TossingBot: Learning to Throw Arbitrary Objects with Residual Physics

Nicolas Frick

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

May 28, 2020

Outline

What robots can do

- Pick (Grasp)
- Place
- Move
- Push
- Interact with humans
- ...

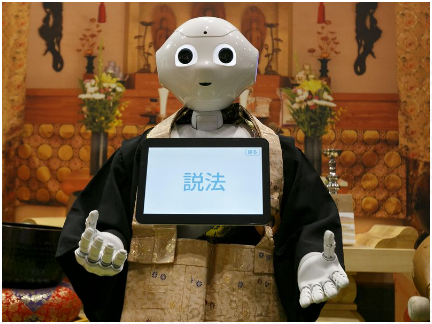


Figure: Robot conducting Buddhist funeral [Mat]

[SK16]

… and toss

- Zheng et al. presented TossingBot in 2019
- End-to-end formalism for grasping and throwing
- Deployed to an UR5 robot

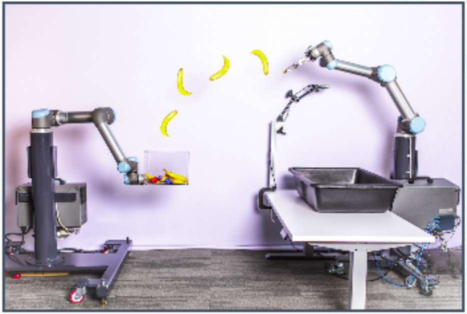


Figure: UR5 throws a banana [Zen19]

Paper:

TossingBot: Learning to Throw Arbitrary Objects with Residual Physics [Zen+19]

- Developed at *Google AI* and the *Princeton University*
- by Andy Zhang
- Other contributors from Columbia University and the MIT
- Best Systems Paper Award, Robotics Science and Systems (2019)

Characteristics:

- (Self-) supervised learning
- Trial and error learning
- Main components:
  - Deep Neural Networks
  - Physics controller
- Key aspects:
  - Joint learning of grasping and throwing policies
  - Residual learning of throw release velocities

Motivation (cont.)

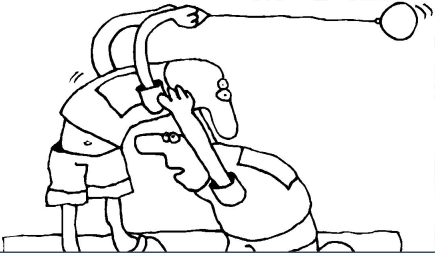
- Not the only tossing approach

Benefits

- **Exploit dynamics** to increase robot's capabilities
- **Extends** the **operation radius**
- **Increase** the **frequency** for pick and place
- Can outperform humans

Challenges

- Acquisition of reliable **pre-throw conditions**
  - e.g grasp of the object
- Handling of **object-centric properties**
  - e.g. mass distribution, friction and shape
- and **dynamics**
  - e.g. aero-dynamics [Gra]



Basics

Self Supervised Learning

- Supervised learning example: Support Vector Machines
- Efforts manual labeling
- (Too) many possibilities for an object to grasp
- Human notions are biased by semantics
- Datasets are restricted in quantity and quality
  $\longrightarrow$ overfitting

[PG16]

Self Supervised Learning (cont.)

- Self-supervised learning tends to limit human involvement
- Task is framed into special form to predict only subset of information
- All information has been provided by the input
- Self-generating its labels

[PG16; Wen19]

Self Supervised Learning (cont.)

- Trial and error training obtains ground truth labels y_i *and* $\bar{\delta}_i$
- At each training step a visual input is fed into the network
- Grasping and throwing parameters are predicted
- Ground truth grasp success label y_i generated either
  - by gripper distance threshold or
  - by throwing success (target hit)

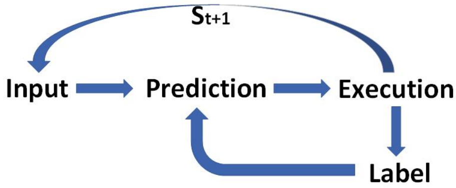


Figure: Training circle

Self Supervised Learning (cont.)

- After throw the landing location is measured
- Landing location p and release velocity v is sampled
- Ground truth residual label $\bar{\delta_i}$ is obtained by $\|v_{x,y}\| - \|\hat{v}_{x,y}\|_{\bar{p}}$
- Training environment is independently reset by the robot

Figure: Reset training environment [Zen19]

Joint Learning of Policies

- DNN[1] maps from visual observations to control parameters:
  - Likelihood of grasping success
  - Throwing release velocities
- Grasping directly supervised by throw accuracy
- Throws directly conditioned on specific grasps
- Stable grasps $\Longleftrightarrow$ predictable throws and throwing velocities

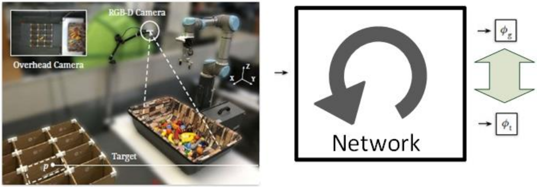


Figure: DNN black box [Zen+19]

[1] Deep Neural Network

Learning of release velocities

- Physics controller predict throw velocities $\hat{v}$
- Based on ideal ballistic motion
- Residual δ is (learned) corrective factor
- Final release velocity: $v = \hat{v} + \delta$

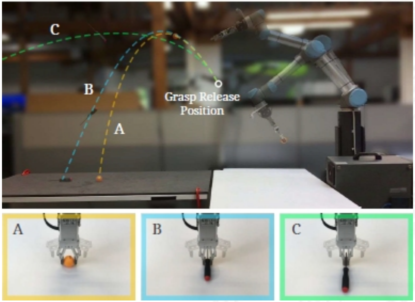


Figure: Different projectile trajectories [Zen+19]

Methods

Network

- Neural Network: $f(I, p)$ [2]
- Output: Prediction of parameters ϕ_g and ϕ_t
- Parameters used by grasping and throwing motion primitives
- Objective: Optimize parameter prediction for a hit

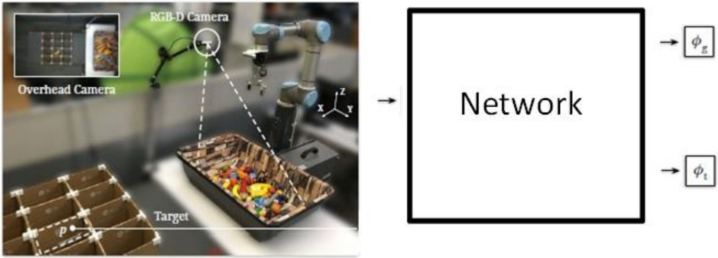


Figure: DNN black box [Zen+19]

[2] I = visual observation, p = landing location

- Input: RGB-D heightmap **I**
- Output: Spatial feature representation μ
- Used by grasping and throwing module

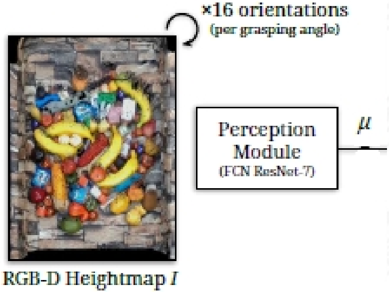


Figure: Input to perception module [Zen+19]

Grasping Module

- Outputs a probability map Q_g (grasping scores)
- Each pixel value represents probability of grasping success
- Input heightmap is rotated 16 x
- Pixel with highest probability determine ϕ_g
- Grasping primitive takes $\phi_g = (x, \theta)$
  where x = pixel location, θ = rotation angle

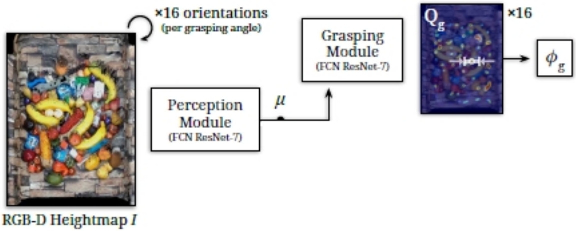


Figure: Grasping module [Zen+19]

Throwing Process

- Predict release position r of throwing primitive
  - Distance $\sqrt{r_x^2 + r_y^2}$ for point of release to base is fixed
- Predict release velocity v of throwing primitive
  - Throw release angle θ constrained to $45°$
  - Only $\|v_{x,y}\|$ is unknown

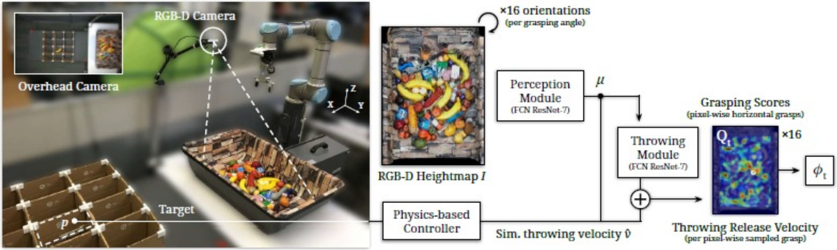


Figure: Throwing process [Zen+19]

Physics-Based Controller

- Physic based controller predicts $\|\hat{v}_{x,y}\|$
- Assume a grasp on the center of mass of the object
- Analytically solves back for $\hat{v}$ given p and r

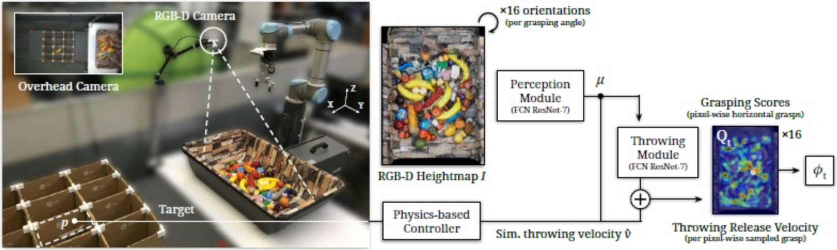


Figure: Throwing process [Zen+19]

Throwing Module

- Output is an image Q_t
- Each pixel holds prediction for residual value δ_i
- δ_i added on top of $\|\hat{v}_{x,y}\|$
- Final release velocity: $\|v_{x,y}\| = \|\hat{v}_{x,y}\| + \delta$
- Throwing primitive takes $\phi_t = (r, v)$

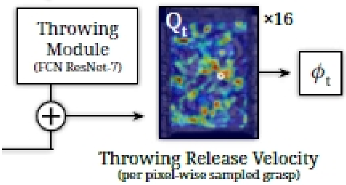


Figure: Throwing module [Zen+19]

Conditioning

- Release velocity $\hat{v}$ is also feed in the grasping and throwing network
- μ concatenated with k-channel image where each pixel holds value of $\hat{v}$
- Conditions the grasping and throwing predictions on $\hat{v}$
- Supervising grasps by accuracy of throws leads to better grasps

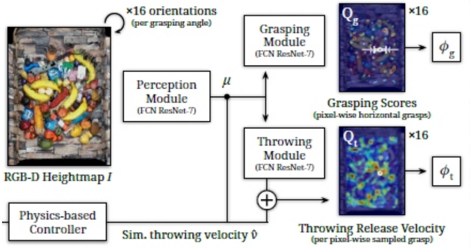


Figure: Release velocity feed [Zen+19]

Full Network

- Self-supervision from trial and error
- Tracking ground truth landing position of thrown objects
- Not a single network that maps states to actions
- Four modules that provide intermediate (differentiable) results
- Output are factors that does not directly control the actuator
- Complex systems are hard to control

[Gla17]

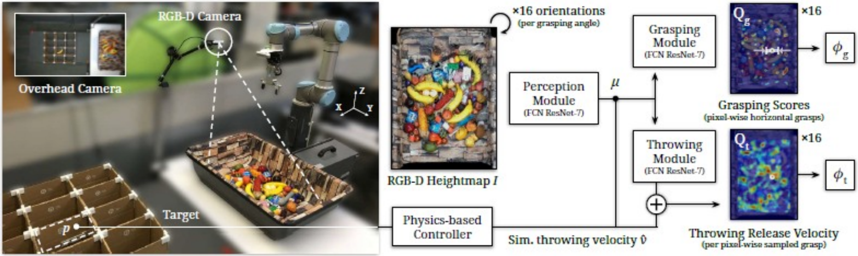


Figure: Overview [Zen+19]

Results

Experimental Setup

- Evaluation metrics
  - Grasping success (% rate of succesfull grasps)
  - Throwing success (% rate of target hits)
- 12 various objects are grasped and thrown
- Target are 12 boxes outside kinematic range
- Real world: UR5 robot with RG2 gripper

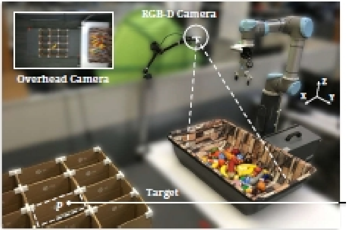


Figure: Workspace [Zen+19]

Figure: RG2 gripper [Onr]

Simulation

- 8 different objects (4 seen, 4 unseen), 12 in total
- Varying center of mass (CoM)
- Simulated environment does not account for aerodynamics
- Real world experiments are conducted

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

Figure: [Zen+19]

Real World

- 15,000 steps training, 1,000 steps testing
- Average grasping and throwing success rates

TABLE III
GRASPING AND THROWING PERFORMANCE IN REAL (MEAN %)

	Grasping		Throwing	
Method	Seen	Unseen	Seen	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**

TABLE IV
PICKING SPEED VS STATE-OF-THE-ART SYSTEMS

System	Mean Picks Per Hour (MPPH)
Cartman [24]	120
Dex-Net 2.0 [20]	250
FC-GQ-CNN [27]	296
Dex-Net 4.0 [21]	312
TossingBot (w/ Placing)	432
TossingBot (w/ Throwing)	**514**

Figure: [Zen+19]

Performance

- Residual physics outperforms by learning residual throwing velocities
- Compensate for for grasping offsets from objects CoM

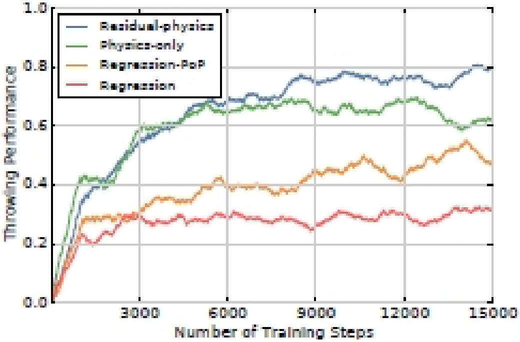


Figure: Throwing performance on hammers[Zen+19]

- 2 variants of grasp success label y_i
- Grasping supervised by throwing yields best throwing results
- Supervised grasps are more restricted, resulting in more dexterous throws

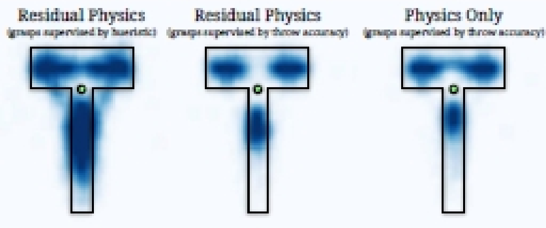


Figure: Histograms of succesfull grasps[Zen+19]

Conclusion

Conclusion

- Paper provides new perspectives on throwing
  - Relationship of throwing to grasping
  - Throwing correlates with grasp quality
  - Learning by combining physics with trial and error
  - Synergies between grasping and throwing is exploited
- *Residual Physics* leverage advantages of physic based controllers while maintaining the capacity to account for dynamics
- Generalization via analytic models
- Data-driven residual corrects the real world projectile velocity

Thank you for your attention!

Questions?

[Gla17] Tobias Glasmachers. "Limits of end-to-end learning". In: *Journal of Machine Learning Research* 77 (2017), pp. 17–32. ISSN: 15337928. arXiv: arXiv:1704.08305v1.

[Gra] Gratis Malvorlagen.de. *Gefaehrlicher Hammerwerfer Ausmalbild und Malvorlage (Sport)*. URL: https://www.gratis-malvorlagen.de/sport/gefaehrlicher-hammerwerfer/ (Retrieved 05/23/2020).

[Mat] Emily Matchar. *Nine Tasks Robots Can Do That May Surprise You | Innovation | Smithsonian Magazine*. URL: https://www.smithsonianmag.com/innovation/nine-tasks-robots-can-do-that-may-surprise-you-180964729/ (Retrieved 05/23/2020).

References (cont.)

[NAS] NASA Glenn Research Center. *Ballistic Flight Equations*. URL: https://www.grc.nasa.gov/www/k-12/airplane/ballflght.html (Retrieved 05/23/2020).

[Onr] Onrobot. *RG2 robot gripper - A flexible end-of-arm tooling gripper | OnRobot*. URL: https://onrobot.com/en/products/rg2-gripper (Retrieved 05/27/2020).

[PG16] Lerrel Pinto and Abhinav Gupta. "Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours". In: *Proceedings - IEEE International Conference on Robotics and Automation* 2016-June (2016), pp. 3406–3413. ISSN: 10504729. DOI: 10.1109/ICRA.2016.7487517. arXiv: 1509.06825.

[SK16] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. 2016, pp. 1–2227. DOI: 10.1007/978-3-319-32552-1.

[Uni16] Universal Robots A/S. *UR5 Technical specifications*. 2016. URL: https://www.universal-robots.com/download-center/#/cb-series/ur5 (Retrieved 05/25/2020).

[Uni18] Universal Robots A/S. *Parameters for calculations of kinematics and dynamics*. 2018. URL: https://www.universal-robots.com/articles/ur-articles/parameters-for-calculations-of-kinematics-and-dynamics/ (Retrieved 05/25/2020).

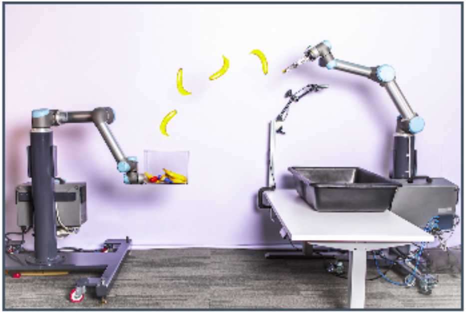
References (cont.)

[Wen19] Lilian Weng. *Self-Supervised Representation Learning*. 2019. URL: https://lilianweng.github.io/lil-log/2019/11/10/self-supervised-learning.html (Retrieved 05/23/2020).

[Zen+19] Andy Zeng et al. "TossingBot: Learning to Throw Arbitrary Objects with Residual Physics". In: (2019). DOI: 10.15607/rss.2019.xv.004. arXiv: 1903.11239.

[Zen19] Andy Zeng. *TossingBot: Learning to Throw Arbitrary Objects with Residual Physics*. 2019. URL: https://tossingbot.cs.princeton.edu/ (Retrieved 05/23/2020).

Video

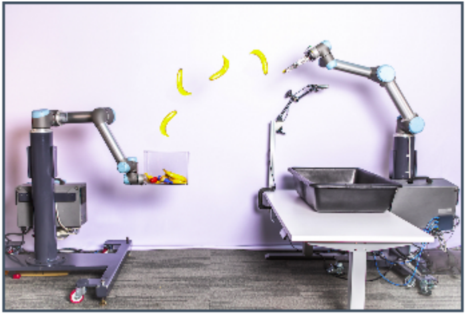


Ballistics [Zen19]

Appendix - Future Work

- Train networks to predict motions that account for fragile objects
- Explore additional sensing modalities such as force-torque
- How should robots learn semantics of the visual world?
- Classic computer vision: predefined semantics using manually constructed class categories
- Here: Implicitly learn object-level semantics from physical interactions

Video



[Zen19]

Appendix - Loss Function

- Loss function for network training:
  - Binary cross-entropy error $\mathcal{L}_g$ from predictions of grasping success
  - Huber-loss $\mathcal{L}_t$ from its regression of δ_i for throwing

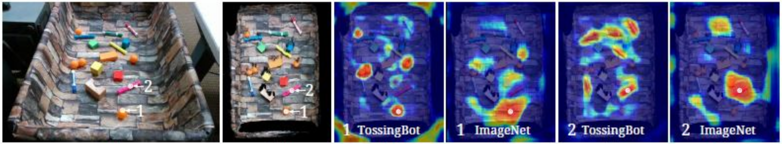
$$(1) \quad \mathcal{L} = \mathcal{L}_g + y_i \mathcal{L}_t$$

$$(2) \quad \mathcal{L}_g = -(y_i \log q_i + (1 - y_i) \log(1 - q_i))$$

$$(3) \quad \mathcal{L}_t = \begin{cases} \frac{1}{2}(\delta_i - \bar{\delta}_i)^2, \ for \ |\delta_i - \bar{\delta}_i| < 1, \\ |\delta_i - \bar{\delta}_i| - \frac{1}{2}, otherwise \end{cases}$$

where y_i is the binary ground truth grasp success label, q_i and $\bar{\delta}_i$ are the predicted values and $\bar{\delta}_i$ is the ground truth residual label

Appendix (cont.)



(a) (b) (c) (d) (e) (f)

Fig. 10. **Emerging semantics from interaction.** Visualizing pixel-wise deep features μ learned by TossingBot (c,e) overlaid on the input heightmap image (b) generated from an RGB-D side-view (a) of a bin of objects. (c) shows a heatmap of pixel-wise feature distances (hotter = smaller distance) from the feature vector of a query pixel on a ping pong ball (labeled 1). Likewise, (e) shows a heatmap of pixel-wise feature distances from the feature vector of a query pixel on a pink marker pen (labeled 2). These visualizations show that TossingBot learns features that distinguish object categories from each other without explicit supervision (*i.e.*, only task-level grasping and throwing). For reference, the same visualization technique is used on deep features generated by a ResNet-18 pre-trained on ImageNet (d,f).

Figure: [Zen+19]

Appendix (cont.) - Image Projection

- Capture RGB-D image from fixed mount camera
- Project data onto 3D-point cloud
- Orthographical back-projection in gravity direction
- color and height-from-bottom channels
- normalization allows sharing of learned convolutional filters

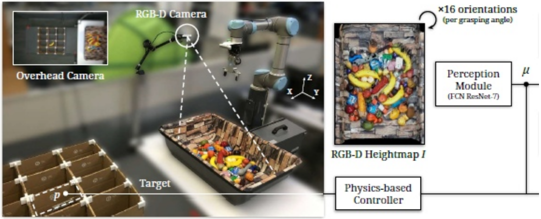


Figure: Projection [Zen+19]

Appendix (cont.) - Gripper Modalities

- Top-down parallel yaw grasp centered at $x = (x_x, x_y, x_z)$
- Oriented θ° around gravity direction
- Gripper approaches x until middle point of finger tips meets x
- Gripper closes and lifts upwards
- Planning by stable, collision-free IK-solver

Appendix (cont.) - Release Position

- Derive release position r from landing location p
- Assume: aerial trajectory is linear on xy-horizontal plane and in the v_x, v_y direction
- Neglect orthogonal aerodynamic forces
- Parallel aerodynamic forces are compensated
- Making all release positions accessible by robot
- Constants: $r_z = 0.04$ m and distance to base $\sqrt{r_x^2 + r_y^2} = 0.7$ m in sim, 0.02m and 0.76 m in reality
- Constraint: $(r_{x,y} - p_{x,y}) x v_{x,y} = 0$

Appendix (cont.) - Physics controller

- Physics based controller provides a closed form solution
- Generalizes well to new landing locations
- Serves as consistent approximation for $\hat{v}$
- Simplified model
- Neglects aerodynamic drag
- Gripper release velocity does not directly determine projectile velocity
- Centripetal forces

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

Appendix (cont.) Example Calculation

- UR5 joint speed: $\frac{180^\circ}{s} \Rightarrow \omega = \frac{\pi}{s}$
- Length of lower arm: $\approx 0.49m$
- Peripheral speed: $v = \frac{pi}{s} \cdot 0.49m \approx 1.54\frac{m}{s}$
- Ballistic equation projectile range: $x = v_0 \cdot \cos\theta \cdot t$
- Ballistic equation ToF[3]: $t = \frac{2 \cdot v_0 \cdot \sin\theta}{g}$
- Throwing angle $\theta =$ impact angle $= 45^\circ$

$$x = \frac{2 \cdot (v_0)^2 \cdot \sin(2 \cdot \theta)}{2 \cdot g} = \frac{2 \cdot (1.54\frac{m}{s})^2 \cdot \sin 90}{2 \cdot 9.81\frac{m}{s^2}} \approx 0.24m$$

[3] Time-of-Flight

Appendix (cont.) - Learning of release velocities

Ballistic calculation:

- Ballistic equation projectile range: $x = \hat{v} \cdot \cos\theta \cdot t$
- Ballistic equation ToF[4]: $t = \frac{2 \cdot v_0 \cdot \sin\theta}{g}$
- Throwing angle θ = impact angle

$$\hat{v} = \sqrt{\frac{x \cdot g}{\sin(2 \cdot \theta)}} \tag{2}$$

e.g. $\sqrt{\frac{0.24m \cdot 9.81\frac{m}{s^2}}{\sin(2 \cdot 45)}} \approx 1.53\frac{m}{s}$ throwing velocity

[NAS]

[4] Time-of-Flight

Appendix (cont.) - Learning of release velocities

- fixed throwing release height r_z
- fixed release distance from robot base origin c_d
- release vel. angled 45° upwards
- landing location p $= (p_x, p_y, p_z)$
- release position r is fixed at $c_d = 0.76$m and r_z at constant height $c_h = 0.02$m
- Release vel. magnitude $\|v\|$

Ballistic calculation:

$$\theta = \arctan(\frac{p_y}{p_x})$$

$$r_x = c_d \sin\theta$$

$$r_y = c_d \cos\theta$$

$$\|v\| = \sqrt{\frac{a(p_x^2 + p_y^2)}{r_z - p_z - \sqrt{p_x^2 + p_y^2}}}$$

Appendix (cont.) - Self Supervised Learning

- Pinto and Gupta [PG16] emphasized benefits of large-scale datasets
- Introduced large robot dataset
- Limit human involvement
- Execute trial and error grasps
- Image patch of grasp feed to CNN
- Output is the likelihood of the grasp

[PG16]

Appendix (cont.) - Self Supervised Learning

- Trained model is used for next grasping stage
- Execute grasp along the predicted output
- Grasps are evaluated by gripper's force sensor
- Correct grasp modalities are reinforced

[PG16]

Appendix (cont.) - Self Supervised Learning

- better utilizing unlabelled data, while learning in a supervised learning manner
- framing a supervised learning task in a special form to predict only a subset of information using the rest
- all the information needed, both inputs and labels, has been provided. This is known as self-supervised learning.
- self-generated labels
- To make use of this much larger amount of unlabeled data, one way is to set the learning objectives properly so as to get supervision from the data itself.
- The self-supervised task, also known as pretext task, guides us to a supervised loss function.

[Wen19]

Appendix (cont.)

Algorithm 1 System Pipeline

```
Initialize robot.
Initialize policy with model f.
Initialize replay buffer.
while step i < N and not terminate do
    I^i = robot.CaptureState()
    p^i = robot.SelectTarget()
    φ_g^i, φ_t^i = f.Inference(I^i, p^i)
    while robot.is_grasping do
        f.ExperienceReplay(buffer)
    y^{i-1} = robot.CheckGraspSuccess()
    robot.ExecuteThrow(φ_t^{i-1}, p^{i-1})        ▷ asynchronous
    while robot.is_throwing do
        f.ExperienceReplay(buffer)
    robot.ExecuteGrasp(φ_g^i)                     ▷ asynchronous
    p̄^{i-1} = robot.TrackLanding()
    buffer.SaveData(I^{i-1}, p^{i-1}, φ_g^{i-1}, φ_t^{i-1}, y^{i-1}, p̄^{i-1})
    i = i + 1
```

Figure: Ideal ballistic equations [Zen+19]

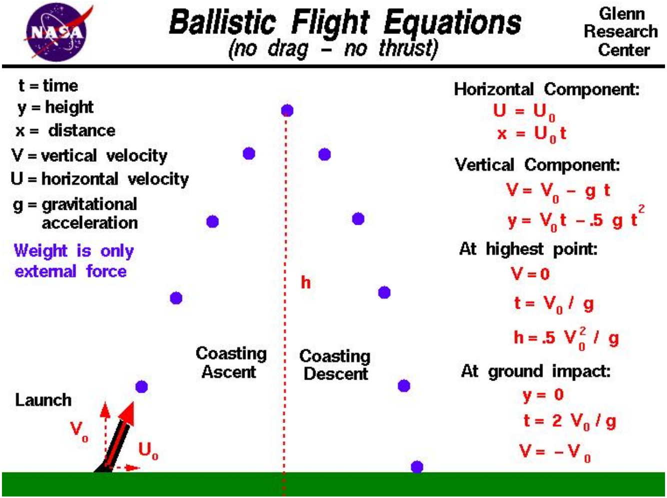


Figure: Ideal ballistic equations [NAS]

Appendix (cont.)

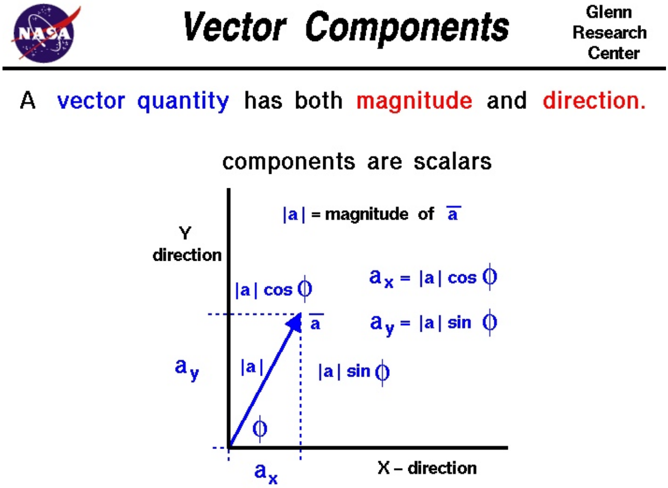


Figure: Vector components [NAS]

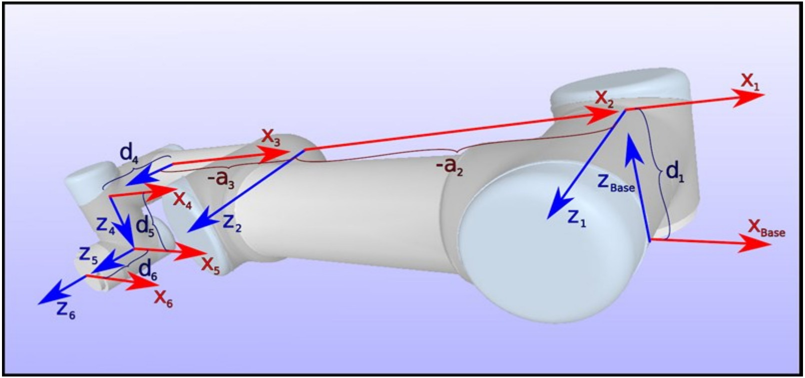


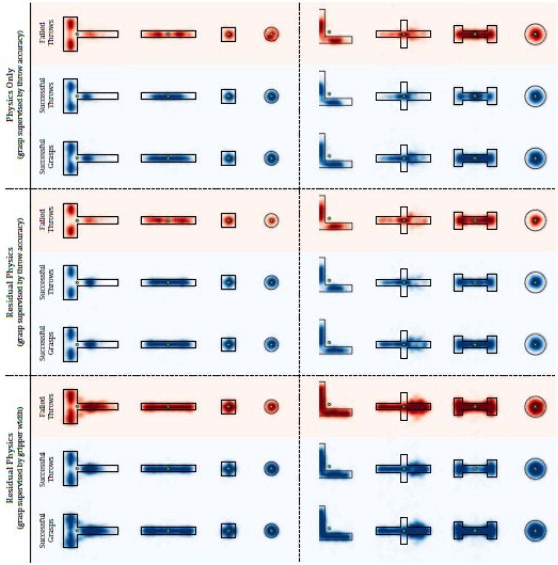
Figure: Denavit–Hartenberg parameters of UR robots [Uni18]

Appendix (cont.)

UR5e							
Kinematics	theta [rad]	a [m]	d [m]	alpha [rad]	**Dynamics**	Mass [kg]	Center of Mass [m]
Joint 1	0	0	0.1625	π/2	Link 1	3.761	[0, -0.02561, 0.00193]
Joint 2	0	-0.425	0	0	Link 2	8.058	[0.2125, 0, 0.11336]
Joint 3	0	-0.3922	0	0	Link 3	2.846	[0.15, 0.0, 0.0265]
Joint 4	0	0	0.1333	π/2	Link 4	1.37	[0, -0.0018, 0.01634]
Joint 5	0	0	0.0997	-π/2	Link 5	1.3	[0, 0.0018,0.01634]
Joint 6	0	0	0.0996	0	Link 6	0.365	[0, 0, -0.001159]

Figure: UR5 Technical specifications [Uni16]

Appendix (cont.)



Appendix (cont.)

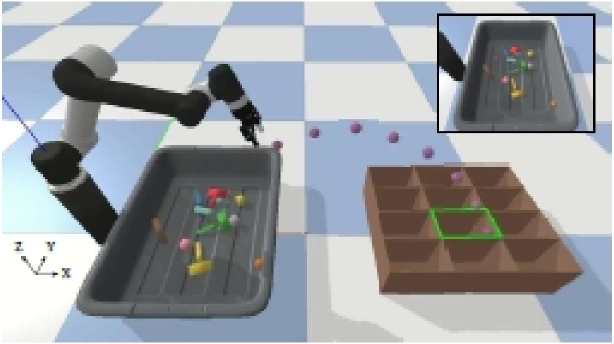
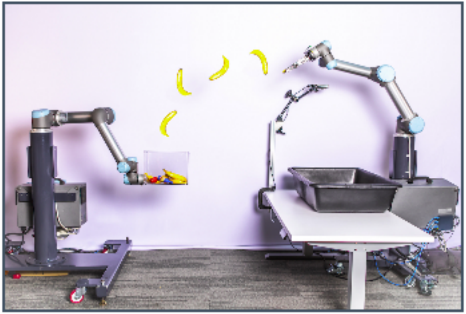


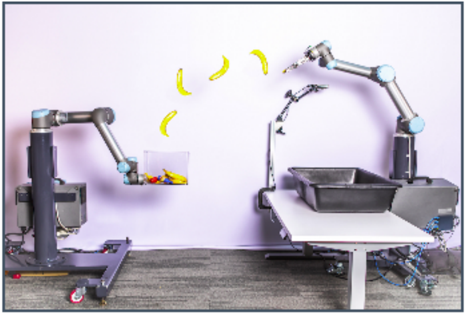
Figure: PyBullet simulation [Zen+19]

Video



[Zen19]

Video



[Zen19]