TossingBot

Berk Güngör

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

24. November 2022

Outline

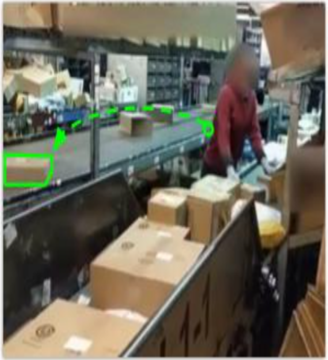
Motivation

- Use of human robot interaction in academia and industry.
- Learning to grasp arbitrary objects.
- Relating grasping and throwing activities.
- Increasing the maximum reach range of robots.

Introduction

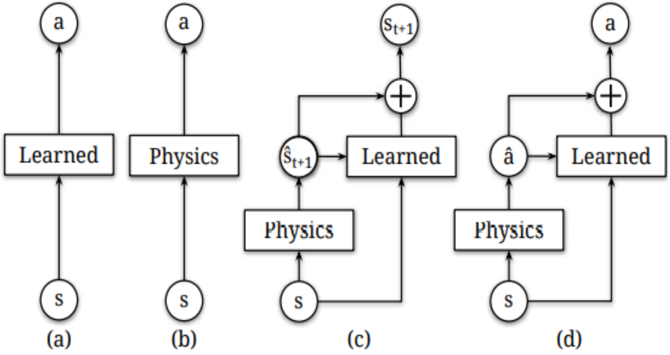
- TossingBot learns to grasp arbitrary objects from an unstructured bin and to throw them into target boxes.
- Throw it and increase the capabilities of manipulator.
- Joint learning of grasping and throwing policies with a deep neural network.
- Residual learning of throw release velocities.

- Can robots toss ?



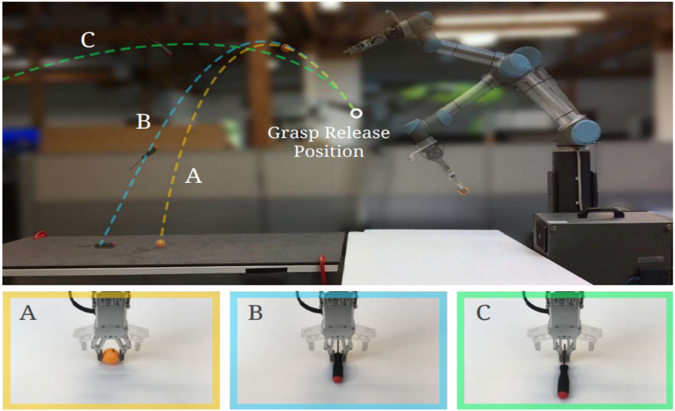
Related Works

- Many previous systems built for throwing, rely on approximating dynamics based on frictional rigid body mechanics.
- Had some assumptions on physical properties.
- Observed limited throwing accuracy(40%).

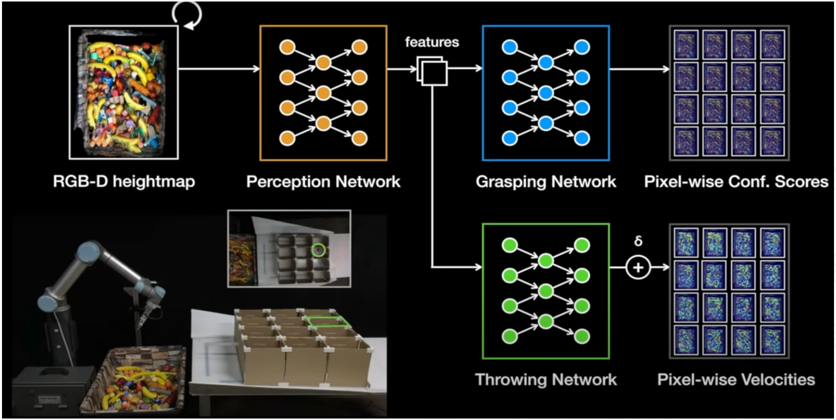


Throwing

- Learning Robotic Throwing
  - Pre-throwing conditions
  - Varying dynamics



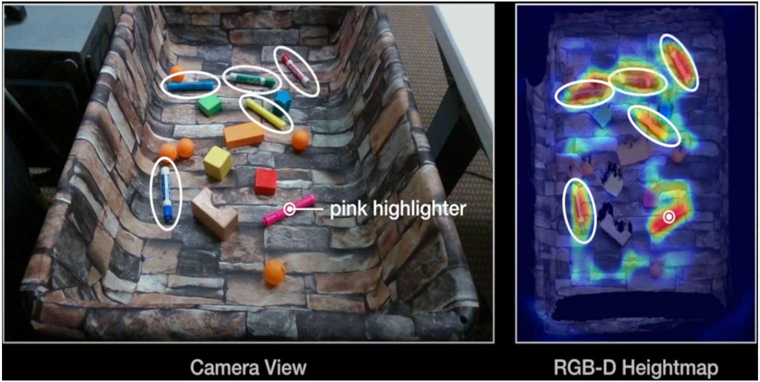
Model



- Throwing network outputs a dense prediction of residual velocity δ.

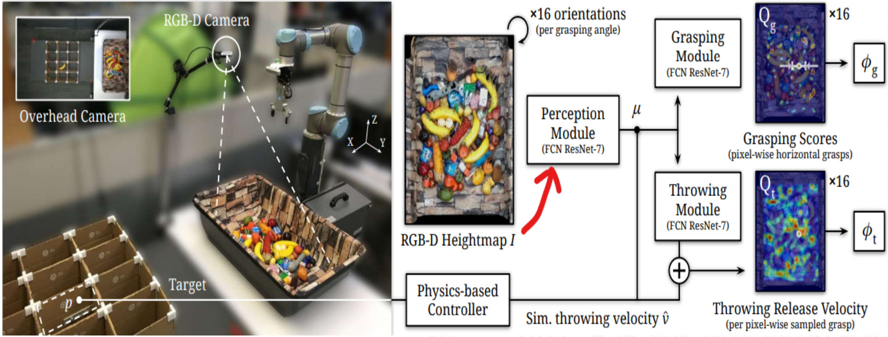
What Does TossingBot Learn ?

- pixel-wise deep features
- relying more on gemoetric cues
- physical properties of objects



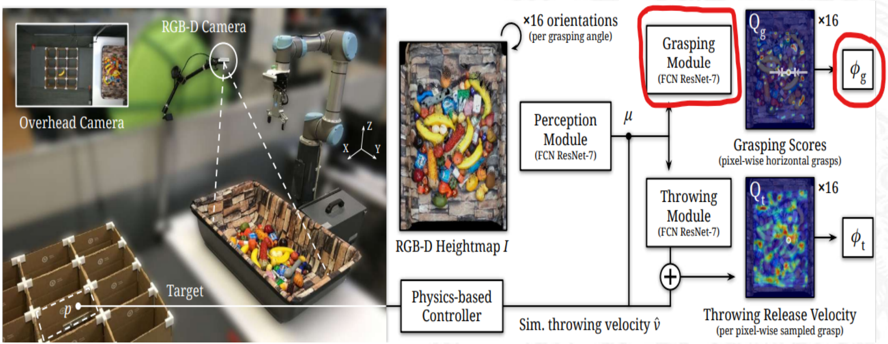
Perception Module: Learning Visual Representations

- Input is an RGB-D heightmap image of workspace
- Get data from fixed camera and project it onto a 3D point cloud.
- The input I is fed into the perception network, which outputs a spatial feature representation μ that is then fed into the grasping and throwing modules.



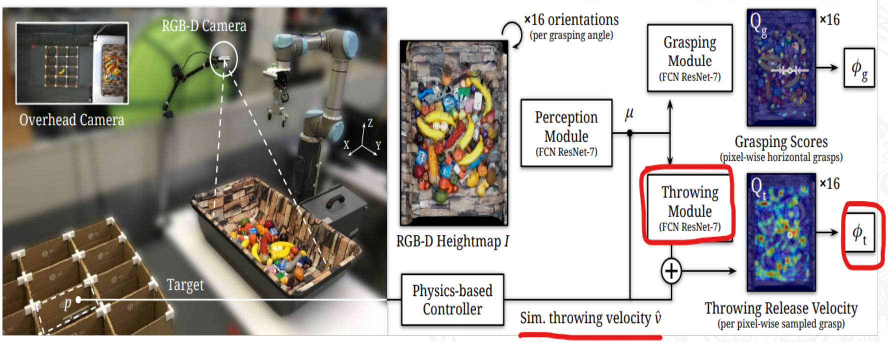
Grasping Module: Learning Parallel-jaw Grasps

- Grasping primitive:
  - takes as input parameters $\phi_g = (x, \theta)$ and executes a top-down parallel-jaw grasp.
- Grasping network:
  - Accepts the visual feature representation μ as input, and outputs a probability map Q_g.
  - Each value of a pixel $qi \in Q_g$ represents the predicted probability of grasping success.



Throwing Module: Learning Throwing Velocities

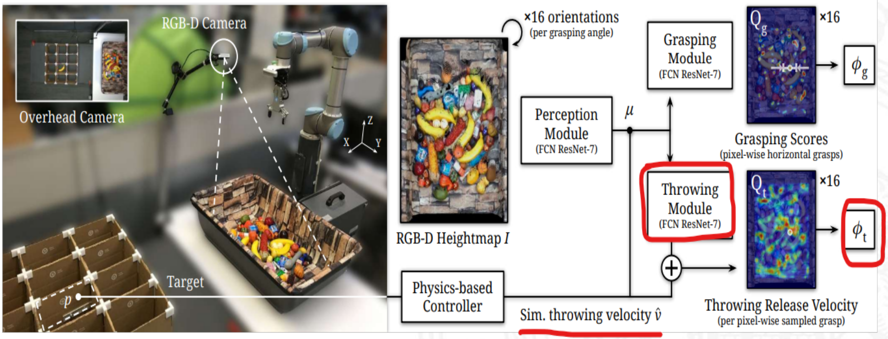
- Accepts the visual feature representation μ as input, and outputs an image Q_t with the same size and resolution as that of the input heightmap I.
- Goal : to predict the release position and velocity of a predefined throwing primitive.
  - Constrain the direction of v to be angled 45 ▫ upwards in the direction of p.



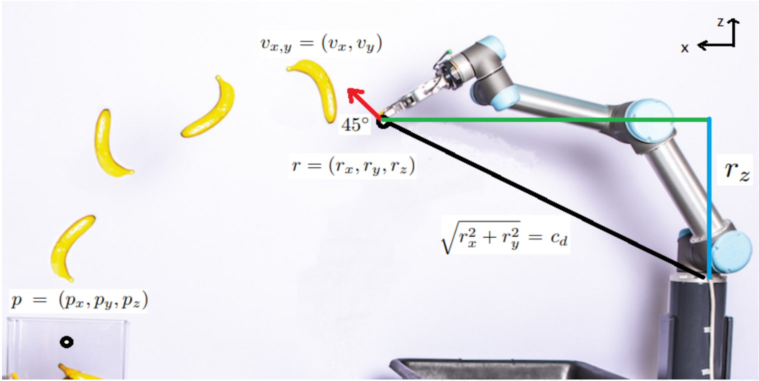
Throwing Module: Learning Throwing Velocities

It is responsible for;

- Planning the release position (r)
  - r is constrained with 2 assumptions.
- Planning the release velocity ($v_{x,y}$)
  - Given a target p and r, there could be multiple solutions of $v_{x,y}$.



Constraints



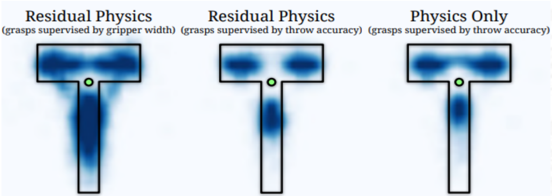
- Fixed throwing height : r_z
- Fixed release distance from robot base : c_d
- Fixed velocity direction angled 45 degree upwards.

Only one unknown variable remained.

Formula

Successful Grasping with Residual Physics

- Green dot indicates theCenter of Mass.
- Darker regions indicate more grasps.
- Leveraging accuracy of throws as supervision enables the grasping policy to learn a more restricted but stable set of grasps.



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

- Physics-based controller
  - It assumes that the effects of aerodynamic drag are negligible.
  - It assumes that the gripper release velocity v directly determines the velocity of the projectile.

- Residual Physics-based controller
  - throwing network that predicts a residual on top of the estimated release velocity ||v^x,y|| for each possible grasp.

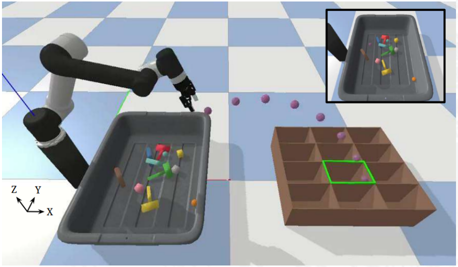
$$\left\|v_{x,y}\right\| = \left\|\hat{v}_{x,y}\right\| + \delta$$

Training

- Trained via self-supervision (based on trial and error)
  - Success after grasping
  - Success after throwing
- Trained jointly with grasping and throwing together.
- Over some time it learns to grasp objects and simultaneously improves its throwing ability.

Evaluation

- Goals of experiment
  - to evaluate the overall accuracy and efficiency of our pick-and-throw system on arbitrary objects.
  - to test its generalization to new objects and target locations unseen during training.
  - to compare our proposed method based on Residual Physics to other baseline alternatives.
- Evaluation metrics
  - grasping success
  - throwing success



Evaluation

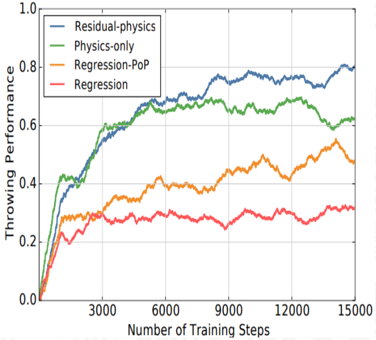
- Simulation environment in PyBullet via trial and error for 15,000 steps.

TABLE I
THROWING PERFORMANCE IN SIMULATION (MEAN %)

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**

TABLE II
GRASPING PERFORMANCE IN 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



Conclusion

- Key is the use of Residual Physics, a hybrid controller that leverages deep learning to predict residuals on top of control parameters estimated with physics.
- In both simulation and real settings show that the system learns to improve grasps for throwing through joint training from trial and error.
- Model Performs significantly better with Residual Physics than comparable alternatives.
- There are constraints.