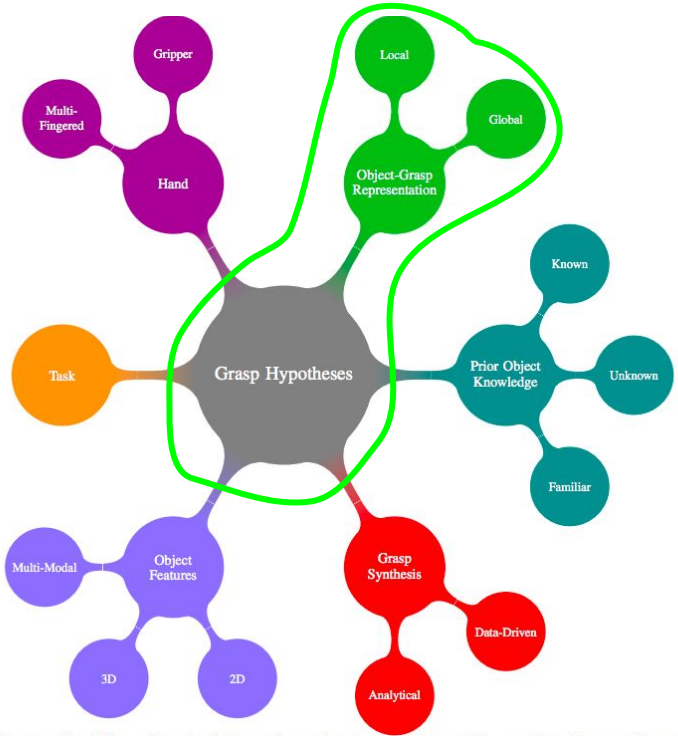
- Multi-Modal → Combine 3D and 2D



Multiview Object Representation (Detry et al, 2009)

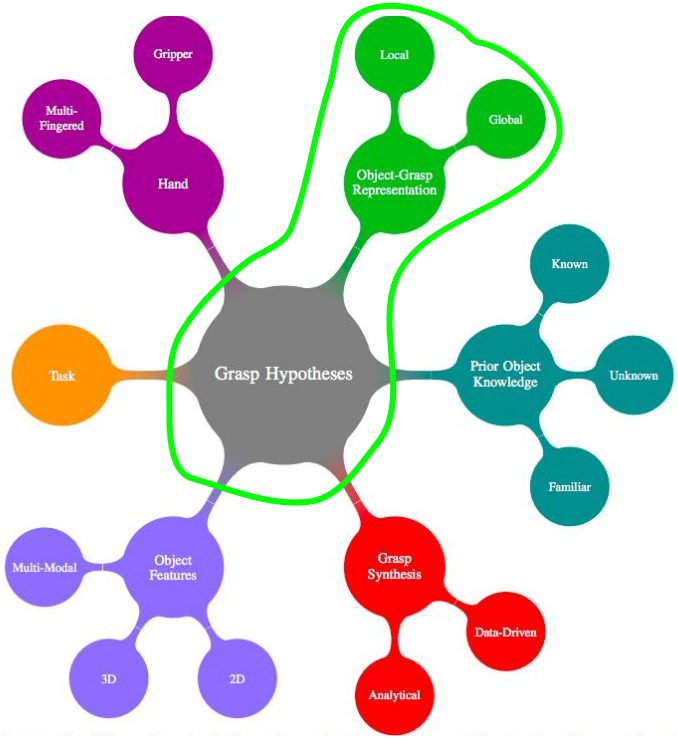
Key Concern: Costs of Complexity

Object-Grasp Representation: Local vs. Global



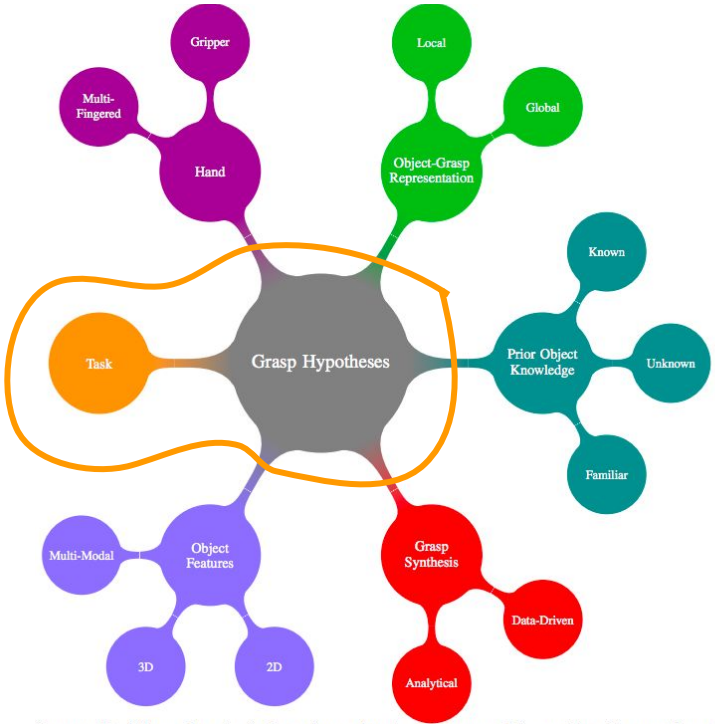
- **Local** → examining low level features (ex. Contour descriptors)
- **Global** → examining shape of object as a whole (ex. Object outline)

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)

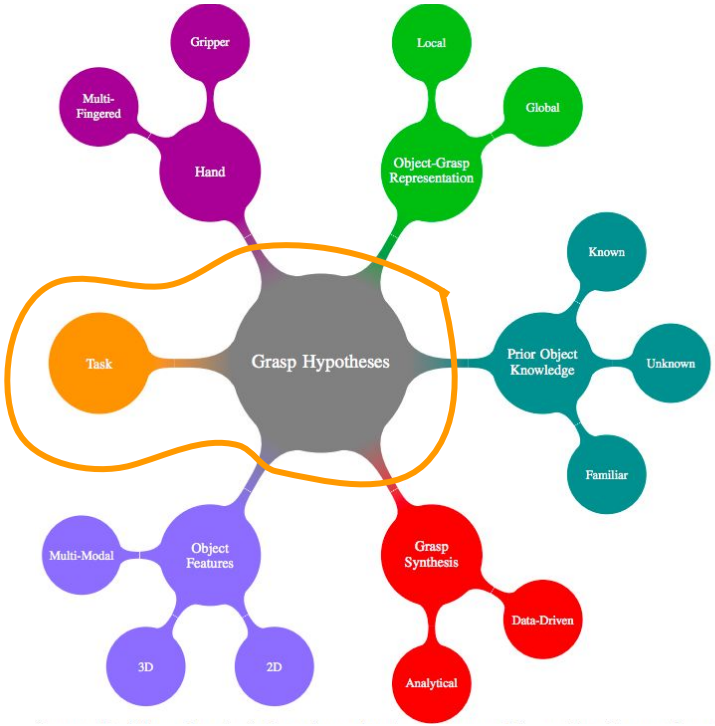
Task: *What do we want to do?*



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!

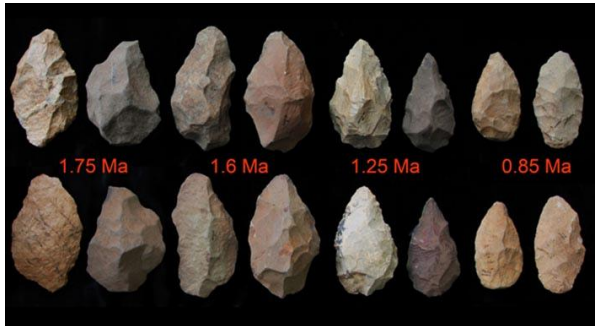
- Grasp database storage constraints?
- Objects it will encounter?
- Sensors on object?
- Etc.

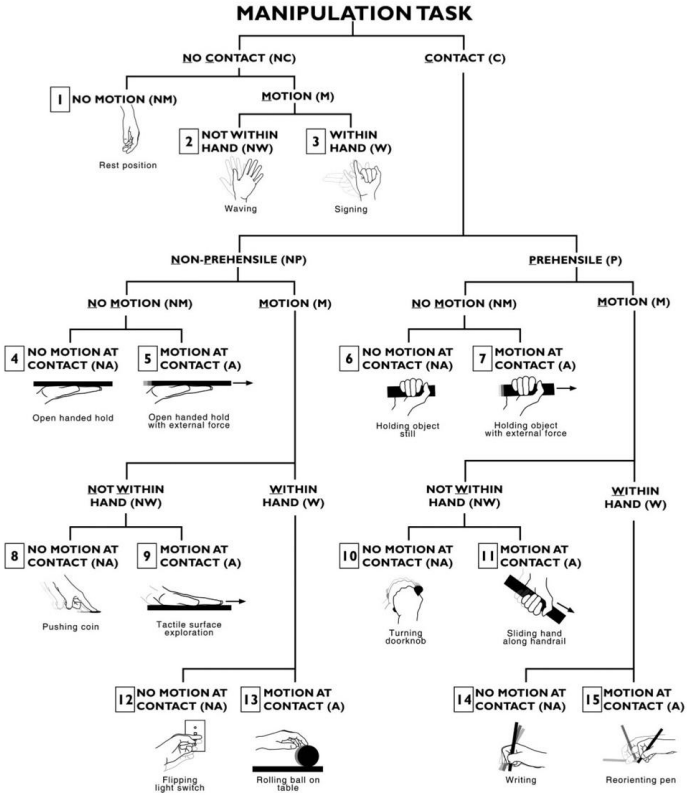
Manipulation

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

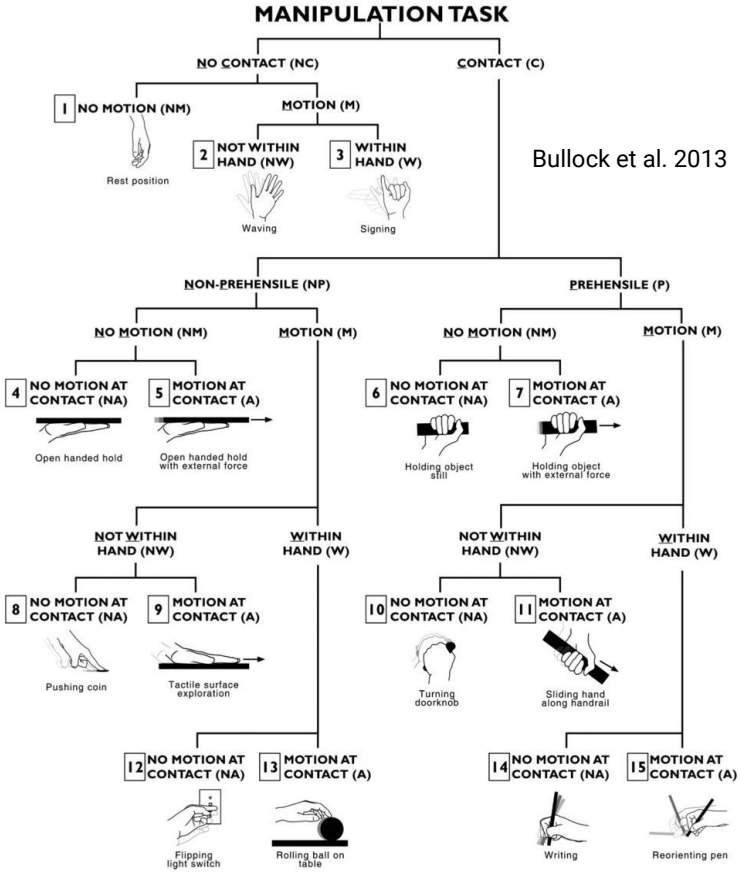
Types of Manipulation

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)

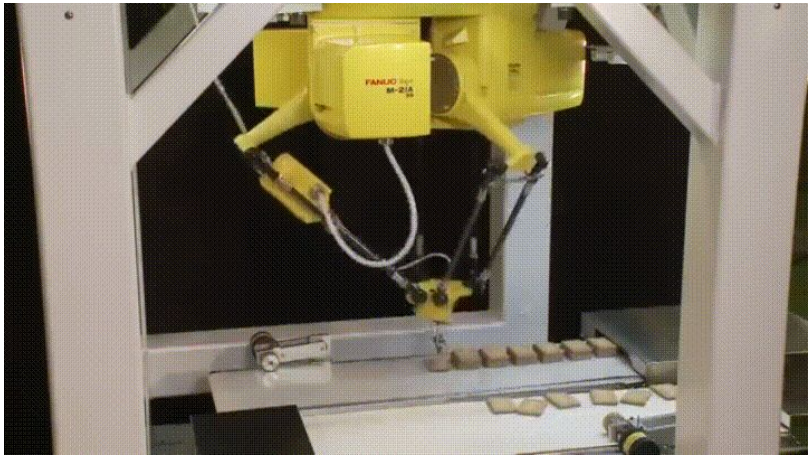


Bullock et al. 2013

Pick-and-Place Manipulation

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

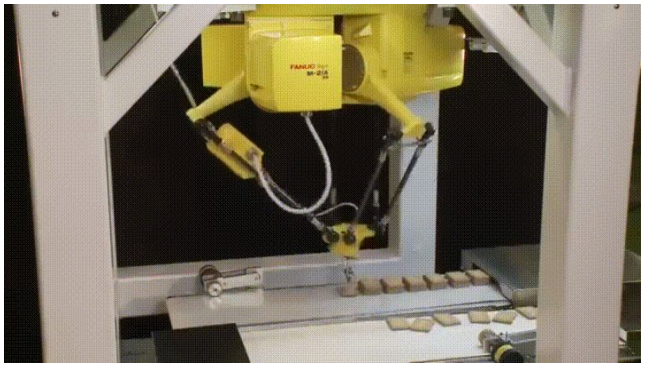
1. **Structured Pick-and-Place**
   - ➢ Structured Environment = Engineered to simplify a task -- arranging objects in a predictable position + eliminating clutter

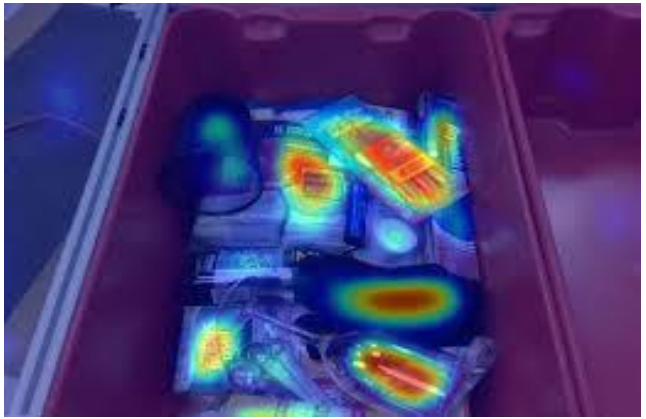


Pick-and-Place Manipulation

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

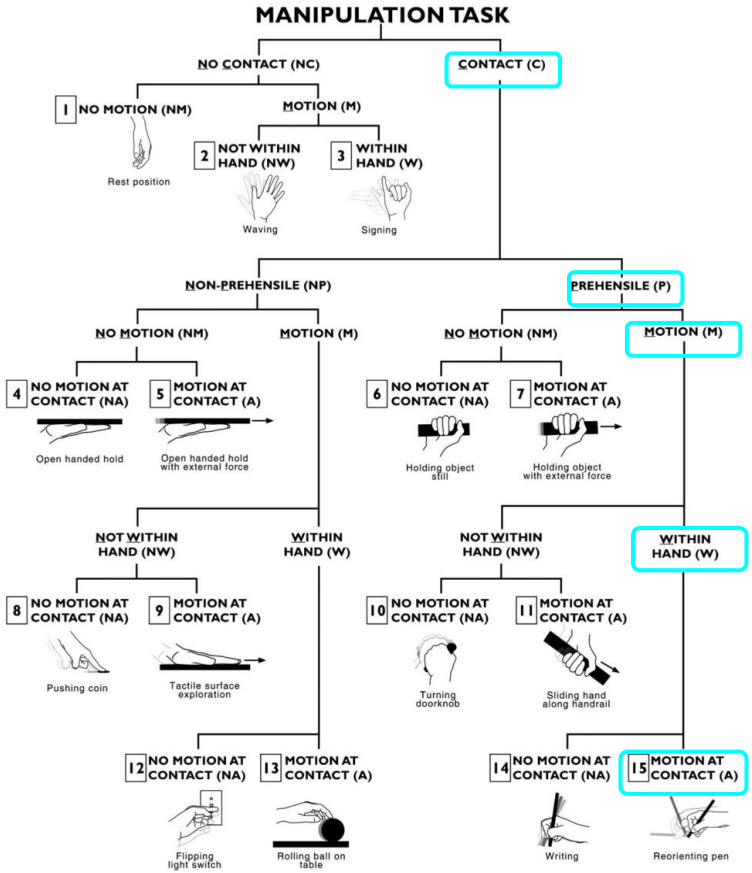
1. **Structured Pick-and-Place**
   - ➢ Structured Environment = Engineered to simplify a task -- arranging objects in a predictable position + eliminating clutter
2. **Unstructured Pick-and-Place**
   - ➢ Path Planning
   - ➢ General-Purpose Grippers
   - ➢ Grasp + Placement Pose Planning





In-Hand Manipulation

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



Bullock et al. 2013

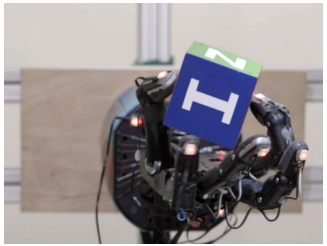
Grasping + Manipulation:
Relating to the Two Papers

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:

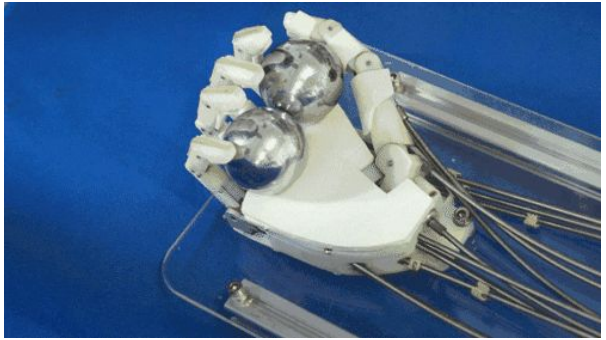
1. Defining Manipulation
2. Training Method

1. Defining Manipulation: Key Challenge

The Regrasp

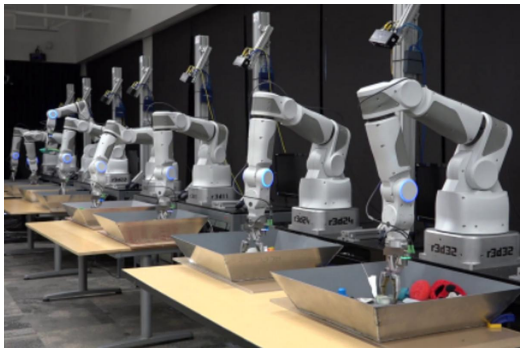
- **Pick and Place** → because the object is held rigidly, in order to regrasp, the object has to be put down and picked up with a new grasp

- **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

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

Overview: Motivation + Background

Motivation:

- A key goal is to create robots with more **generalizable** grasping + manipulation skills to allow more meaningful applications

Background:

- **Grasping:** Key Considerations
  1. Grasp Synthesis
  2. Means of Grasping
  3. Prior Object Knowledge
  4. Object Features
  5. Object-Grasp Representation
  6. TASK!!!
- **Manipulation**: Key Types
  1. Pick-and-Place (Unstructured + Structured) → Google Paper
  2. In-Hand Manipulation → OpenAI Paper
- **Key Differences**
  1. Defining Manipulation
  2. Training Method

Overview: Motivation + Background

Motivation:

- A key goal is to create robots with more **generalizable** grasping + manipulation skills to allow more meaningful applications

Background:

- **Grasping:** Key Considerations
  1. Grasp Synthesis
  2. Means of Grasping
  3. Prior Object Knowledge
  4. Object Features
  5. Object-Grasp Representation
  6. TASK!!!
- **Manipulation**: Key Types
  1. Pick-and-Place (Unstructured + Structured) → Google Paper
  2. In-Hand Manipulation → OpenAI Paper
- **Key Differences**
  1. Defining Manipulation
  2. Training Method

Overview: Motivation + Background

Motivation:

- A key goal is to create robots with more **generalizable** grasping + manipulation skills to allow more meaningful applications

Background:

- **Grasping:** Key Considerations
  1. Grasp Synthesis
  2. Means of Grasping
  3. Prior Object Knowledge
  4. Object Features
  5. Object-Grasp Representation
  6. TASK!!!
- **Manipulation**: Key Types
  1. Pick-and-Place (Unstructured + Structured) → Google Paper
  2. In-Hand Manipulation → OpenAI Paper
- **Key Differences**
  1. Defining Manipulation
  2. Training Method

Overview: Motivation + Background

Motivation:

- A key goal is to create robots with more **generalizable** grasping + manipulation skills to allow more meaningful applications

Background:

- **Grasping:** Key Considerations
  1. Grasp Synthesis
  2. Means of Grasping
  3. Prior Object Knowledge
  4. Object Features
  5. Object-Grasp Representation
  6. TASK!!!
- **Manipulation**: Key Types
  1. Pick-and-Place (Unstructured + Structured) → Google Paper
  2. In-Hand Manipulation → OpenAI Paper
- **Key Differences**
  1. Defining Manipulation
  2. Training Method

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

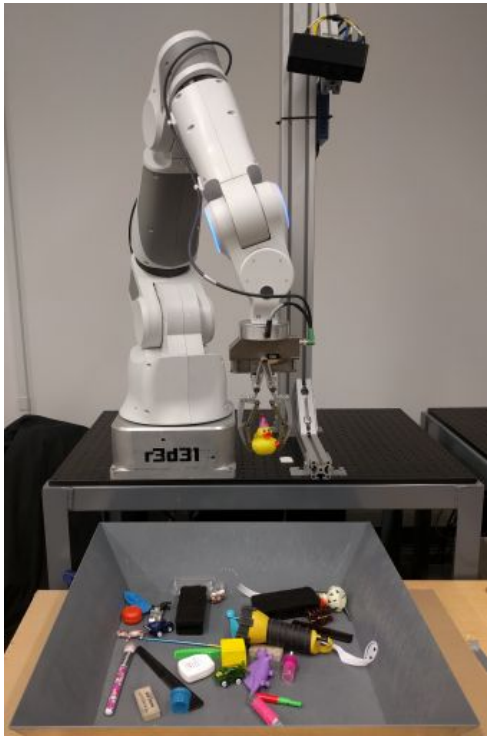
Learning **Hand-Eye Coordination** for Robotic Grasping with **Deep Learning** and **Large-Scale Data Collection**

Levine et al. (Google 2016)

Main Contributions

- Physical system: **7 DOF robot arm**, **2 finger gripper**, and an **uncalibrated monocular camera**

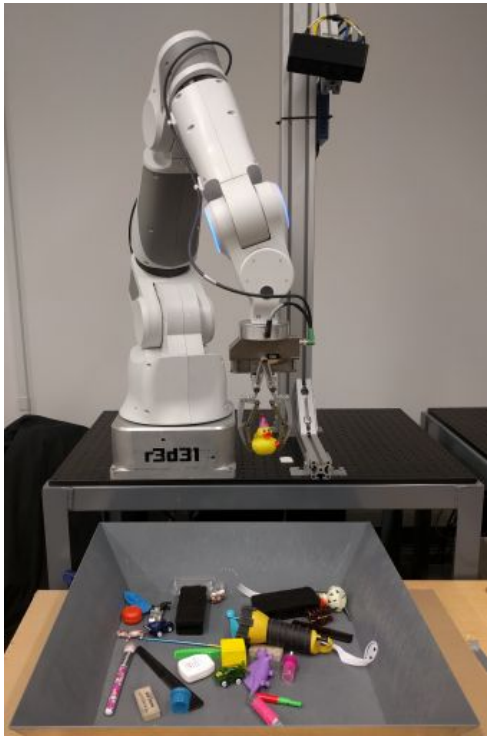
1. Novel **CNN architecture** that predicts grasping success
2. **Continuous visual servoing function** controls the robot
3. **Large-scale data collection** framework for robotic grasps



Main Contributions

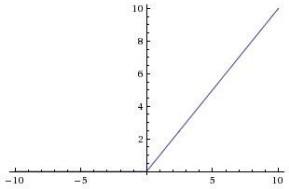
- Physical system: **7 DOF robot arm**, **2 finger gripper**, and an **uncalibrated monocular camera**

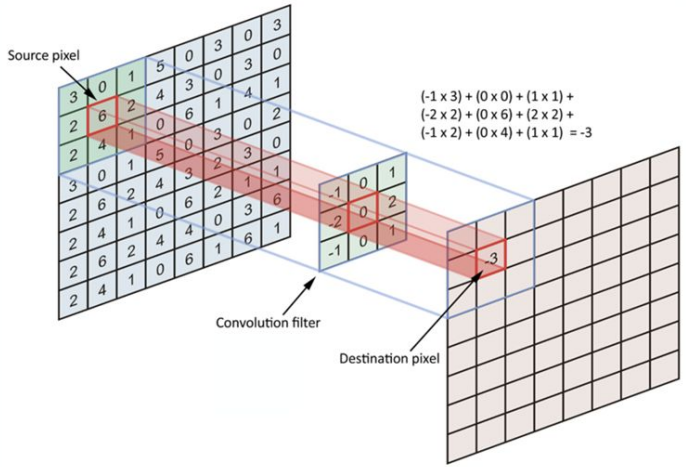
1. Novel **CNN architecture** that predicts grasping success
2. **Continuous visual servoing function** controls the robot
3. **Large-scale data collection** framework for robotic grasps

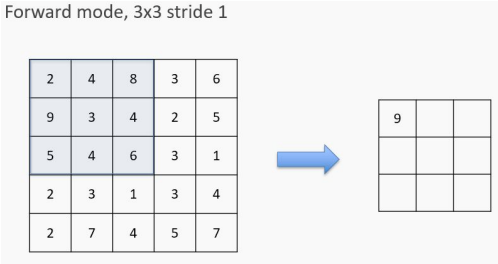


Convolutional Neural Network (CNN) Refresher

- 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 $a(\mathbf{x}) = max(\mathbf{x}, 0)$

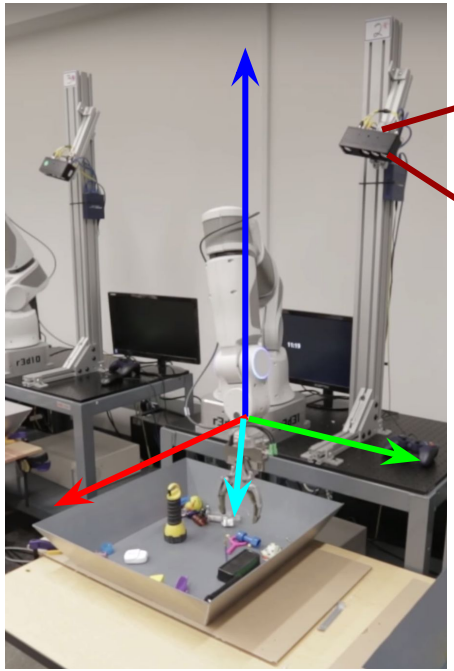






Images Credit: CS 109b Lecture Notes, Spring 2019

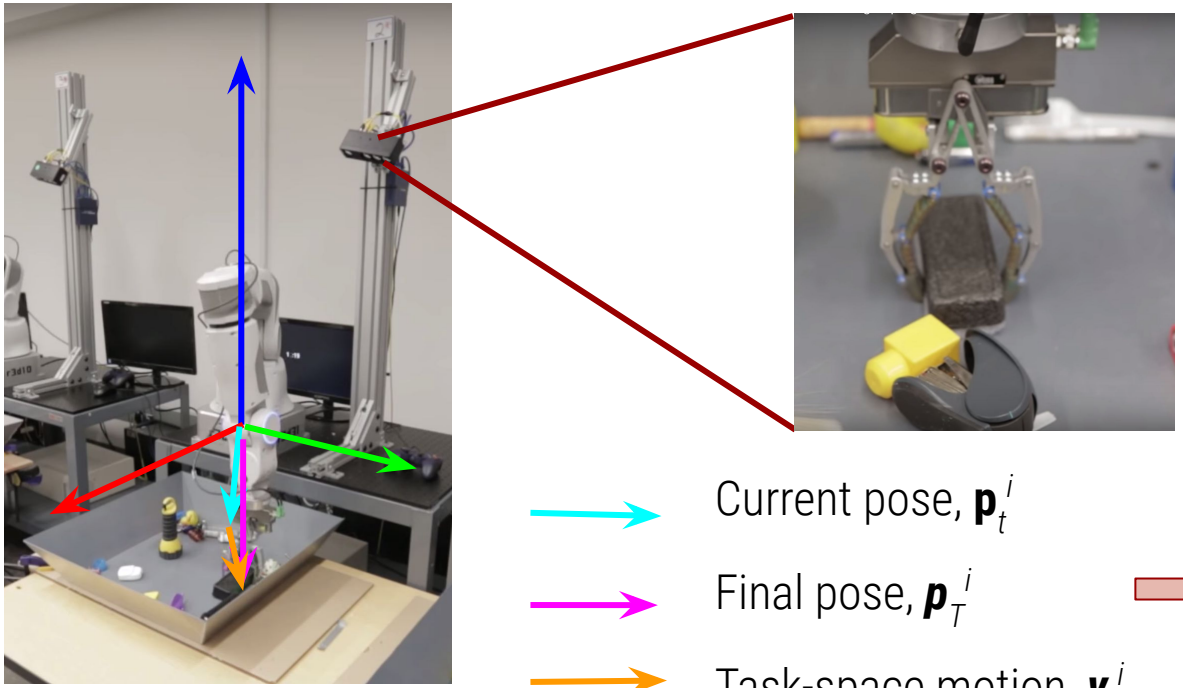
Grasp Sample Setup: Current Time Step, *t*



$\mathbf{I}_t^i$

Grasp attempt *i*

Grasp Sample Setup: Final Time Step, *T*



Current pose, $\mathbf{p}_t^i$

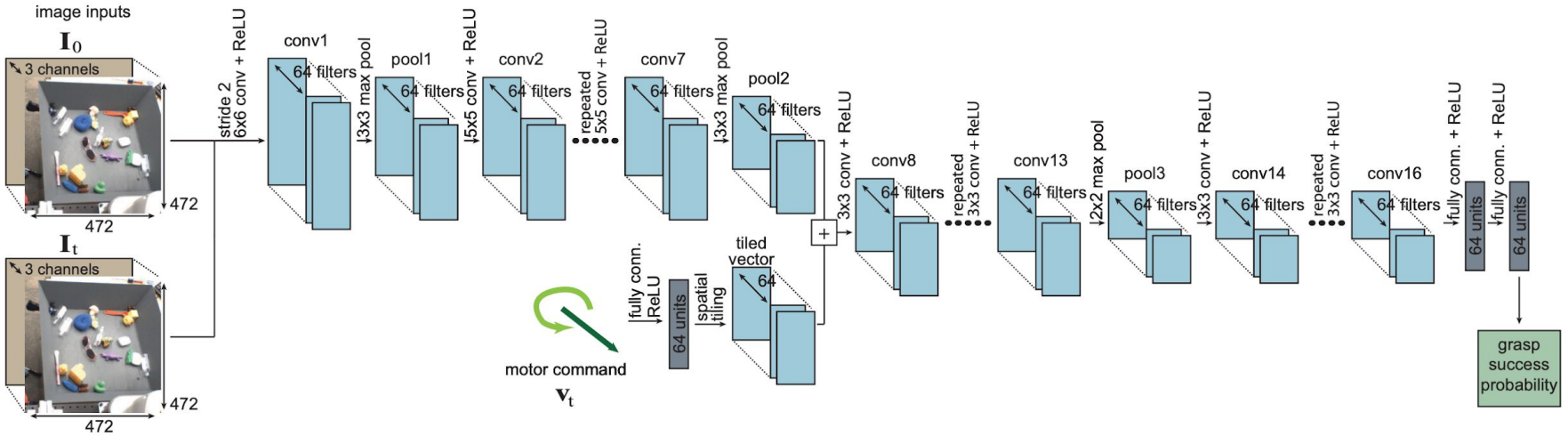
Final pose, $\boldsymbol{p}_T^i$

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

Label, $\mathbf{l}_t^i$

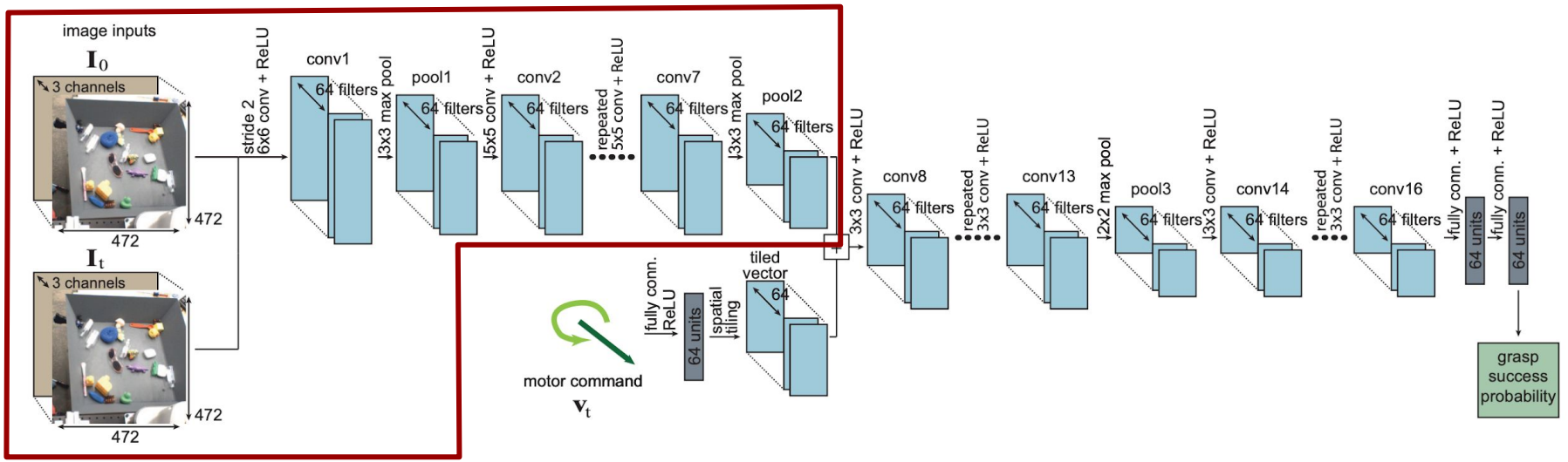
$(\mathbf{l}_t^i, \mathbf{p}_T^i - \mathbf{p}_t^i, \mathbf{l}_t^i)$

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



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

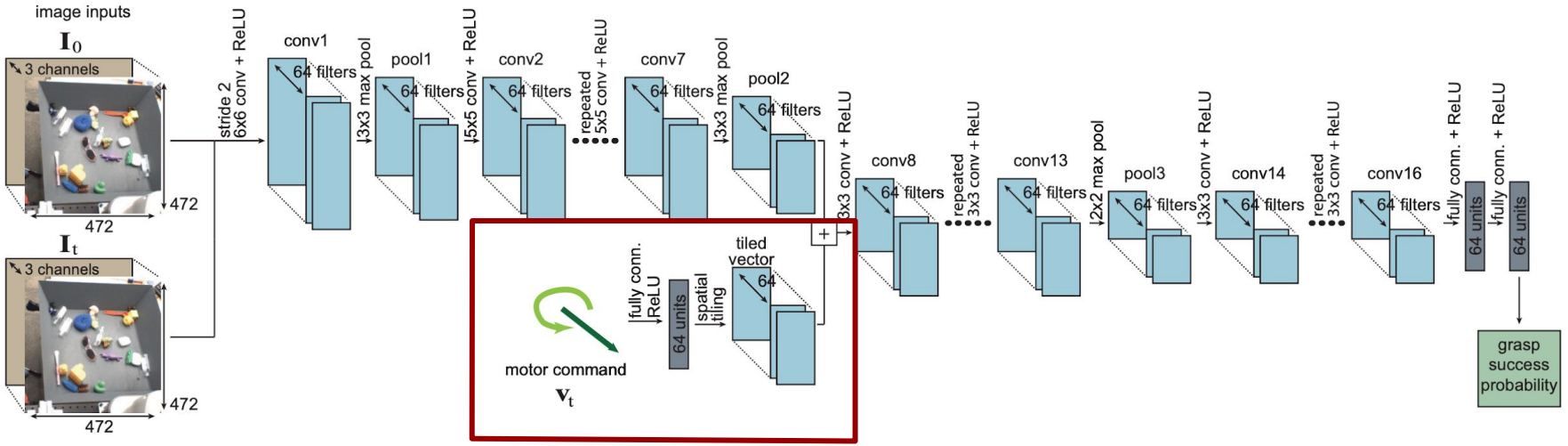
Neural Network Architecture for $g(\mathbf{I}_t, \mathbf{v}_t)$



Input $\mathbf{I}_t$ and $\mathbf{I}_0$

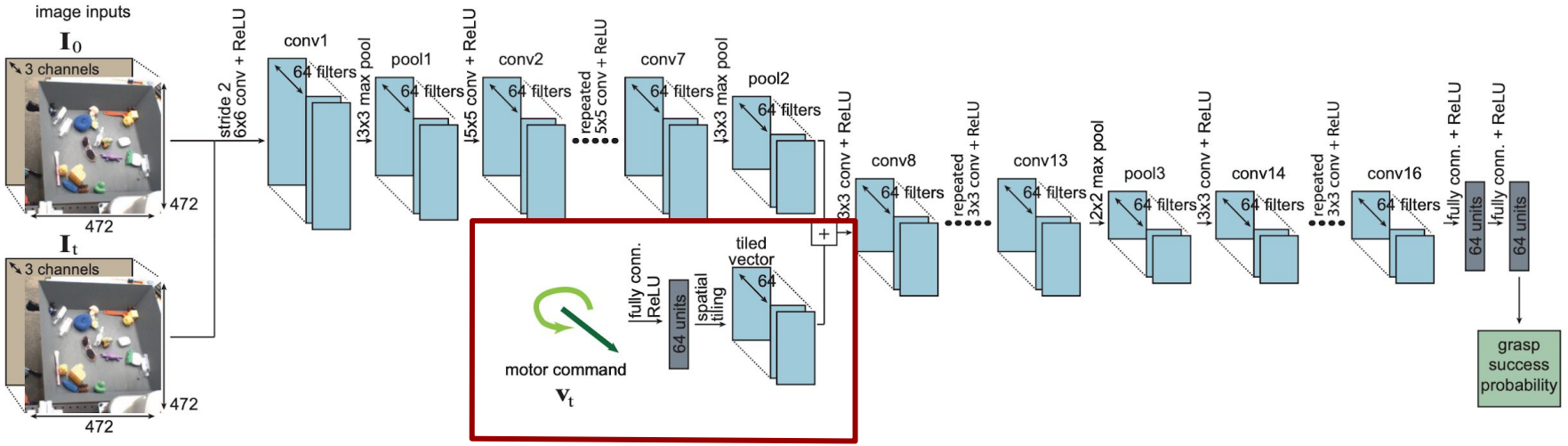
5 convolutional layers with batch normalization, followed by max pooling

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



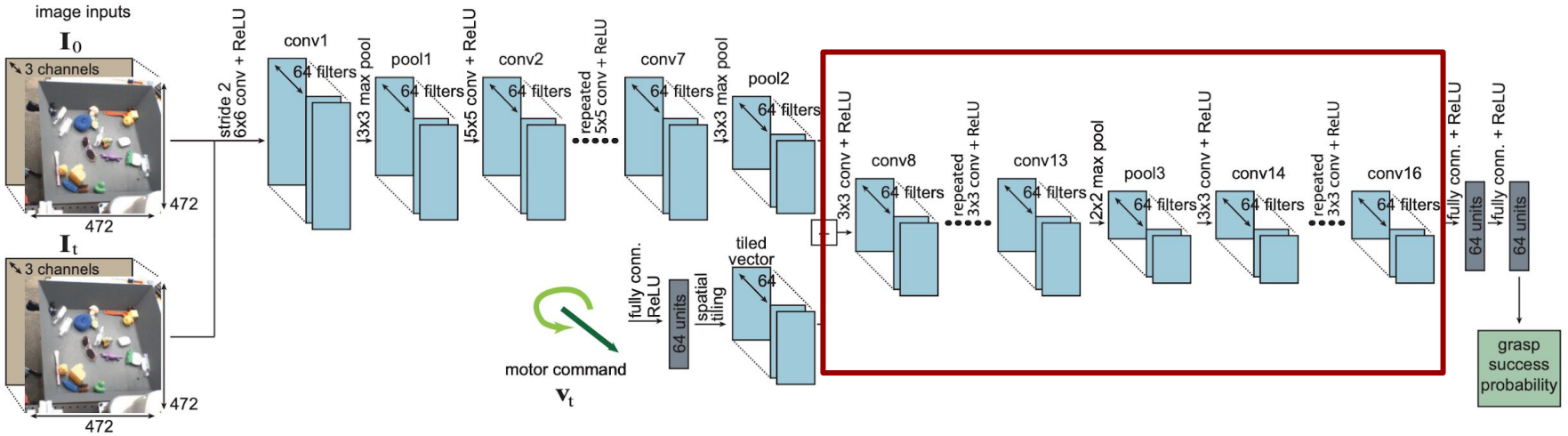
The motor command $\mathbf{v}_t = \mathbf{p}_T - \mathbf{p}_t$ is sent to 64 FC units and then concatenated with visual activations

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



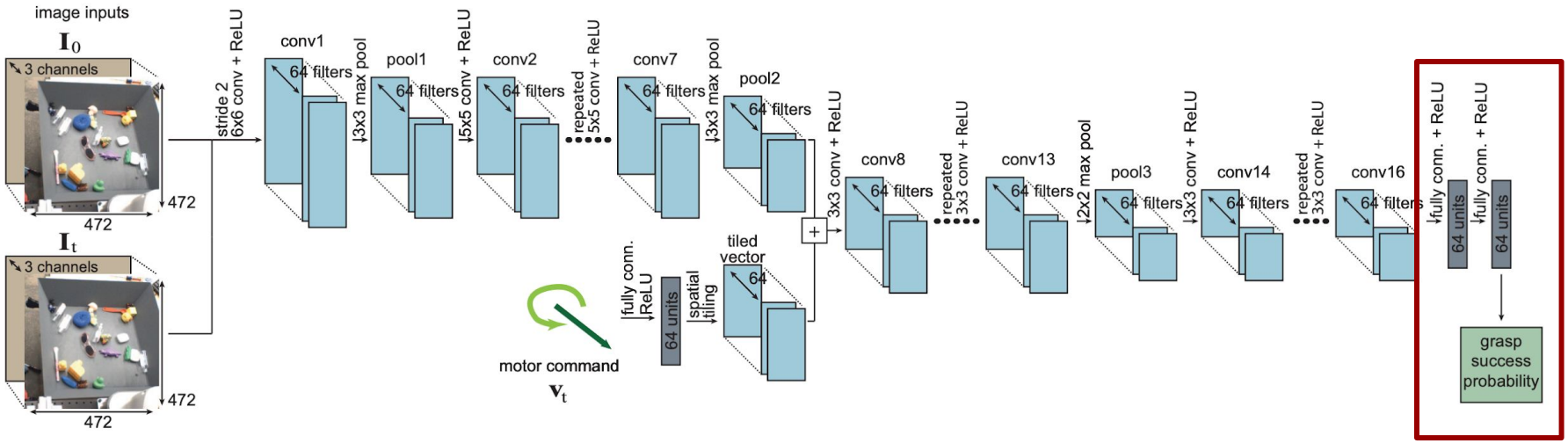
Is it just standard CNN architecture to add in the vector after the image has been processed for a few steps, or was there some underlying reason why the authors chose to do that?

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

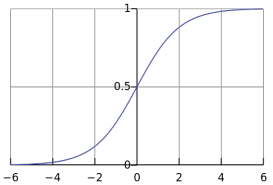


The combined activations are fed through another 9 CNN layers

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

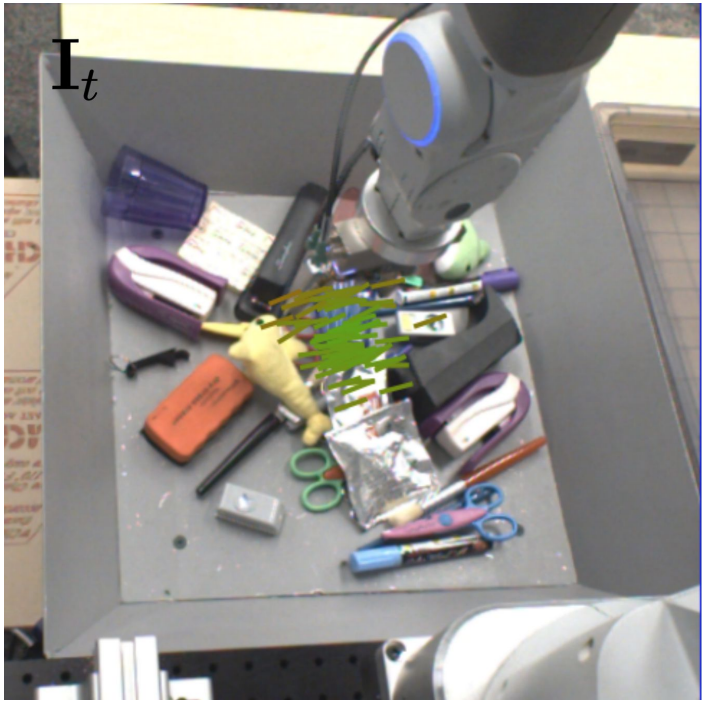


The output features are fed into 2 fully connected layers with sigmoid activation to predict grasp success probability



$$A = \frac{1}{1+e^{-x}}$$

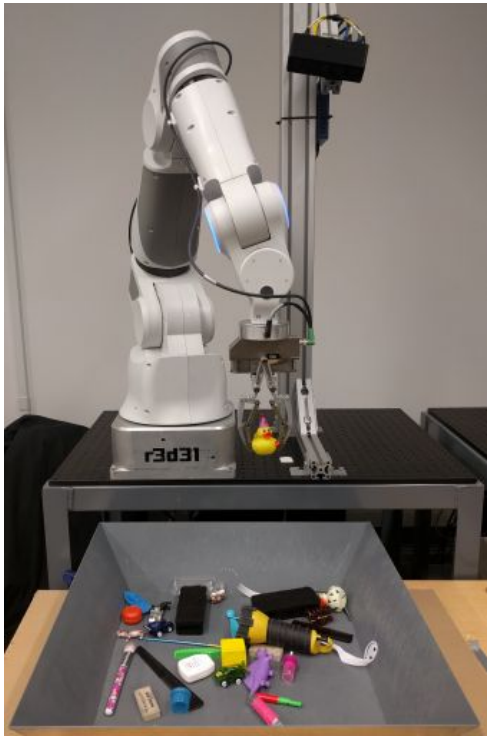
Example $\mathbf{I}_t$



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

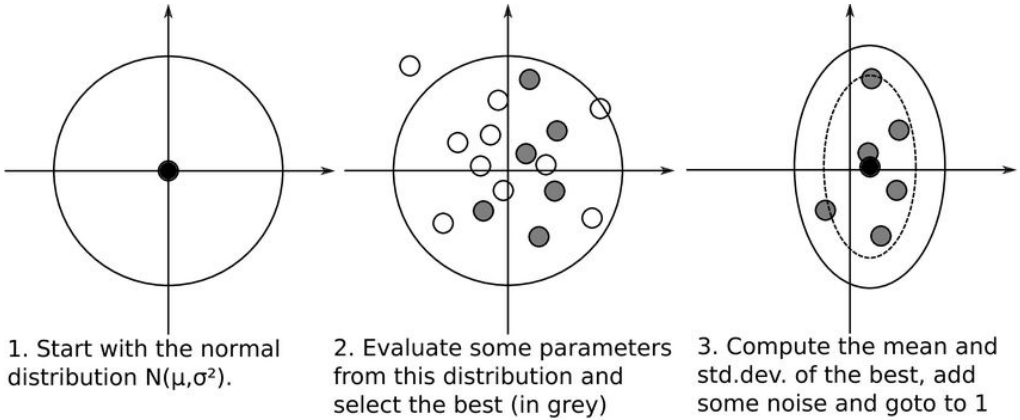


Servoing Mechanism $f(\mathbf{I}_t)$ - Algorithm

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 $p <= 0.5$, 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 $f(\mathbf{I}_t)$, 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**

1. Novel **CNN architecture** that predicts grasping success
2. **Continuous visual servoing function** controls the robot

} Parallels with RL

3. **Large-scale data collection** framework for robotic grasps

Parallels with Reinforcement Learning

Recall: Q function

- $Q(\mathbf{s}, \mathbf{a})$ is a measure of the overall expected reward assuming the agent is in state **s** and performs action **a**, and then continuous playing until the end of an episode by following some policy π

$$Q(s,a) = \mathbf{E}\left[\sum_{n=0}^{N} \gamma^n r_n\right]$$

Parallels with Reinforcement Learning

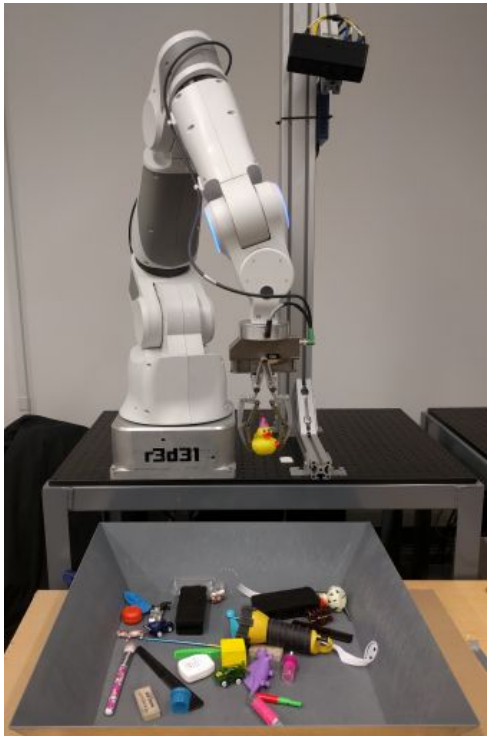
The *g* network is similar to a Q-function in RL, but not quite the same

- If **T = 2**, then ***g* really is a Q function**
  - Policy is the servoing mechanism $f(\mathbf{I}_t)$
  - Reward function is 1 when the grasp succeeds and 0 otherwise
  - Fitted Q iteration: repeatedly deploying the latest grasp network, collecting additional data, and refitting the network
- 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
  - Standard reinforcement learning: prediction is based on the current state and motor command, which in this case is given by $p_{t+1} - p_t$ (doesn't use p_T)

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**!
- **6-14** robot arms running in parallel over 2 months and made **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

Evaluation Questions

1. Does continuous servoing significantly improve grasping accuracy and success rate?
2. How well does our learning-based system perform when compared to alternative approaches?

Evaluation Questions

1. Does **continuous servoing** significantly improve **grasping accuracy and success rate**?

Compared approach to open-loop method that:

- Observes the scene prior to the grasp
- Extracts image patches
- Chooses the patch with the highest probability of a successful grasp
- Uses a known camera calibration to move the gripper to that location

Evaluation Questions

2. How well does our **learning-based system** perform when compared to alternative approaches?

Compared approach to:

- **Random baseline method**
- **Hand-engineered grasping system** that uses depth images and heuristic positioning of the fingers

Experimental Results

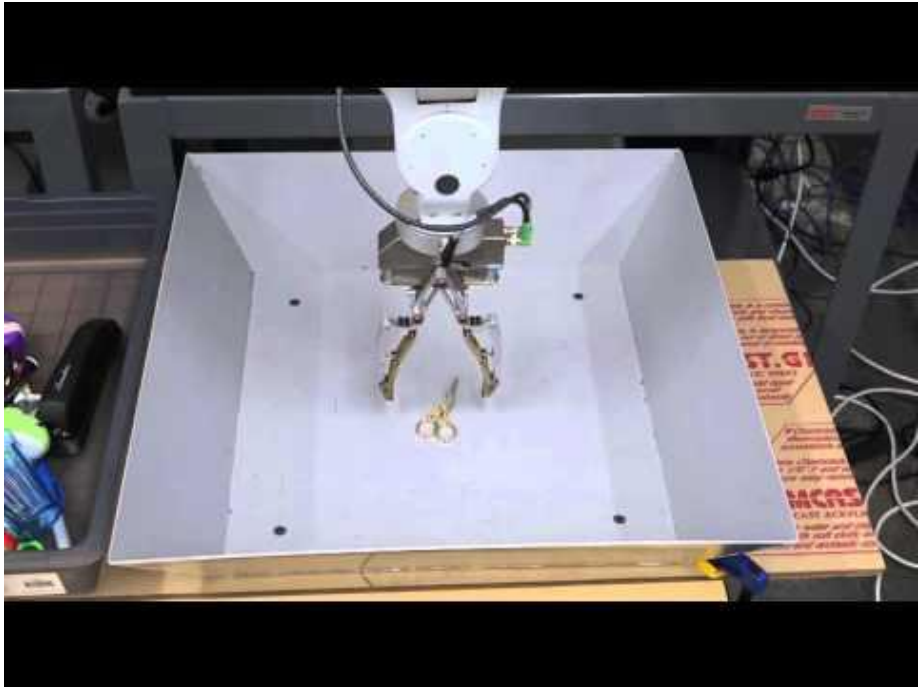
- The network was trained on a **diverse set of household and office objects**
  - These had varying size, shape, hardness, and texture
- The robot was tested on **novel objects** (unseen in training) with broadly similar attributes
- The most important test was success rate in moving objects from one bin to another (grasping without replacement)
  - The 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

Experimental Results

Without replacement	First 10 ($N = 40$)	First 20 ($N = 80$)	First 30 ($N = 120$)
Random	67.5%	70.0%	72.5%
Hand-designed	32.5%	35.0%	50.8%
Open loop	27.5%	38.7%	33.7%
Our method	**10.0%**	**17.5%**	**17.5%**
With replacement	Failure rate ($N = 100$)		
Random	69%		
Hand-designed	35%		
Open loop	43%		
Our method	**20%**		

Without replacement	First 10 $N=40$	First 20 $N=80$	First 30 $N=120$
12%: $M = 182,249$	52.5%	45.0%	47.5%
25%: $M = 407,729$	30.0%	32.5%	36.7%
50%: $M = 900,162$	25.0%	22.5%	25.0%
100%: $M = 2,898,410$	**10.0%**	**17.5%**	**17.5%**

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

Second Experiment: 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

Testing Transfer Between Robots

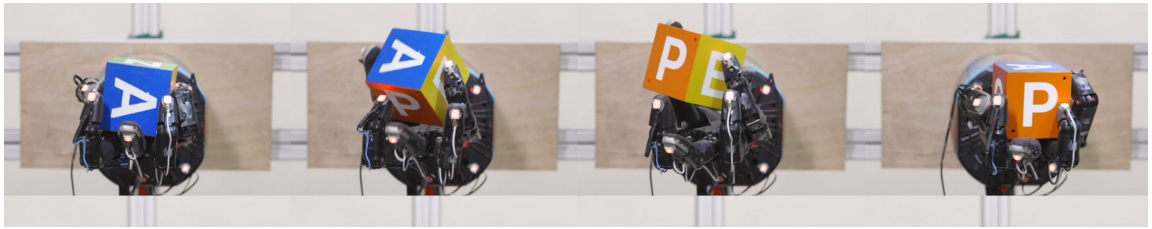
- Evaluating whether grasping data collected by one type of robot could be used to improve the grasping proficiency of a different robot
- 900,000 additional grasp attempts using a different robotic manipulator with a substantially larger variety of objects

Hardware Setup of Transfer Testing

Learning Dexterous In-Hand Manipulation

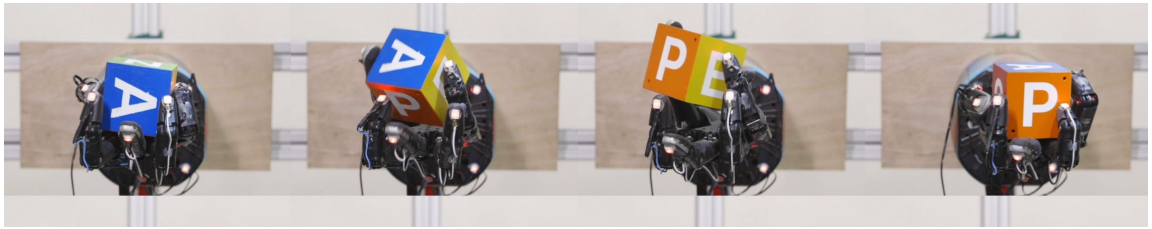
Overview

- 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



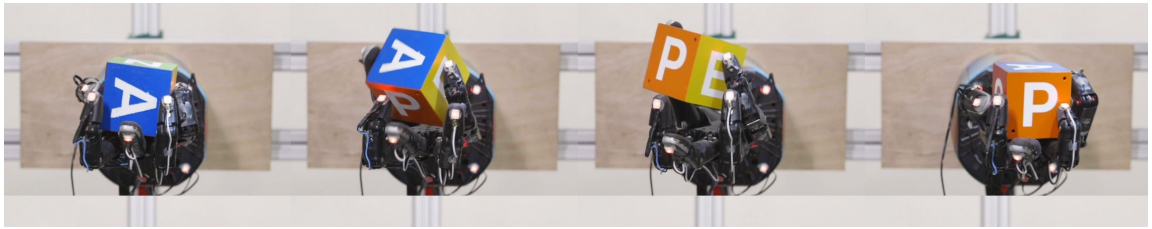
Overview

- 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



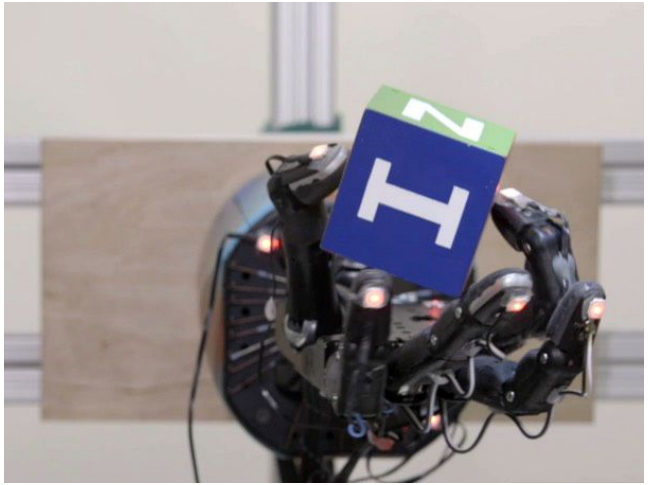
Overview

- 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



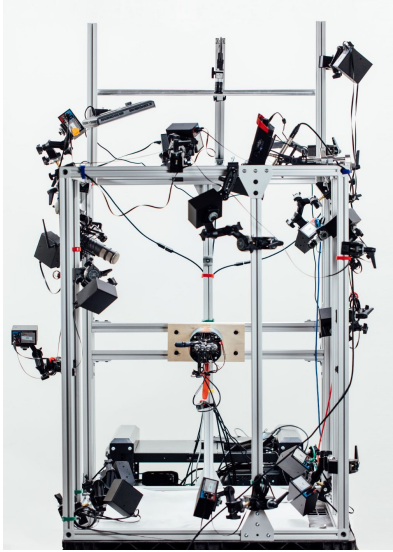
Task Overview

- 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

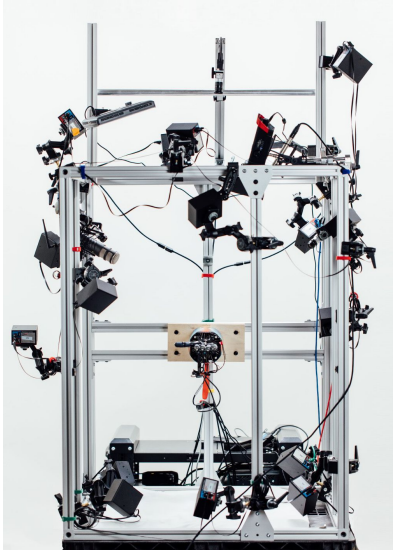
- Shadow Dexterous Hand robot
  - Most advanced humanoid hand robot on the market, released 2005
  - 24 degrees of freedom; 20 controls with tendon pairs, DC motors
  - Hand includes tactile sensors, but these were NOT used!
  - Cost is not public, discussion thread ~$100k in 2005
- Motion capture system: 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

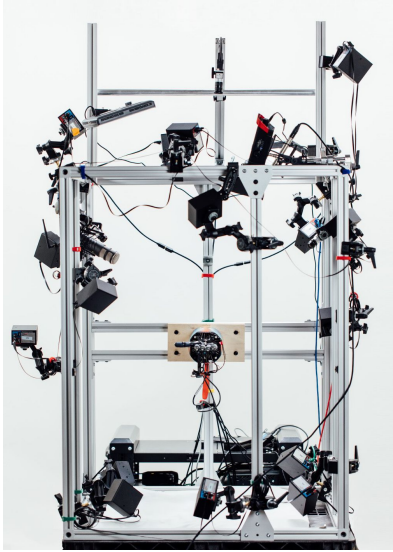
- Shadow Dexterous Hand robot
  - Most advanced humanoid hand robot on the market, released 2005
  - 24 degrees of freedom; 20 controls with tendon pairs, DC motors
  - Hand includes tactile sensors, but these were NOT used!
  - Cost is not public, discussion thread ~$100k in 2005
- Motion capture system: 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

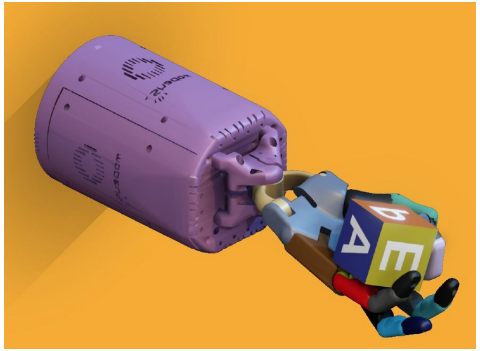
- Shadow Dexterous Hand robot
  - Most advanced humanoid hand robot on the market, released 2005
  - 24 degrees of freedom; 20 controls with tendon pairs, DC motors
  - Hand includes tactile sensors, but these were NOT used!
  - Cost is not public, discussion thread ~$100k in 2005
- Motion capture system: 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

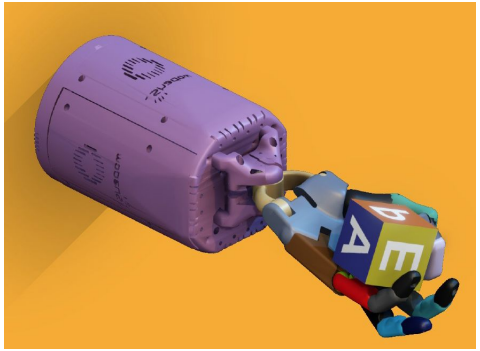
- Simulation uses model of S.D.H. built in OpenAI Gym
- Physics engine provided by MuJoCo
  - Tendons are ignored; simplified as a single torque
  - Rigid body simulator; doesn't handle deformable or semi-rigid finger tips, which are critical for modeling contact points
- Scene rendering with Unity video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



Rendering of a simulated environment.

Simulation Overview

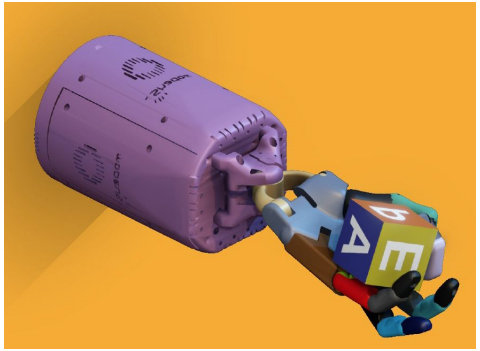
- Simulation uses model of S.D.H. built in OpenAI Gym
- Physics engine provided by MuJoCo
  - Tendons are ignored; simplified as a single torque
  - Rigid body simulator; doesn't handle deformable or semi-rigid finger tips, which are critical for modeling contact points
- Scene rendering with Unity video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

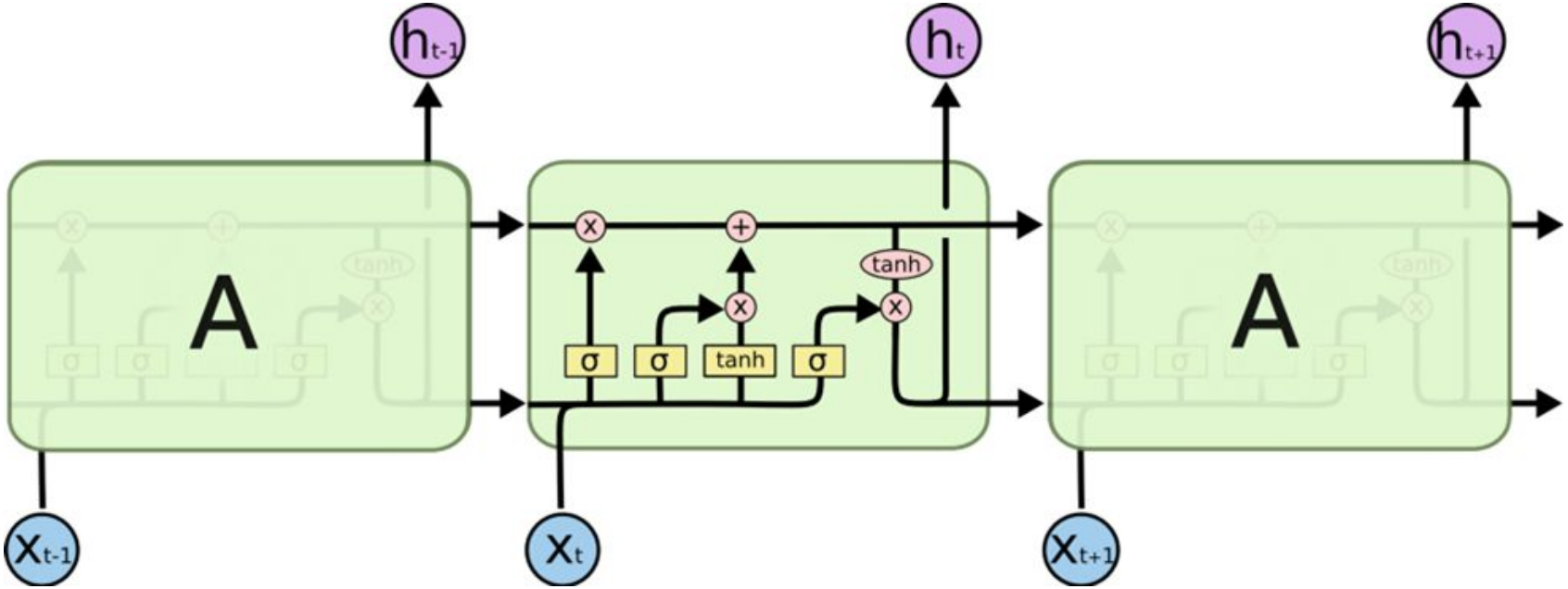
$$L_{\text{PPO}} = \mathbb{E} \min \left(\frac{\pi(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)} \hat{A}_t^{\text{GAE}}, \text{clip} \left(\frac{\pi(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t^{\text{GAE}} \right)$$

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

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

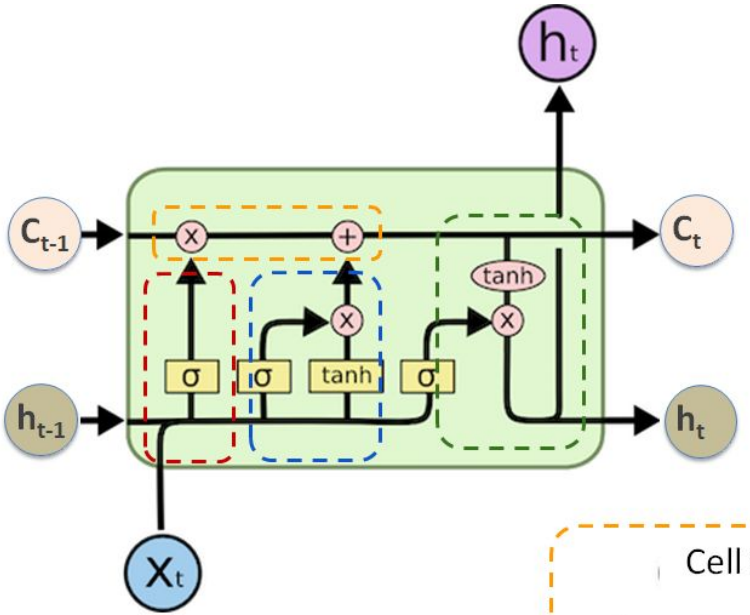
PPO encourages the policy to shift actions with better than average payoffs, but the clipping discourages changes that are too large ("proximal")

LSTM Overview



Images Credit: CS 109b Lecture Notes, Spring 2019

LSTM: Detail of One Cell



Forget Gate

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Output Gate

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Input Gate

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_f)$$

Cell State

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

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 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

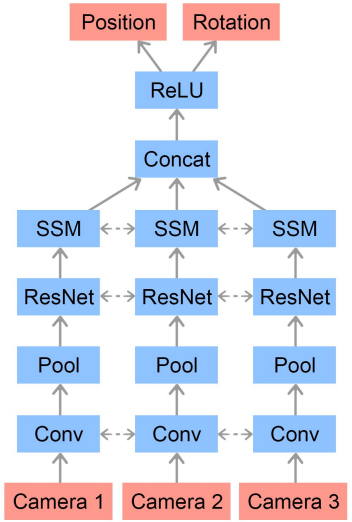
- Policy is a recurrent neural network (RNN) with memory (LSTM)
  - As this network process inputs, it updates its internal state (memory)
  - Trained using Proximal Policy Optimization (PPO) with policy and value networks
  - Asymmetric actor-critic method; critic has more inputs than actor
- State 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 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

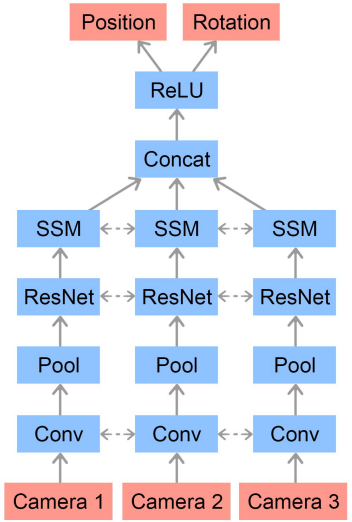
Estimating State with Vision

- 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



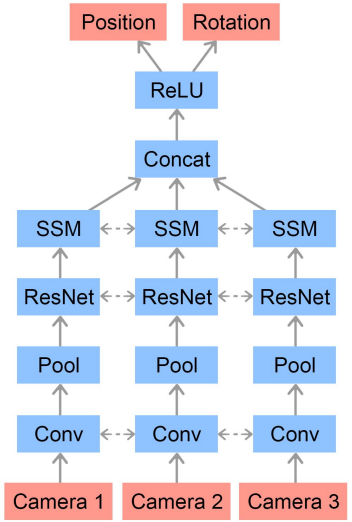
Estimating State with Vision

- 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



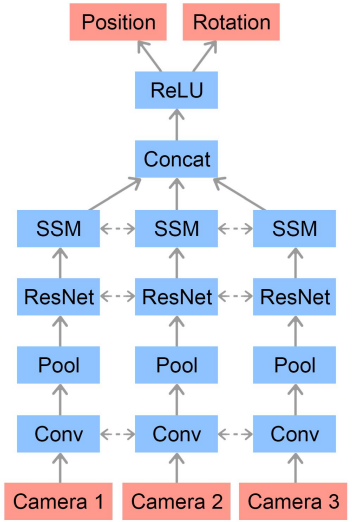
Estimating State with Vision

- 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

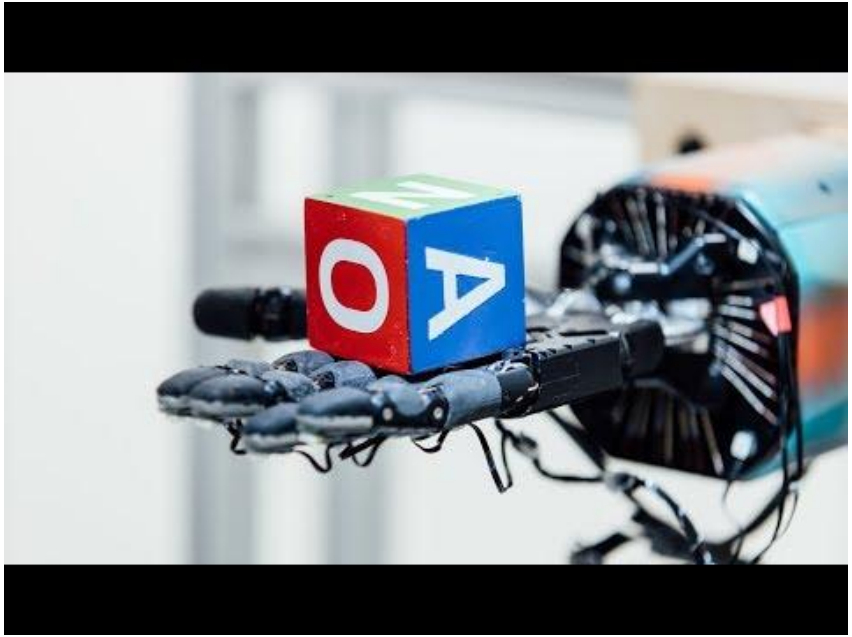
- The robot learned this task with performance comparable to a person
  - It also learned grasp types similar to those used by people
- Benchmark experiment: 80 attempts, ending after 80 seconds or 1 drop

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		
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



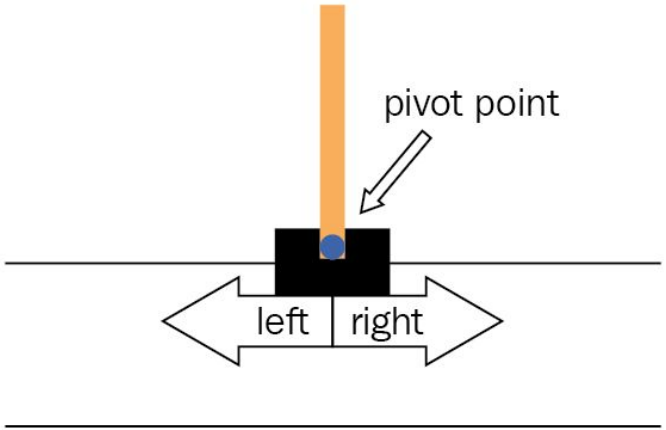
Stupid Human Tricks: Block Manipulation

- 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

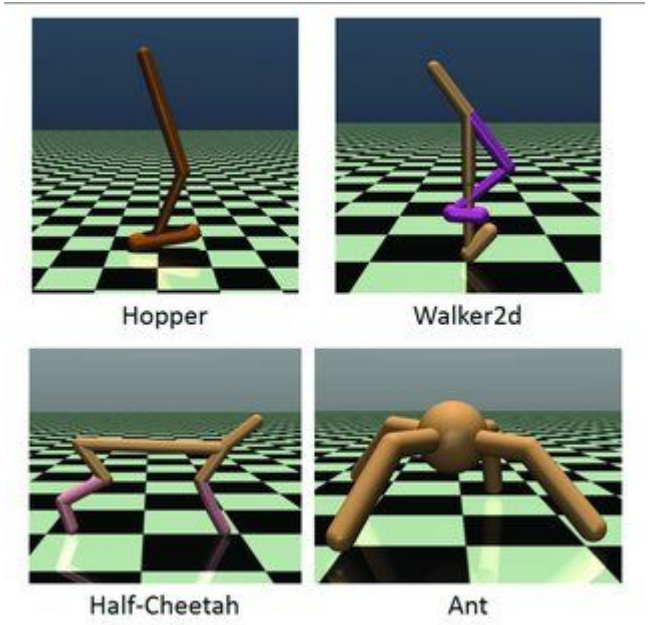
- OpenAI gym is a python environment for RL
  - https://gym.openai.com/docs/, https://github.com/openai/gym
  - Very user friendly; pip install gym and you are off to the races
- Here is a minimal program demonstrating it

```
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



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

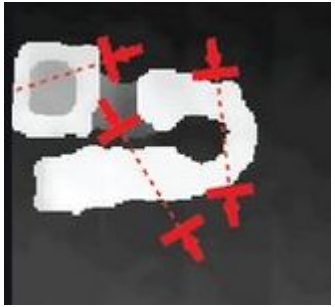
- Green AI is a recent trend
  - Participants considering energy usage in model training, not just final task performance
- Some ML achievements have come at the cost of huge energy usage
  - OpenAI estimated to have spent $30 million (!) training its agent to play DOTA II
  - It cost an estimated $2 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

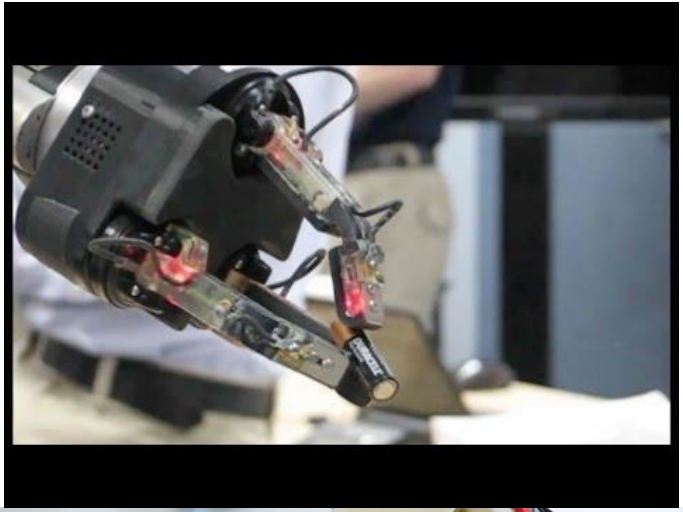
Green AI and Energy Usage in ML

- 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





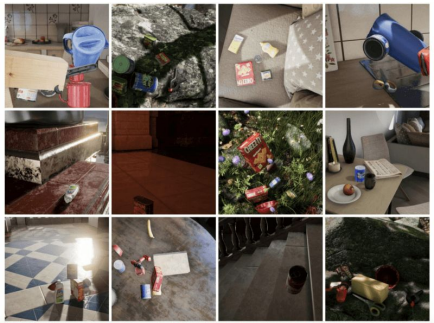
- Mechanical Hands
  - Simplistic
    - Antipodal - 2 parameters: Angle, location, then just close
  - Complicated:
    - Shadow-Hand
      - Tendon-based, very painful to re-thread
    - Similar method in Yale OpenHand
  - How to make them robust?
- Robust Hand
  - iHY

Hardware Systems

- Tactile Sensors
  - Gelsight
  - Barometric
- Robot arm
  - Collaborative robots: Way safer and lighter
    - No longer need cages
    - IK / path planning can be ignored
    - ..but slow / low-torque

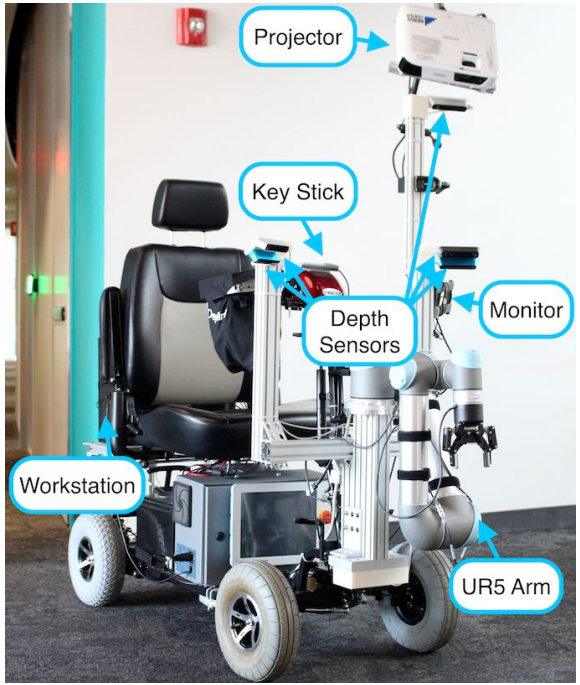
Novelty of These Results -- Limitations & Future Work

- Work Since Then (software)
  - Domain randomization
    - FAT
  - Hierarchical (e.g. dishwasher)
  - Grasping in cluttered / occluded environments
- Future work: dynamic movements?
    - https://giant.gfycat.com/AcceptablePoorAsiaticlesserfreshwaterclam.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?



UR5: slow , fast

Future Work - Humans!

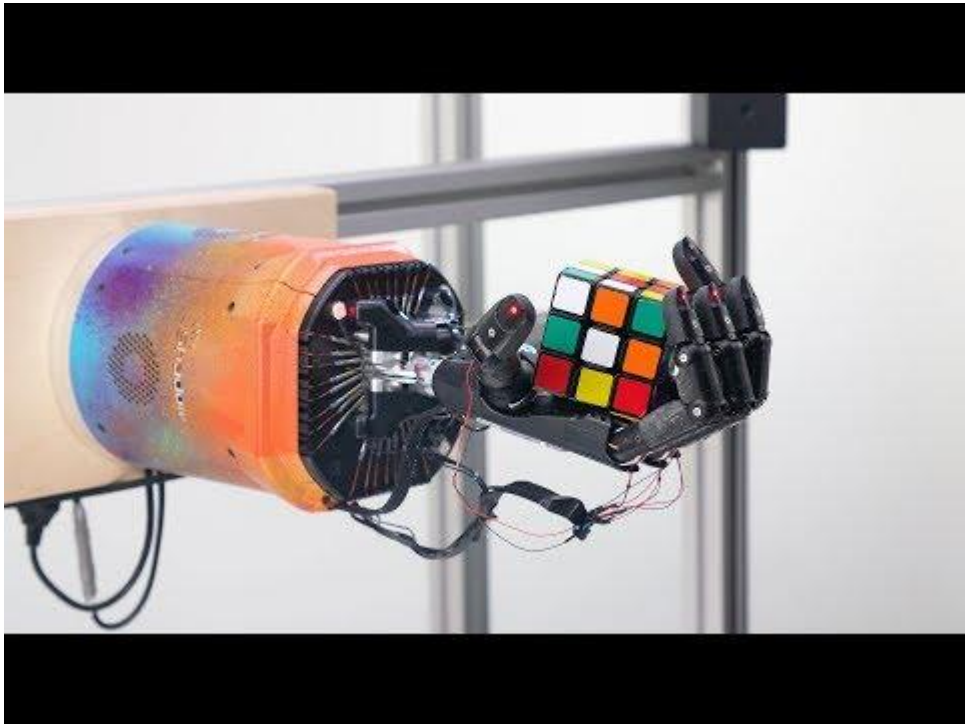
Tossing bricks with orientation

Late-Breaking Results!

The biggest challenge we faced was to create environments in simulation diverse enough to capture the physics of the real world. Factors like friction, elasticity and dynamics are incredibly difficult to measure and model for objects as complex as Rubik's Cubes or robotic hands and we found that domain randomization alone is not enough.

To overcome this, we developed a new method called *Automatic Domain Randomization* (ADR), which endlessly generates progressively more difficult environments in simulation.[2] This frees us from having an accurate model of the real world, and enables the transfer of neural networks learned in simulation to be applied to the real world.

ADR starts with a single, nonrandomized environment, wherein a neural network learns to solve Rubik's Cube. As the neural network gets better at the task and reaches a performance threshold, the amount of domain randomization is increased automatically. This makes the task harder, since the neural network must now learn to generalize to more randomized environments. The network keeps learning until it again exceeds the performance threshold, when more randomization kicks in, and the process is repeated.



OpenAI Software

Emergent meta-learning

We believe that meta-learning, or learning to learn, is an important prerequisite for building general-purpose systems, since it enables them to quickly adapt to changing conditions in their environments. The hypothesis behind ADR is that a memory-augmented networks combined with a sufficiently randomized environment leads to *emergent meta-learning*, where the network implements a learning algorithm that allows itself to rapidly adapt its behavior to the environment it is deployed in.[3]

More concretely, we are looking for signs where our policy updates its belief about the true transition probability $P(s_{t+1} \mid s_t, a_t)$ as it observes data over time.

In other words, when we say "meta-learning", what we really mean is learning to learn about the environment dynamics. Within other communities, this is also called on-line system identification. In our case though, this is an emergent property.

Overall Summary

OpenAI Hardware

Xiaomi cube

(a) The components of the Giiker cube.

(b) An assembled Giiker cube while charging.

Figure 5: We use an off-the-shelf Giiker cube but modify its internals (subfigure a, right) to provider higher resolution for the 6 face angles. The components from left to right are (i) bottom center enclosure, (ii) lithium polymer battery, (iii) main PCBa with BLE, (iv) top center enclosure, (v) cubelet bottom, (vi) compression spring, (vii) contact brushes, (viii) absolute resistive rotary encoder, (ix) locking cap, (x) cubelet top. Once assembled, the Giiker cube can be charged with its "headphones on" (right).

OpenAI Sensing

7 State Estimation from Vision

As in [79], the control policy described in Section 6 receives object state estimates from a vision system consisting of three cameras and a neural network predictor. In this work, the policy requires estimates for all six face angles in addition to the position and orientation of the cube.

Note that the absolute rotation of each face angle in $[-\pi, \pi]$ radians is required by the policy. Due to the rotational symmetry of the stickers on a standard Rubik's cube, it is not possible to predict these absolute face angles from a single camera frame; the system must either have some ability to track state temporally[12] or the cube has to be modified.

We therefore use two different options for the state estimation of the Rubik's cube throughout this work:

1. **Vision only via asymmetric center stickers.** In this case, the vision model is used to produce the *cube position, rotation, and six face angles*. We cut out one corner of each center sticker on the cube (see Figure 13), thus breaking rotational symmetry and allowing our model to determine absolute face angles from a single frame. No further customizations were made to the Rubik's cube. We use this model to estimate final performance of a vision only solution to solving the Rubik's cube.
2. **Vision for pose and Giiker cube for face angles.** In this case, the vision model is used to produce the *cube position and rotation*. For the face angles, we use the previously described customized Giiker cube (see Section 3) with built-in sensors. We use this model for most experiments in order to not compound errors of the challenging face angle estimation from vision only with errors of the policy.

Since our long-term goal is to build robots that can interact in the real world with arbitrary objects, ideally we would like to fully solve this problem from vision alone using a standard Rubik's cube (i.e. without any special stickers). We believe this is possible, though it may require either more extensive work on a recurrent model or moving to an end-to-end training setup (i.e. where the vision model is learned jointly with the policy). This remains an active area of research for us.

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!