TossingBot by Zeng et al. 2020

Learning to Throw Arbitrary Objects With Residual Physics

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

11. Januar 2024

Outline

Motivation
What is TossingBot?

Motivation

Tossing compared to Mini-Golf

Similarities

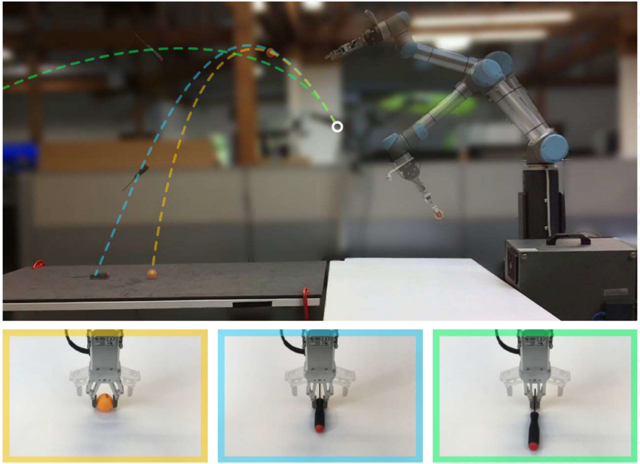
1. Object manipulation
2. Single action
3. Dynamics estimation
4. UR5

Differences

1. Different objects - different course
2. Grasping subtask

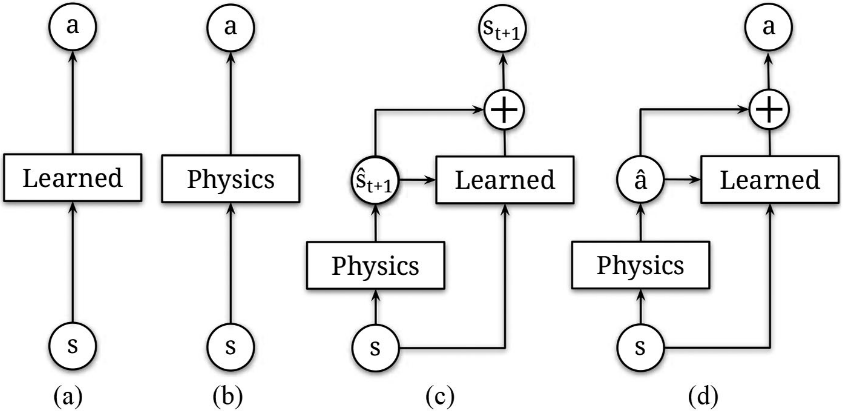
Motivation

Projectile Trajectories and Grasping



Approach

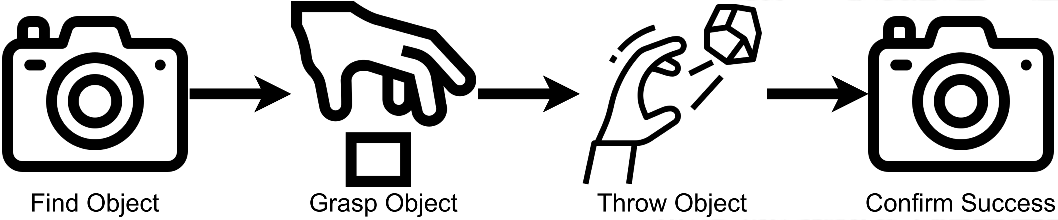
(a) Data-driven
(b) Analytical
(c) Hybrid (predict future state)
(d) Hybrid (analytical + learning)



Learning Approaches

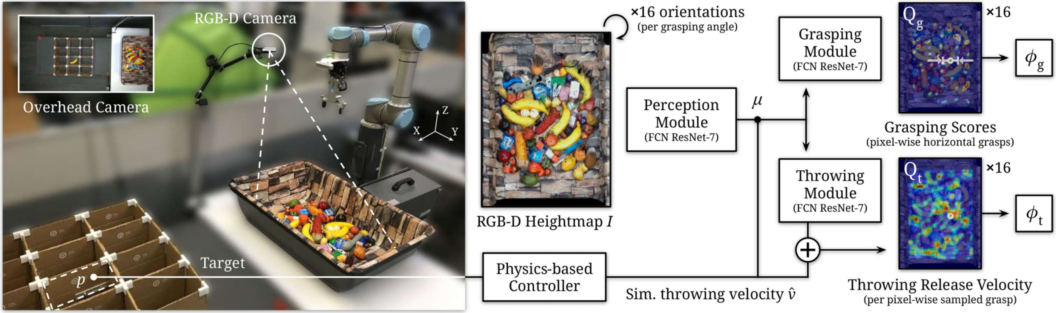
Residual physics controller

1. Analytical model for estimate of control parameter
2. Residuals for compensation of unknown dynamics



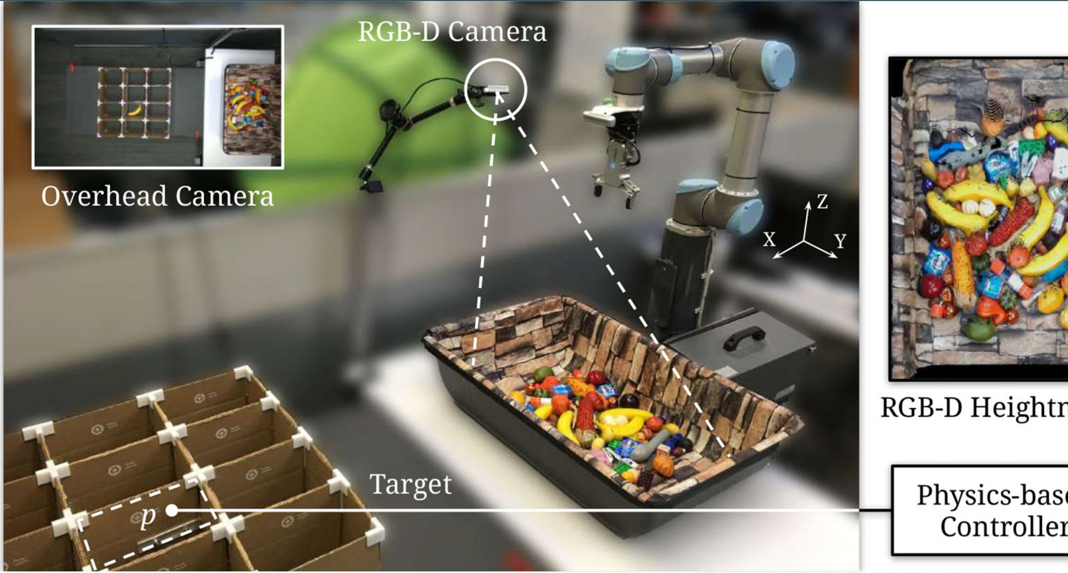
Task components of TossingBot

Model

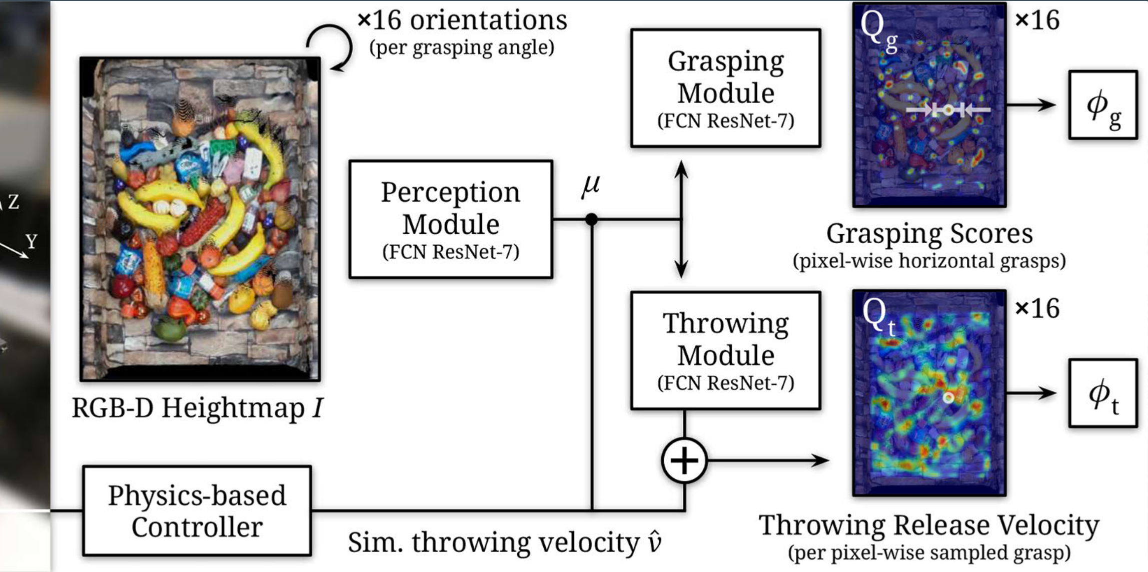


Model overview

Model


Overhead Camera
RGB-D Camera
Z
X
Y
Target
p
RGB-D Heightr
Physics-base
Controller

Model



×16 orientations
(per grasping angle)
RGB-D Heightmap I
Perception Module
(FCN ResNet-7)
μ
Grasping Module
(FCN ResNet-7)
Throwing Module
(FCN ResNet-7)
Q_g
×16
ϕ_g
Grasping Scores
(pixel-wise horizontal grasps)
Q_t
×16
ϕ_t
Throwing Release Velocity
(per pixel-wise sampled grasp)
Physics-based Controller
Sim. throwing velocity $\hat{v}$
Z
Y

Model
Primitives

Grasping Primitive

- Input: $\phi_g = (x, \theta)$
- 3-D location $x = (x_x, x_y, x_z)$
- Orientation θ (around direction of gravity)

Throwing Primitive

- Endeffector trajectory
- Input: $\phi_t = (r, v)$
- Release position $r = (r_x, r_y, r_z)$
- Release velocity $v = (v_x, v_y, v_z)$
- Axis between fingers orthogonal to plane of object trajectory

- RGB-D heightmap
- Fixed-mounted overhead camera
- Project image onto 3-D pointcloud
- normalized

Perception Network

- Fully convolutional network ResNet-7
- convolutional layer $\rightarrow$ max pooling $\rightarrow$ residual block $\rightarrow$ max pooling $\rightarrow$ residual block $\rightarrow$ residual block
- residual block: 2 convolutional layers with bypass
- Output: spacial feature representation μ

Grasping Network

- Fully convolutional network ResNet-7
- Input: μ
- Output: probability map Q_g of grasp success
- Different grasping orientations by rotation of input heightmap
- 16 orientations
- highest probability pixel $\rightarrow$ position and orientation
- sample efficient

Model
Throwing Module

- Constrain release position r
  1. arial trajectory on same plane as on release
  2. fixed release distance and height to robot base
- Constrain release velocity v
  1. 45 deg upwards in direction of target p

Model

Physics-Based Controller

Standard Equation of Linear Projectile Motion

$$p = r + \hat{v}t + \frac{1}{2}at^2 \tag{1}$$

Assumptions

- Grasp at CoM
- Point particle
- No drag
- Release velocity v is tossing velocity (no spin)

p: landing location

r: release position

$\hat{v}$: release velocity

a: acceleration ($a_z = -9.8m/s^2$)

t: time

Model

Residual Physics-Based Controller

- Compensation for assumptions
- Residual value δ
- δ added to $\hat{v}$

Residual Network

- Fully convolutional network ResNet-7
- Input: μ
- Output: Q_t
- One-on-one correspondence between Q_g and Q_t
- Prediction of residual value δ_i

Learning

- All modules as one network f
- End-to-end training

y_i: Grasp success ground truth

$\overline{\delta}_i$: Residual ground truth

Loss function

$$\mathcal{L} = \mathcal{L}_g + y_i \mathcal{L}_t \tag{2}$$

- Binary cross-entropy error

$$\mathcal{L}_g = -(y_i \log q_i + (1 + y_i) \log(1 - q_i)) \tag{3}$$

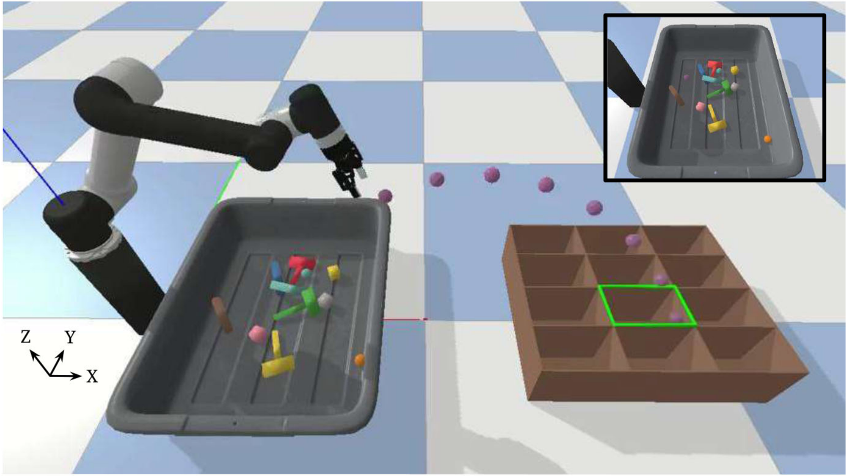
- Huber loss

$$\mathcal{L}_t = \begin{cases} \frac{1}{2}(\delta_i - \overline{\delta}_i)^2, & \text{for } |\delta_i - \overline{\delta}_i| < 1 \\ |\delta_i - \overline{\delta}_i| - \frac{1}{2}, & \textit{otherwise} \end{cases} \tag{4}$$

Learning
Self-Supervision

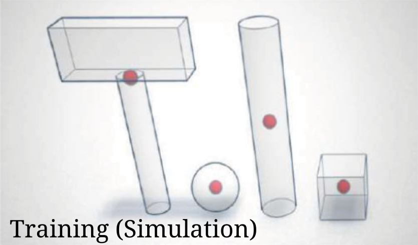
1. Capture visual input
2. Perform forward pass (get ϕ_g and ϕ_t)
3. Execute grasping
4. Execute throwing
5. Get ground truth y_i by success of throw
6. Approximate landing position $\hat{p}$
7. Ground truth residual $\overline{\delta}_i = ||v_{x,y}|| - ||\hat{v}_{x,y}||_{\hat{p}}$

Experiments
Simulation

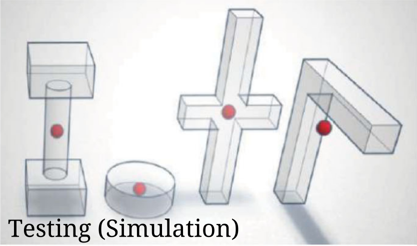


Experiments

Objects



Training (Simulation)

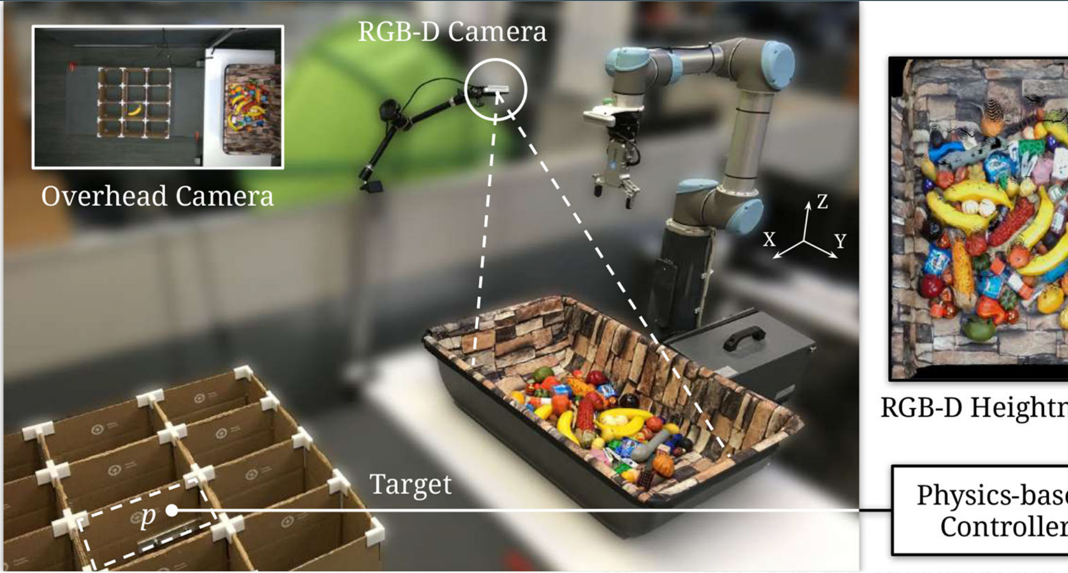


Testing (Simulation)

Training (Real)

Testing (Real)

Experiments



Results
Simulation

Method	Balls	Cubes	Rods	Hammers	Seen	Unseen
Regression	70.9	48.8	37.5	32.8	41.8	28.4
Regression-PoP	96.1	73.5	52.8	47.8	56.2	35.0
Physics-only	98.6	83.5	77.2	70.4	82.6	50.0
Residual-physics	**99.6**	**86.3**	**86.4**	**81.2**	**88.6**	**66.5**

Throwing performance simulation (Mean %)

Method	Balls	Cubes	Rods	Hammers	Seen	Unseen
Regression	99.4	99.2	89.0	87.8	95.6	69.4
Regression-PoP	99.2	98.0	89.8	87.0	96.4	70.6
Physics-only	99.4	99.2	87.6	85.2	96.6	64.0
Residual-physics	98.8	99.2	89.2	84.8	96.0	74.6

Grasping performance simulation (Mean %)

Results
Video

- `https://www.youtube.com/watch?v=f5Zn2Up2RjQ&t=04m39s`

Results
Real

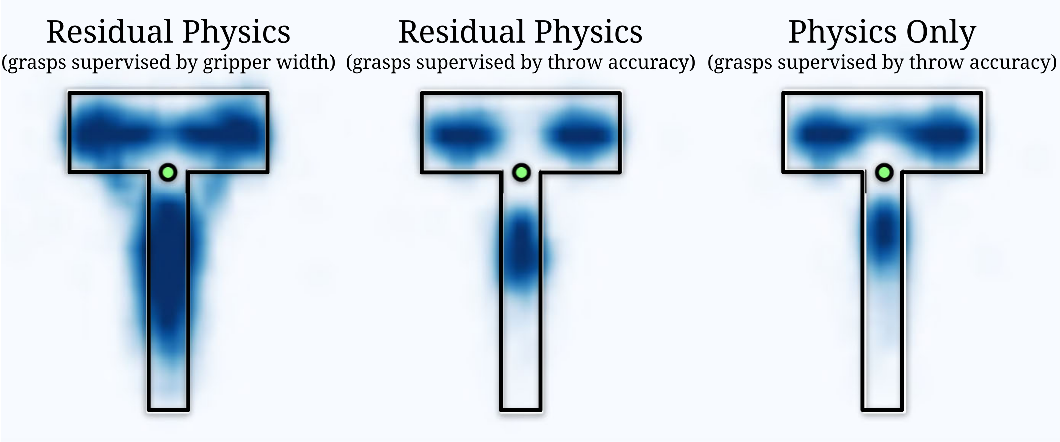
Method	Grasping Seen	Grasping Unseen	Throwing Seen	Throwing Unseen
Human-baseline	-	-	-	80.1±10.8
Regression-PoP	83.4	75.6	54.2	52.0
Physics-only	85.7	76.4	61.3	58.5
Residual-physics	86.9	73.2	**84.7**	**82.3**

Grasping and throwing performance real (Mean %)

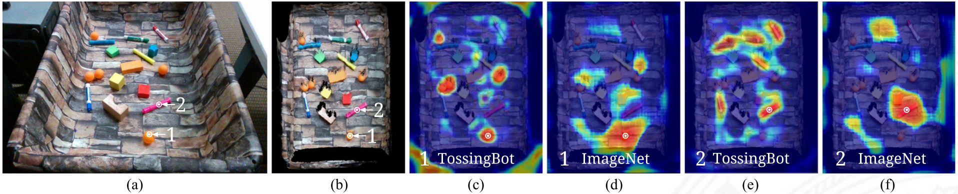
Method	Simulation	Real
Regression-PoP	26.5	32.7
Physics-only	79.6	62.2
Residual-physics	**87.2**	**83.9**

Throwing to unseen locations (Mean %)

Results
Grasping



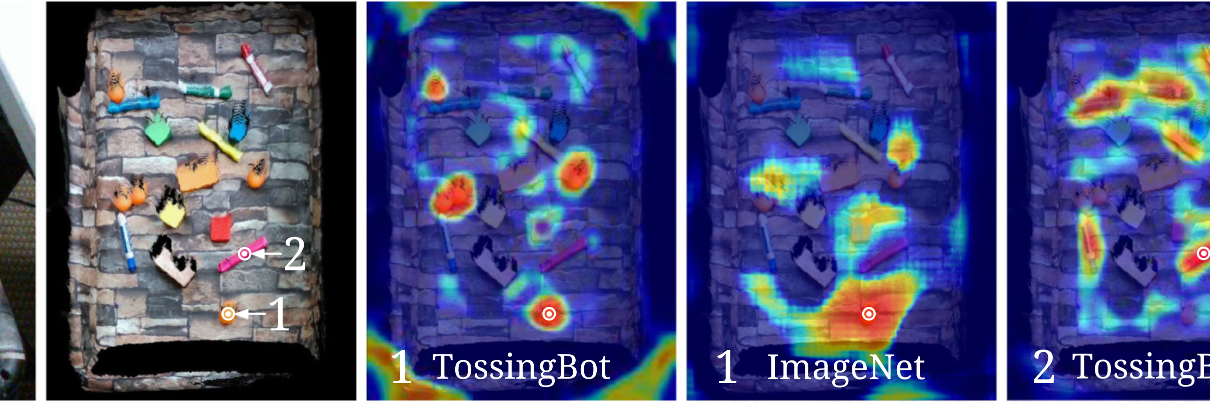
Results
Semantics



Emerging semantics from interaction with objects

- (a): Bin with objects
- (b): RGB-D view
- (c,e): Heatmap of pixel-wise distances
- (d,f): ResNet-18 pre-trained on ImageNet

Results
Semantics



(b) (c) (d) (e)

Project Takeaways

- Structure: Physics based guess and learning of residual
  1. Baseline solution
  2. Deep learning
- Motion primitive
- Training time of 14h
- Automatic reset
- ...

Conclusion

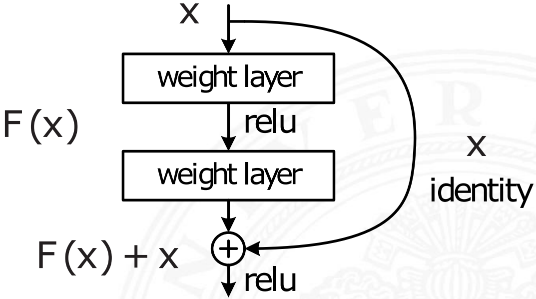
1. TossingBot: physics based estimate + learned residual
2. Robust grasping from downstream learning signal
3. Learning of implicit features by interaction with objects

Thank you for your attention

Thank you for your attention.

Are there any questions ?

- Degredation during training: saturation of accuracy →degrading accuracy
- Shortcut connections



Residual learning: building block (He et al. 2015)

Huber Loss

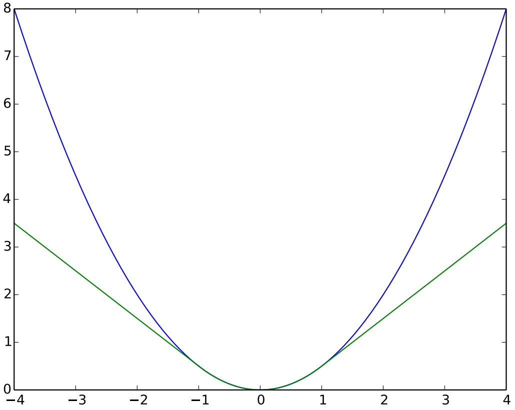
Huber Loss

$$\mathcal{L}_1(\delta, \overline{\delta}) = \begin{cases} \frac{1}{2}(\delta_i - \overline{\delta}_i)^2, & \text{for } |\delta_i - \overline{\delta}_i| < 1 \mid \text{Mean squared error for small errors} \\ |\delta_i - \overline{\delta}_i| - \frac{1}{2}, & \textit{otherwise} \mid \text{Mean absolute error for large errors} \end{cases} \tag{5}$$

AI 2022

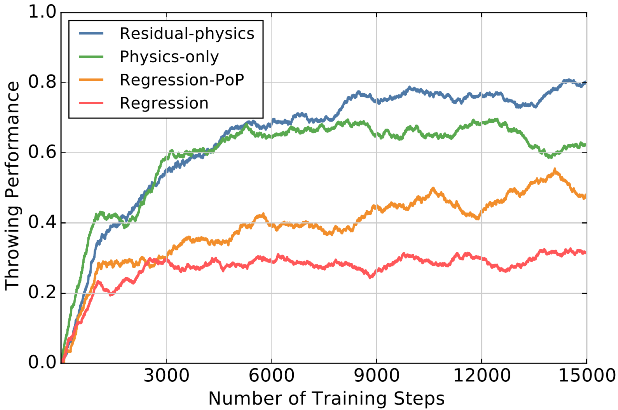
- Less weight on outlies compared to MSE
- Higher loss for errors below 1
- Small loss for small error

Huber Loss



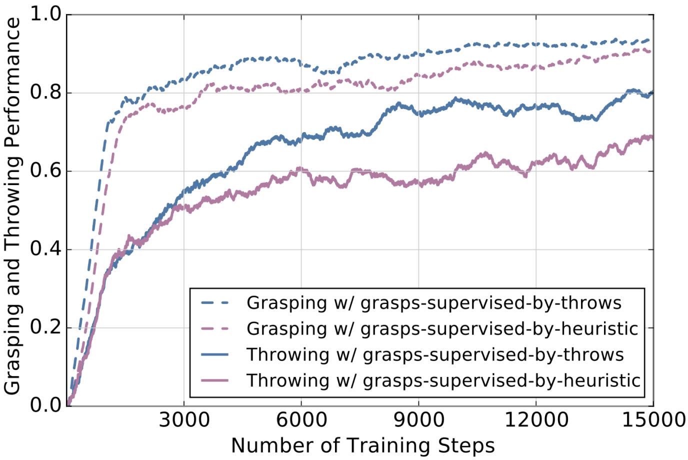
Huber loss (green) and mean squared error (blue) (*Huber loss* 2023)

Training



Training performance in simulation

Grasp Training



Grasping success in simulation

AI, Practicus (Feb. 11, 2022). *Understanding the 3 most common loss functions for Machine Learning Regression*. Medium. URL: https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3 (visited on 01/10/2024).

He, Kaiming et al. (Dec. 10, 2015). *Deep Residual Learning for Image Recognition*. version: 1. arXiv: 1512.03385 [cs]. URL: http://arxiv.org/abs/1512.03385 (visited on 01/10/2024).

Huber loss (Nov. 9, 2023). In: *Wikipedia*. Page Version ID: 1184310836. URL: https://en.wikipedia.org/w/index.php?title=Huber_loss&oldid=1184310836 (visited on 01/10/2024).

Zeng, Andy et al. (Aug. 2020). "TossingBot: Learning to Throw Arbitrary Objects With Residual Physics". In: *IEEE Transactions on Robotics* 36.4. Conference Name: IEEE Transactions on Robotics, pp. 1307–1319. ISSN: 1941-0468. DOI: 10.1109/TRO.2020.2988642. URL: https://ieeexplore.ieee.org/abstract/document/9104757 (visited on 12/13/2023).

Diagram Icon Sources

- https://www.flaticon.com/free-icons/camera
- https://www.flaticon.com/free-icons/grab
- https://www.flaticon.com/free-icons/throw