Robotics: Introduction to AI in robotics

Vladimír Petrík

vladimir.petrik@cvut.cz

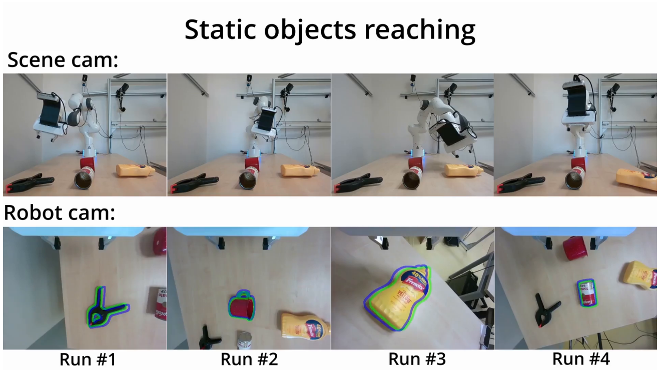
05.01.2026

Motivation

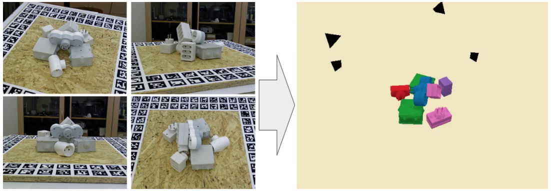
- You know how to control robot to reach the target pose (SE3)
- Where to get the pose for the given task?

Motivation

- You know how to control robot to reach the target pose (SE3)
- Where to get the pose for the given task? **Vision**



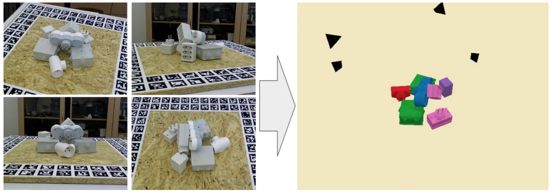
6D pose estimation



$$T_{CO}, M = f_{\text{estimate}}(I, K, \mathcal{D})$$

- I image
- K camera matrix
- $\mathcal{D}$ database of meshes
- $M \in \mathcal{D}$ mesh of the object

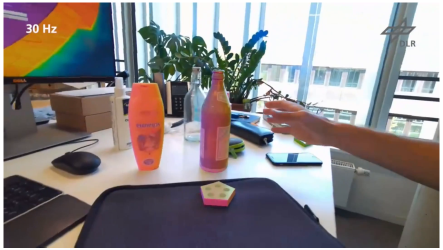
6D pose estimation



$$T_{CO}, M = f_{\text{estimate}}(I, K, \mathcal{D})$$

- I image
- K camera matrix
- $\mathcal{D}$ database of meshes
- $M \in \mathcal{D}$ mesh of the object

6D pose tracking



$$T^{i+1}_{CO} = f_{\text{track}}(I, K, M, T^{i}_{CO})$$

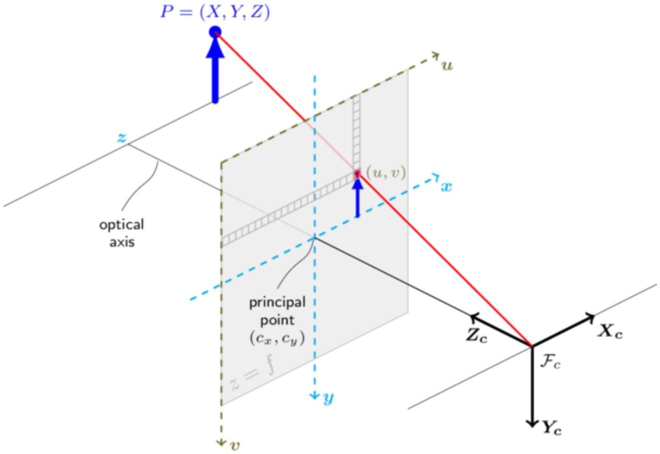
- I image
- K camera matrix
- M mesh

Why is 6D pose estimation difficult?

[1]https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html

Why is 6D pose estimation difficult?

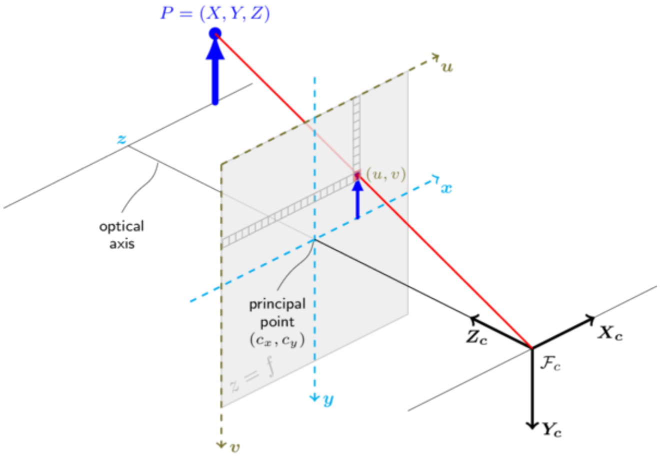
- Projection, pinhole camera model[1]



[1] https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html

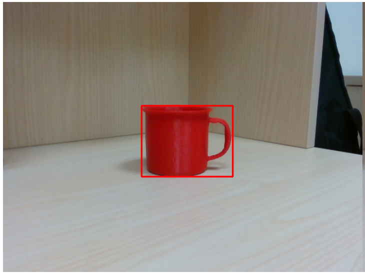
Why is 6D pose estimation difficult?

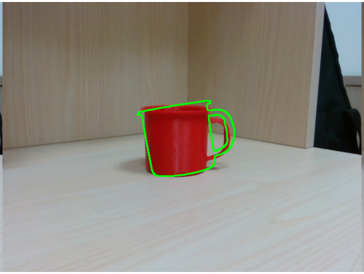
- Projection, pinhole camera model[1]
- $\lambda \begin{pmatrix} u & v & 1 \end{pmatrix}^\top = K\boldsymbol{x}_c$
  - u, v - pixel coordinates
  - $\boldsymbol{x}_c$ - 3D point in camera frame
  - K - camera matrix
  - $K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$
- With projection we are loosing information about depth

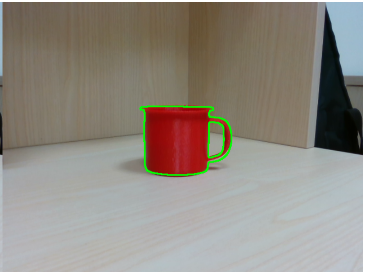


[1]https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html

6D pose estimation pipeline


Object detection in image


Coarse pose estimation


Pose refinement

Object detection

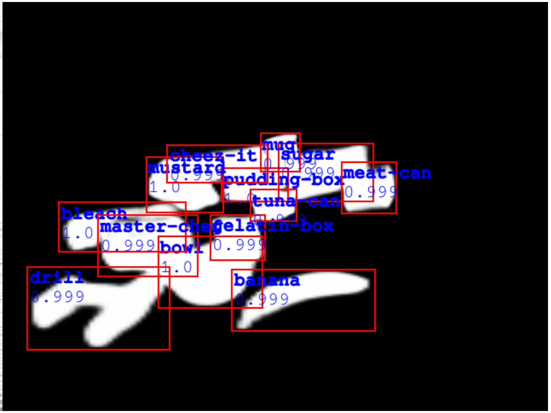
Object detection

- Goal: detect object in image
  - mask
  - bounding box
  - object instance id
  - confidence of prediction

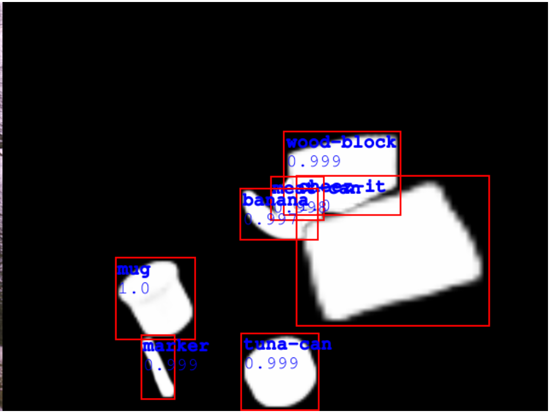
Object detection

- Goal: detect object in image
  - mask
  - bounding box
  - object instance id
  - confidence of prediction
- Neural network - Mask R-CNN
  - needs **good** training data
  - annotated images
  - synthetic images

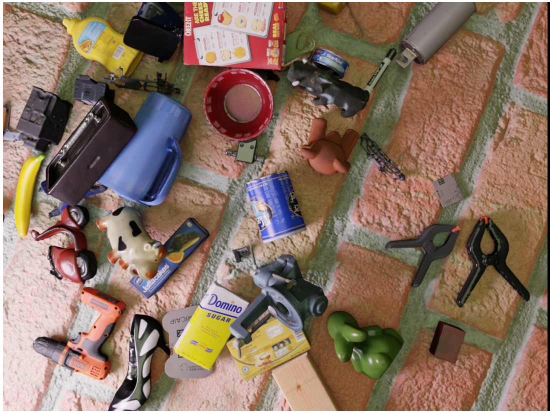
Trained Mask R-CNN results

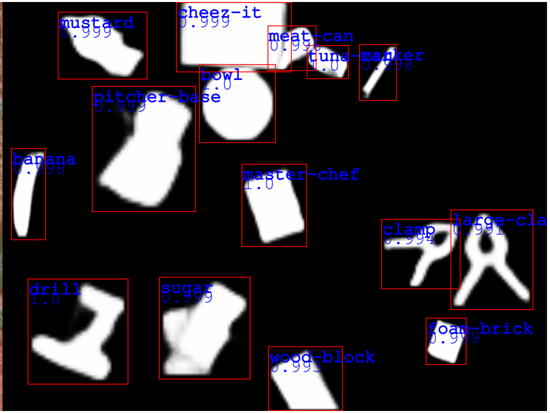


Trained Mask R-CNN results

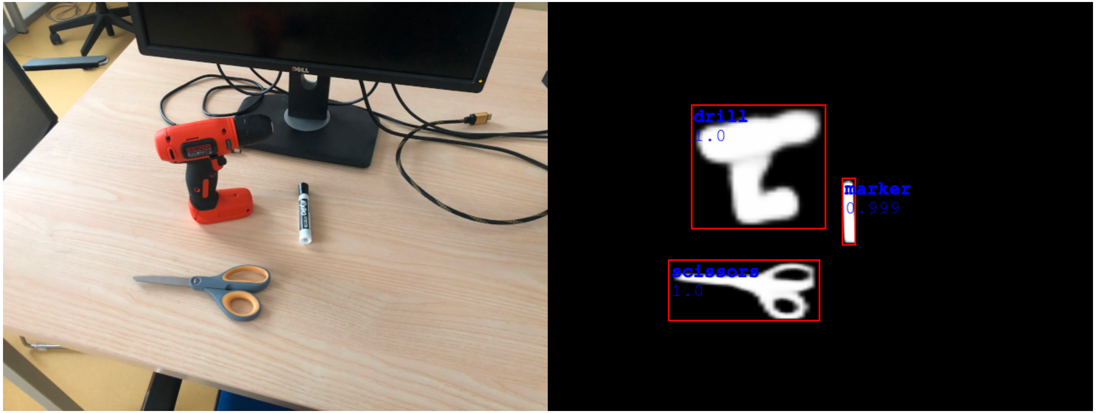


Trained Mask R-CNN results

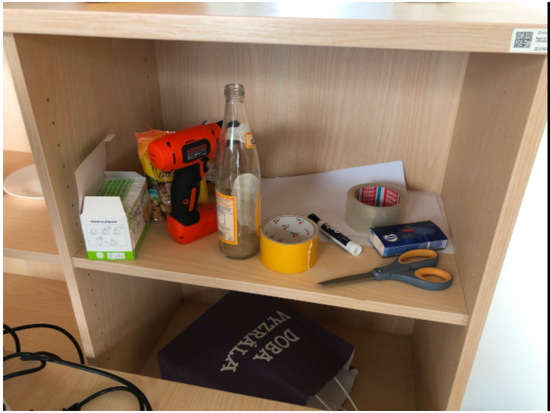


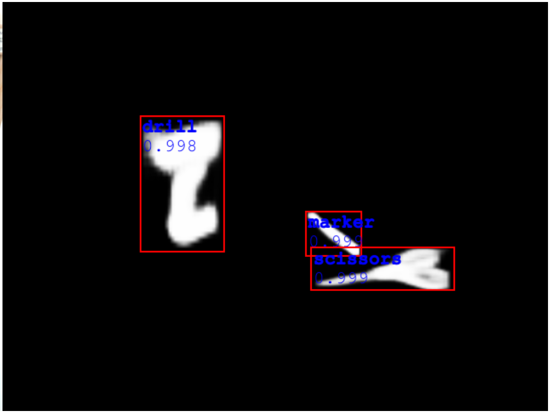


Trained Mask R-CNN results



Trained Mask R-CNN results

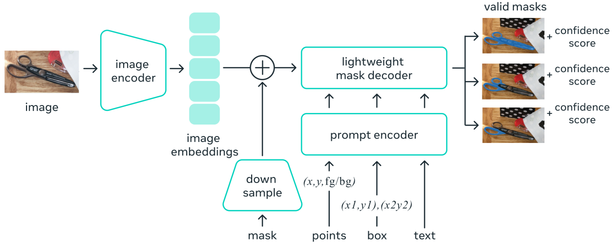




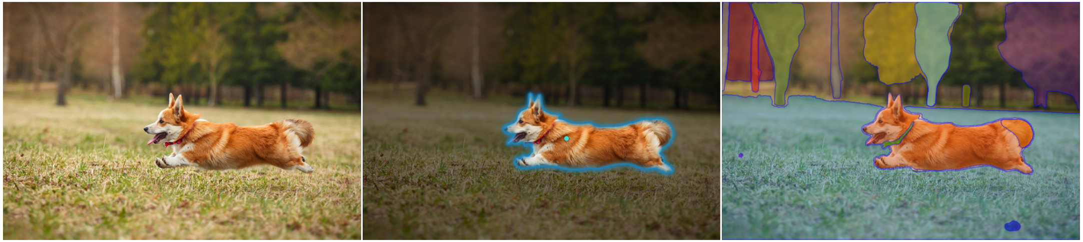
Object detection without retraining

- Segment Anything Model (SAM)
  - segment any object, in any image, with a single click
  - dataset of 10M images, 1B masks

Universal segmentation model

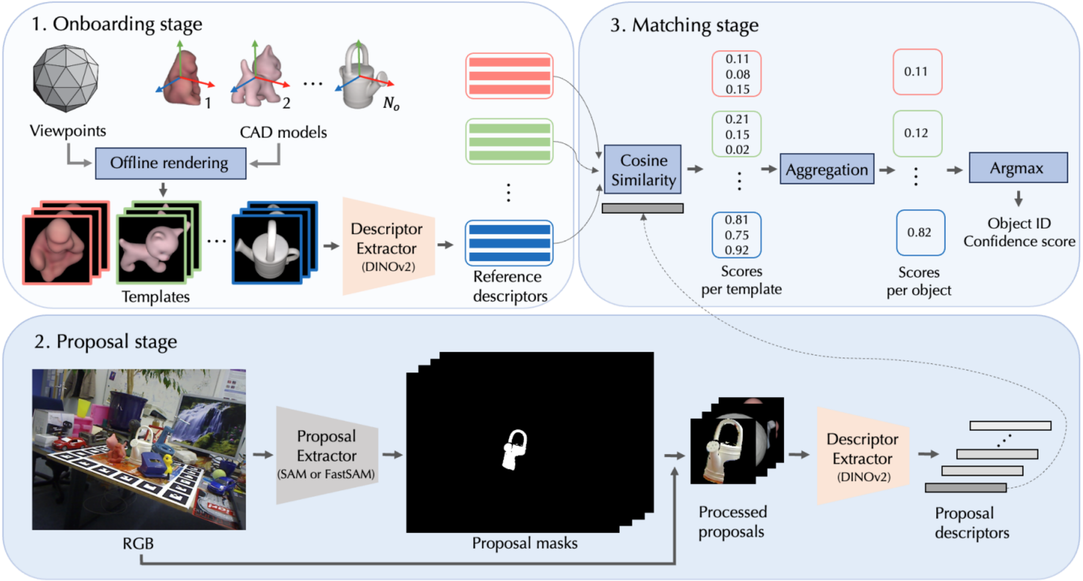


SAM results



SAM results

Mesh model from segmentation mask - CNOS

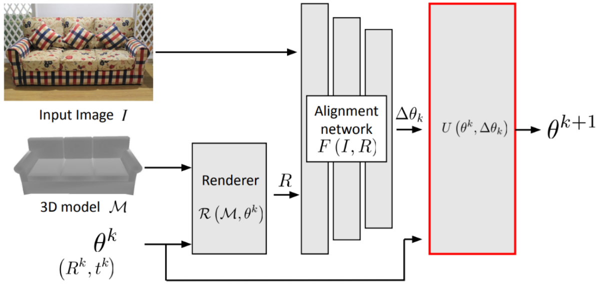


CosyPose

Consistent multi-view multi-object 6D pose estimation

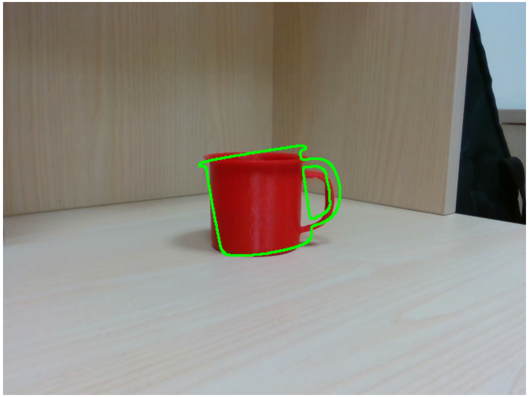
Coarse pose estimation

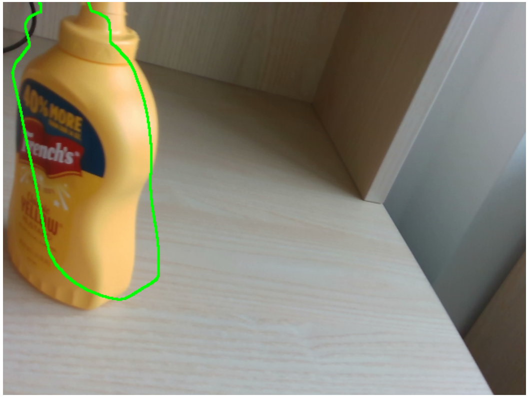
- Input: image crop and mesh model[2]
- Goal: estimate 6D pose
- Approach:
  - render and compare strategy
  - neural network
  - initial position is estimated from camera matrix
  - initial orientation is identity
- Training
  - synthetic and real data
  - 10 hours on 32 GPUs



[2]Image based on: https://arxiv.org/pdf/2204.05145.pdf

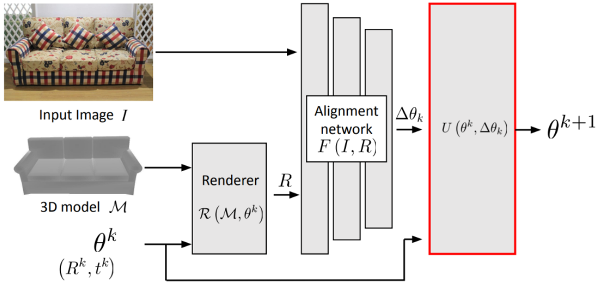
Coarse pose estimation results



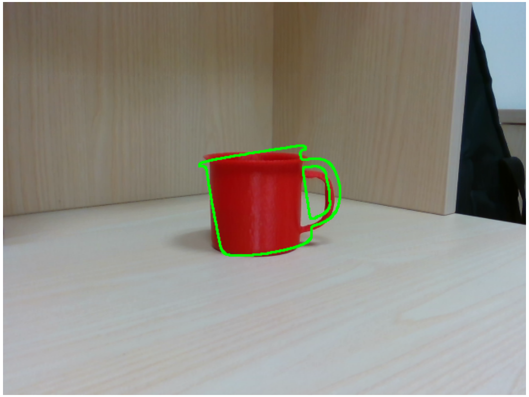


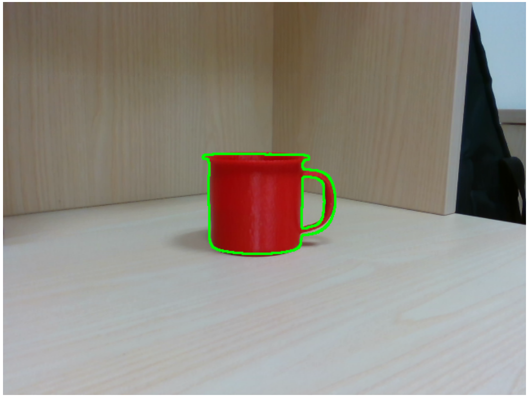
Refiner

- The same render-and-compare strategy
- Network learns to predict small corrections
- Evaluated iteratively
- Another 10 hours on 32 GPUs

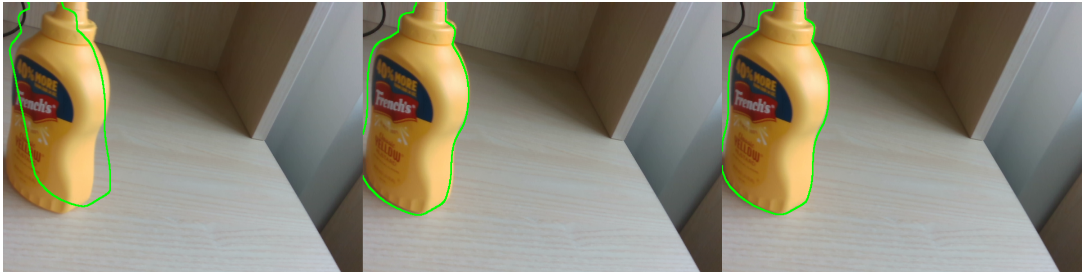


Refiner results





Refiner results



BOP challenge

- BOP: Benchmark for 6D Object Pose Estimation
- Main benchmark/competition for 6D pose estimation

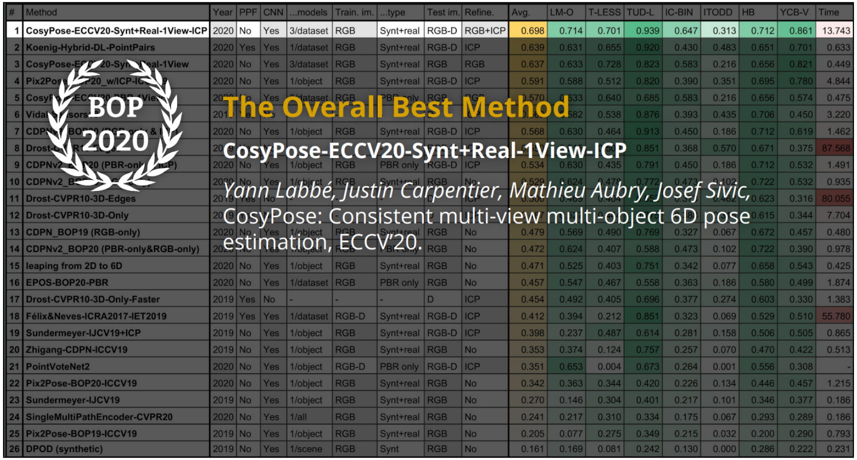
BOP challenge

- BOP: Benchmark for 6D Object Pose Estimation
- Main benchmark/competition for 6D pose estimation
- Tasks on seen objects
    - Model-based 2D detection/segmentation of seen objects [new in 2022]
    - Model-based 6D localization of seen objects

BOP challenge

- BOP: Benchmark for 6D Object Pose Estimation
- Main benchmark/competition for 6D pose estimation
- Tasks on seen objects
  - Model-based 2D detection/segmentation of seen objects [new in 2022]
  - Model-based 6D localization of seen objects
- Tasks on unseen objects [new in 2023]
  - Model-based 2D detection/segmentation of unseen objects
  - Model-based 6D localization of unseen objects

CosyPose at BOP challenge



BOP 2020

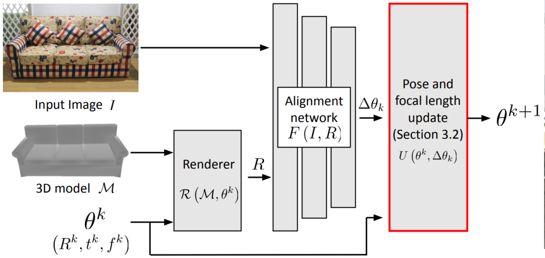
The Overall Best Method

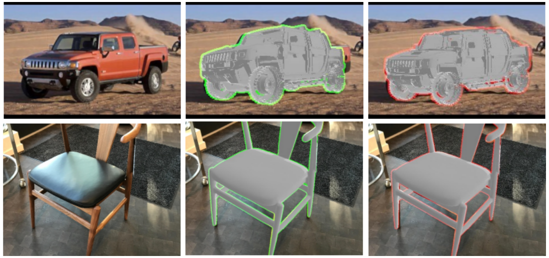
CosyPose-ECCV20-Synt+Real-1View-ICP

Yann Labbé, Justin Carpentier, Mathieu Aubry, Josef Sivic,
CosyPose: Consistent multi-view multi-object 6D pose estimation, ECCV'20.

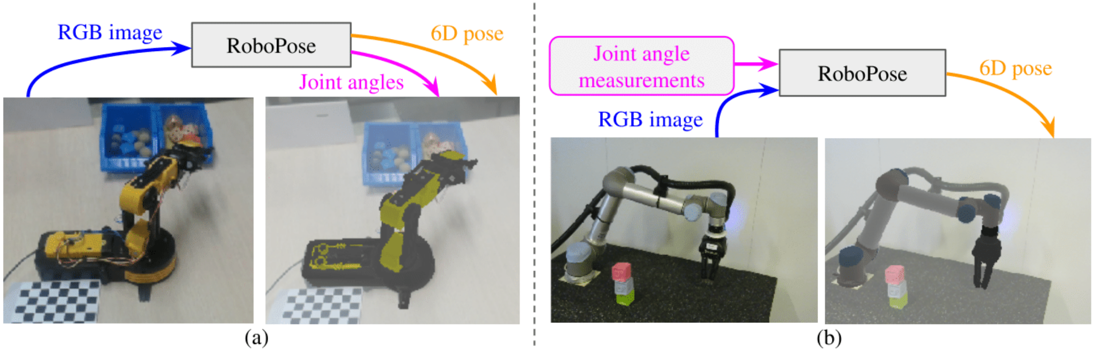
#	Method	Year	PPF	CNN	...models	Train. im.	...type	Test im.	Refine.	Avg.	LM-O	T-LESS	TUD-L	IC-BIN	ITODD	HB	YCB-V	Time
1	CosyPose-ECCV20-Synt+Real-1View-ICP	2020	No	Yes	3/dataset	RGB	Synt+real	RGB-D	RGB+ICP	0.698	0.714	0.701	0.939	0.647	0.313	0.712	0.861	13.743
2	Koenig-Hybrid-DL-PointPairs	2020	Yes	Yes	1/dataset	RGB	Synt+real	RGB-D	ICP	0.639	0.631	0.655	0.920	0.430	0.483	0.651	0.701	0.633
3	CosyPose-ECCV20-Synt+Real-1View	2020	No	Yes	3/dataset	RGB	Synt+real	RGB	RGB	0.637	0.633	0.728	0.823	0.583	0.216	0.656	0.821	0.449
4	Pix2Pose-BOP20_w/ICP-ICCV19	2020	No	Yes	1/object	RGB	Synt+real	RGB-D	ICP	0.591	0.588	0.512	0.820	0.390	0.351	0.695	0.780	4.844
5	CosyPose-ECCV20-PBR-1View	2020	No	Yes	3/dataset	RGB	PBR only	RGB	RGB	0.570	0.633	0.640	0.685	0.583	0.216	0.656	0.574	0.475
6	Vidal-Sensors18	2019	Yes	No	-	-	-	D	ICP	[illegible]	0.582	0.538	0.876	0.393	0.435	0.706	0.450	3.220
7	CDPNv2_BOP20 (RGB-only & ICP)	2020	No	Yes	1/object	RGB	Synt+real	RGB-D	ICP	0.568	0.630	0.464	0.913	0.450	0.186	0.712	0.619	1.462
8	Drost-CVPR10-Edges	2019	Yes	No	-	-	-	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.851	0.368	0.570	0.671	0.375	87.568
9	CDPNv2_BOP20 (PBR-only & ICP)	2020	No	Yes	1/object	RGB	PBR only	RGB-D	ICP	0.534	0.630	0.435	0.791	0.450	0.186	0.712	0.532	1.491
10	CDPNv2_BOP20 (RGB-only)	2020	No	Yes	1/object	RGB	Synt+real	RGB	No	0.529	0.624	0.478	0.772	0.473	0.102	0.722	0.532	0.935
11	Drost-CVPR10-3D-Edges	2019	Yes	No	-	-	-	D	ICP	[illegible]	[illegible]	0.404	[illegible]	[illegible]	0.462	0.623	0.316	80.055
12	Drost-CVPR10-3D-Only	2019	Yes	No	-	-	-	D	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.615	0.344	7.704
13	CDPN_BOP19 (RGB-only)	2020	No	Yes	1/object	RGB	Synt+real	RGB	No	0.479	0.569	0.490	0.769	0.327	0.067	0.672	0.457	0.480
14	CDPNv2_BOP20 (PBR-only&RGB-only)	2020	No	Yes	1/object	RGB	PBR only	RGB	No	0.472	0.624	0.407	0.588	0.473	0.102	0.722	0.390	0.978
15	leaping from 2D to 6D	2020	No	Yes	1/object	RGB	Synt+real	RGB	No	0.471	0.525	0.403	0.751	0.342	0.077	0.658	0.543	0.425
16	EPOS-BOP20-PBR	2020	No	Yes	1/dataset	RGB	PBR only	RGB	No	0.457	0.547	0.467	0.558	0.363	0.186	0.580	0.499	1.874
17	Drost-CVPR10-3D-Only-Faster	2019	Yes	No	-	-	-	D	ICP	0.454	0.492	0.405	0.696	0.377	0.274	0.603	0.330	1.383
18	Félix&Neves-ICRA2017-IET2019	2019	Yes	Yes	1/dataset	RGB-D	Synt+real	RGB-D	ICP	0.412	0.394	0.212	0.851	0.323	0.069	0.529	0.510	55.780
19	Sundermeyer-IJCV19+ICP	2019	No	Yes	1/object	RGB	Synt+real	RGB-D	ICP	0.398	0.237	0.487	0.614	0.281	0.158	0.506	0.505	0.865
20	Zhigang-CDPN-ICCV19	2019	No	Yes	1/object	RGB	Synt+real	RGB	No	0.353	0.374	0.124	0.757	0.257	0.070	0.470	0.422	0.513
21	PointVoteNet2	2020	No	Yes	1/object	RGB-D	PBR only	RGB-D	ICP	0.351	0.653	0.004	0.673	0.264	0.001	0.556	0.308	-
22	Pix2Pose-BOP20-ICCV19	2020	No	Yes	1/object	RGB	Synt+real	RGB	No	0.342	0.363	0.344	0.420	0.226	0.134	0.446	0.457	1.215
23	Sundermeyer-IJCV19	2019	No	Yes	1/object	RGB	Synt+real	RGB	No	0.270	0.146	0.304	0.401	0.217	0.101	0.346	0.377	0.186
24	SingleMultiPathEncoder-CVPR20	2020	No	Yes	1/all	RGB	Synt+real	RGB	No	0.241	0.217	0.310	0.334	0.175	0.067	0.293	0.289	0.186
25	Pix2Pose-BOP19-ICCV19	2019	No	Yes	1/object	RGB	Synt+real	RGB	No	0.205	0.077	0.275	0.349	0.215	0.032	0.200	0.290	0.793
26	DPOD (synthetic)	2019	No	Yes	1/scene	RGB	Synt	RGB	No	0.161	0.169	0.081	0.242	0.130	0.000	0.286	0.222	0.231

CosyPose variants: FocalPose, FocalPose++

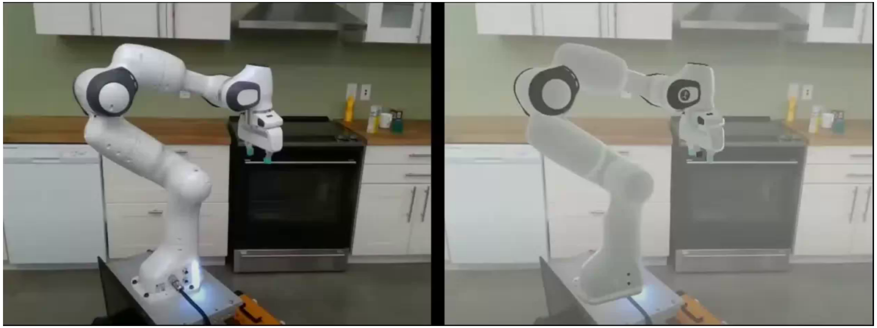




CosyPose variants: RoboPose



CosyPose variants: RoboPose



CosyPose limitations

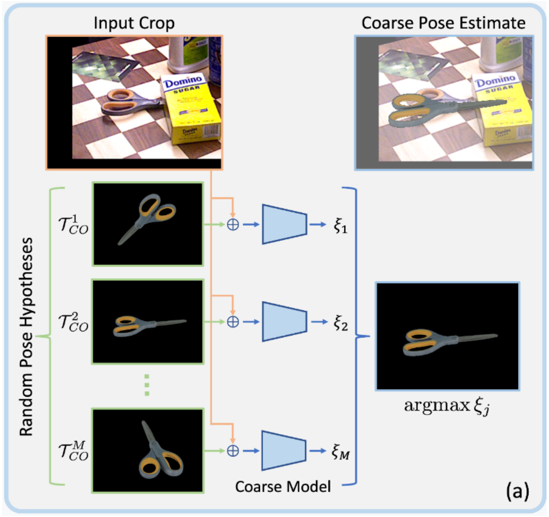
- Training time
- For each dataset
  - 10 hours on 32 GPUs for coarse estimator
  - 10 hours on 32 GPUs for refiner
- Coarse pose estimation often not accurate enough for refinement

MegaPose

6D Pose Estimation of Novel Objects via Render & Compare

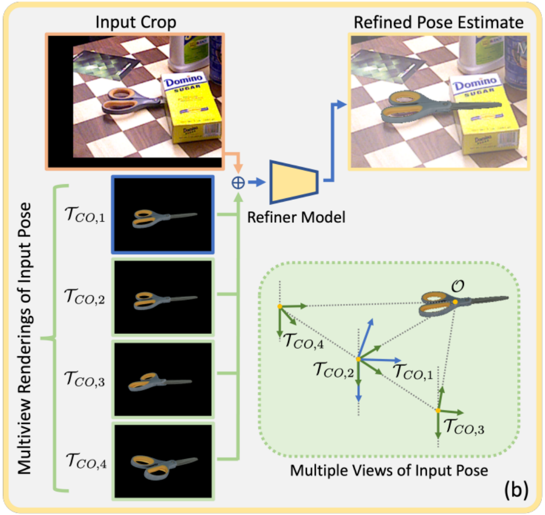
MegaPose - coarse estimation

- Re-casted estimation into classification
- Poses sampled randomly [original]
- Poses uniformly distributed [new]
- Allows multi-hypothesis evaluation



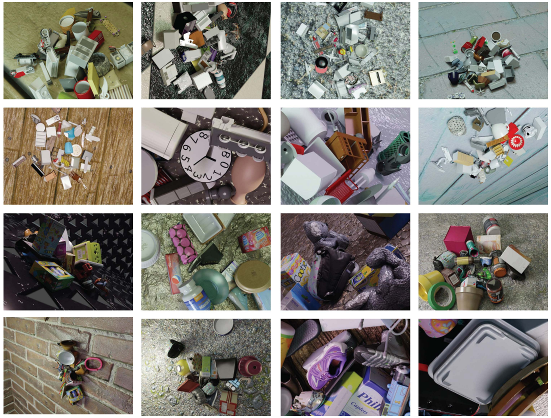
MegaPose - refiner

- Multi-view rendering
- Render and compare
- Iterative refinement



MegaPose - training data

- Generalization to unseen object achieved by big training dataset
  - only synthetic dataset
  - thousands of objects
  - 2 millions of images
- Training
  - 100 hours on 32 GPUs
  - trained only once, models are available



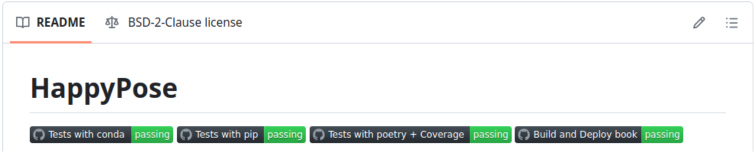
MegaPose - results

HappyPose

Open-source toolbox for 6D pose estimation

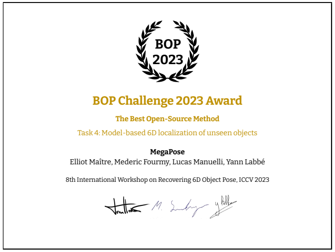
HappyPose

- Developed in AGIMUS project (https://github.com/agimus-project/happypose)
- Re-implements CosyPose and MegaPose
- Packaging, testing, documentation
- https://github.com/agimus-project/winter-school-2023/


README BSD-2-Clause license

HappyPose

Tests with conda passing | Tests with pip passing | Tests with poetry + Coverage passing | Build and Deploy book passing


BOP 2023

BOP Challenge 2023 Award

The Best Open-Source Method

Task 4: Model-based 6D localization of unseen objects

MegaPose

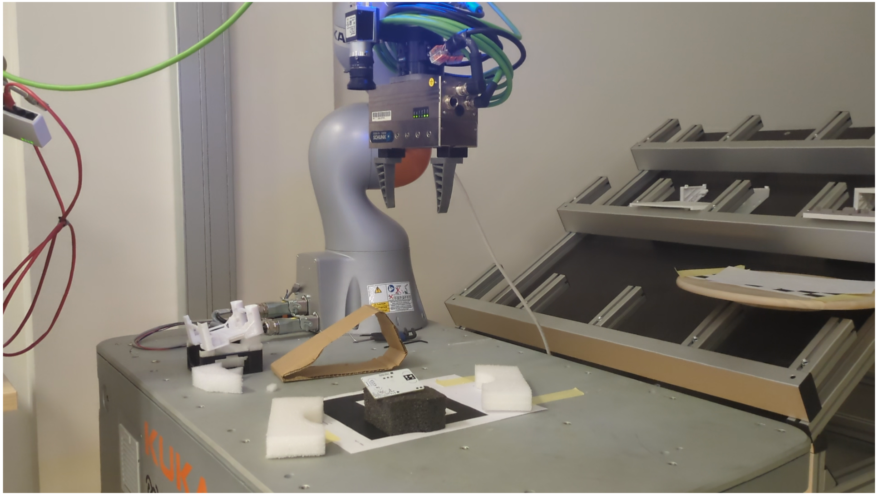
Elliot Maître, Mederic Fourmy, Lucas Manuelli, Yann Labbé

8th International Workshop on Recovering 6D Object Pose, ICCV 2023

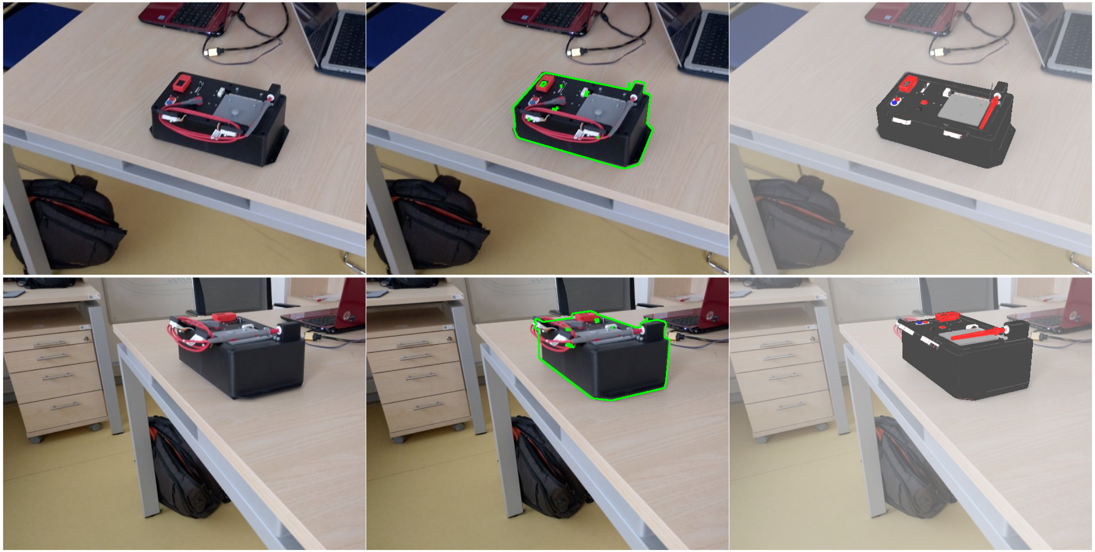
[illegible]

Applications

PCB manipulation based on the estimated pose

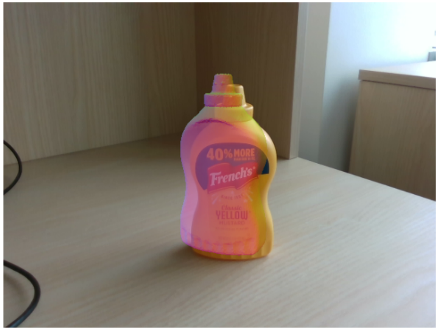


euROBIN taskboard pose estimation



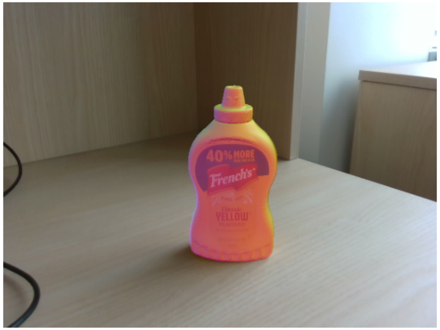
Model-based object pose tracking

Object pose tracking



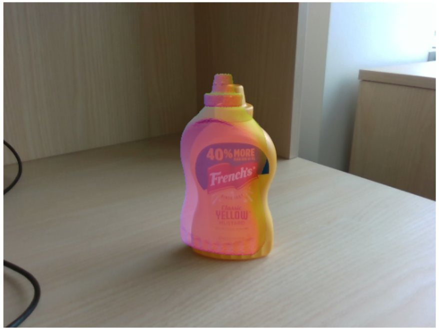
Initial pose

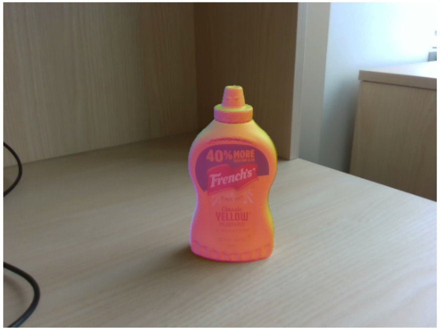
$\longrightarrow$



Converged

Object pose tracking

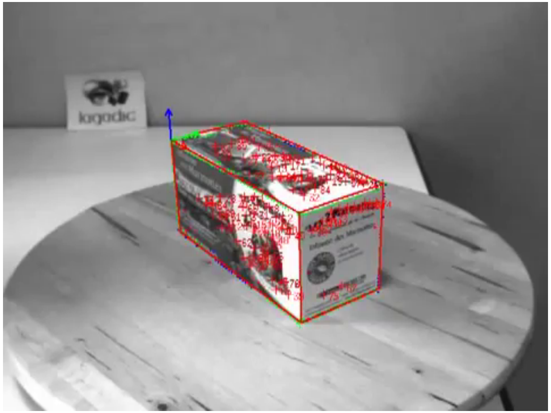

Initial pose


Converged

- Assumptions: object detected, matched with model, initial pose given

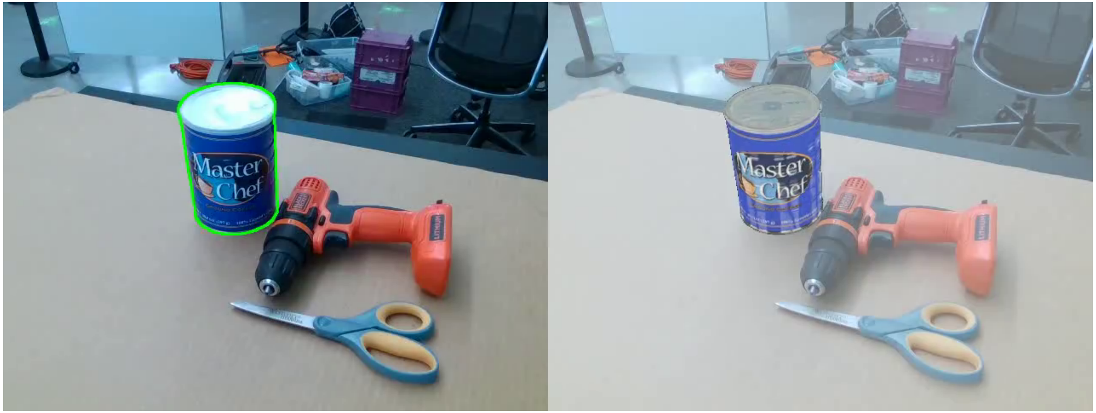
Keypoint matching approach

- Model
  - 3D points on mesh
  - descriptors of points
- Method
  - 3D-2D matching
  - minimize reprojection error
- Efficient and robust for rich textures



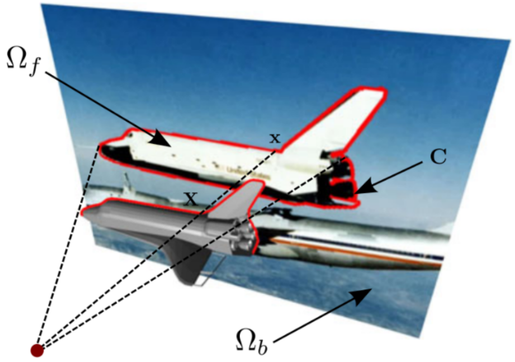
MegaPose as tracking?

MegaPose as tracking?



Region based tracking

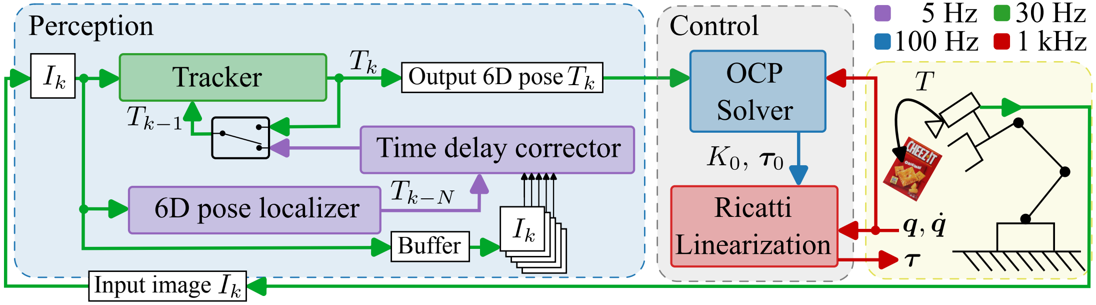
- Mesh model as input
- Probabilistic silhouette alignment (Newton's method)
- Assumes foreground and background colors sufficiently different
- Robust to occlusion, efficient



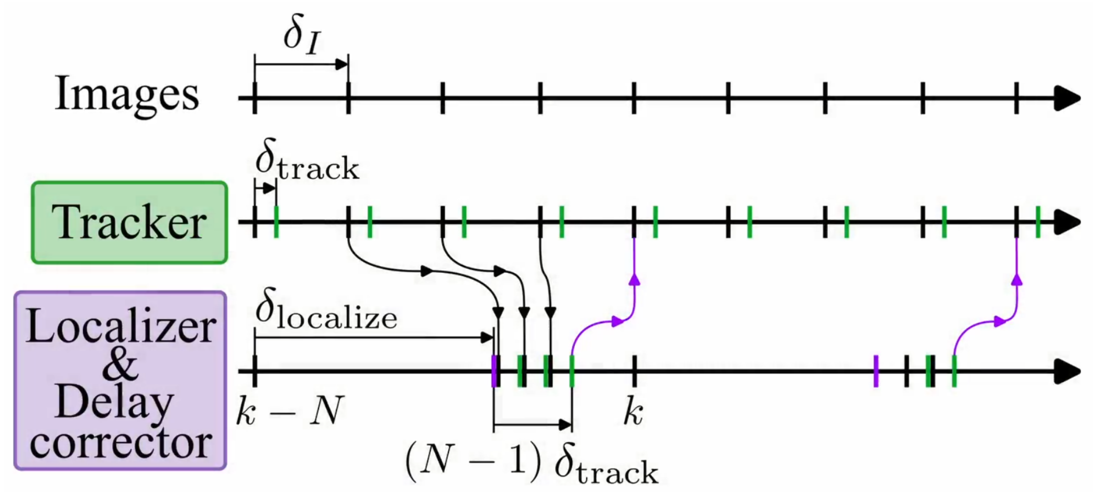
Region based tracker

Object localization and tracking

- Combines slow localization and fast tracker
- Goal: fast feedback for control

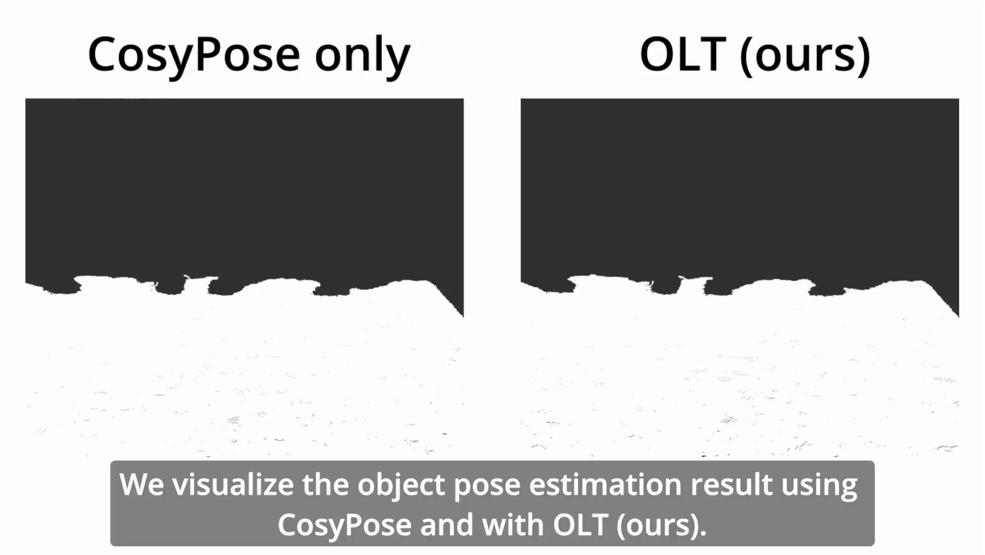


OLT timeline



δ_I
Images
δ_{track}
Tracker
δ_{localize}
Localizer & Delay corrector
$k-N$
k
$(N-1)\,\delta_{\text{track}}$

OLT delay



We visualize the object pose estimation result using CosyPose and with OLT (ours).

Control

- Optimal control solver

$$\begin{aligned} \underset{\substack{\boldsymbol{u}_0,\dots,\boldsymbol{u}_{M-1}\\ \boldsymbol{x}_1,\dots,\boldsymbol{x}_M}}{\arg\min} \; & \sum_{i=0}^{M-1} l_i(\boldsymbol{x}_i, \boldsymbol{u}_i) + l_M(\boldsymbol{x}_M)\,, \\ \text{s.t.} \quad \boldsymbol{x}_{i+1} &= f(\boldsymbol{x}_i, \boldsymbol{u}_i), \forall i \in \{0, \dots, M-1\}, \\ \boldsymbol{x}_0 &= \hat{\boldsymbol{x}}\,, \end{aligned} \tag{1}$$

Control

- Optimal control solver

$$\begin{aligned}
&\arg\min_{\substack{\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{M-1}\\ \boldsymbol{x}_1,\ldots,\boldsymbol{x}_M}} \sum_{i=0}^{M-1} l_i(\boldsymbol{x}_i, \boldsymbol{u}_i) + l_M(\boldsymbol{x}_M)\,, \\
&\text{s.t.} \quad \boldsymbol{x}_{i+1} = f(\boldsymbol{x}_i, \boldsymbol{u}_i),\ \forall i \in \{0, \ldots, M-1\}, \\
&\qquad\quad \boldsymbol{x}_0 = \hat{\boldsymbol{x}}\,,
\end{aligned} \tag{1}$$

- Ricatti linearization

$$\boldsymbol{\tau}(\boldsymbol{x}) = \boldsymbol{\tau}_0 + K_0(\boldsymbol{x} - \boldsymbol{x}_0) \tag{2}$$

Control

- Optimal control solver

$$\begin{aligned}
&\underset{\substack{\boldsymbol{u}_0,\dots,\boldsymbol{u}_{M-1}\\ \boldsymbol{x}_1,\dots,\boldsymbol{x}_M}}{\arg\min} \sum_{i=0}^{M-1} l_i(\boldsymbol{x}_i,\boldsymbol{u}_i) + l_M(\boldsymbol{x}_M)\,,\\
\text{s.t.}\quad &\boldsymbol{x}_{i+1} = f(\boldsymbol{x}_i,\boldsymbol{u}_i),\ \forall i \in \{0,\dots,M-1\},\\
&\boldsymbol{x}_0 = \hat{\boldsymbol{x}}\,,
\end{aligned} \tag{1}$$

- Ricatti linearization

$$\boldsymbol{\tau}(\boldsymbol{x}) = \boldsymbol{\tau}_0 + K_0(\boldsymbol{x} - \boldsymbol{x}_0) \tag{2}$$

Costs for optimal control

- Tracking cost

$$\left\| \log \left(\left(T_{\mathsf{BC}}(\boldsymbol{q}_k) T_k \right)^{-1} T_{\mathsf{BC}}(\boldsymbol{q}) T_{\mathsf{ref}} \right) \right\|^2 \tag{3}$$

Costs for optimal control

- Tracking cost

$$\left\| \log \left(\left(T_{\mathsf{BC}}(\boldsymbol{q}_k) T_k \right)^{-1} T_{\mathsf{BC}}(\boldsymbol{q}) T_{\mathsf{ref}} \right) \right\|^2 \tag{3}$$

  - is solution unique?

Costs for optimal control

- Tracking cost

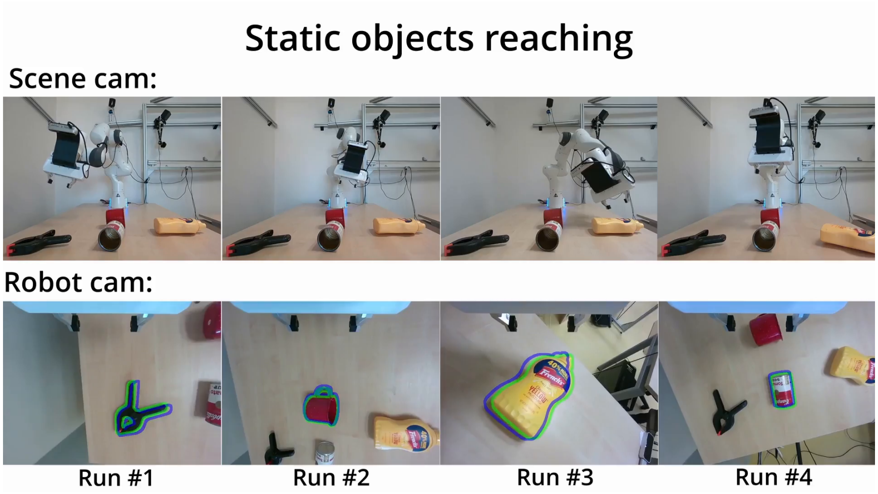
$$\left\|\log\left(\left(T_{\mathsf{BC}}(\boldsymbol{q}_k)T_k\right)^{-1}T_{\mathsf{BC}}(\boldsymbol{q})T_{\mathsf{ref}}\right)\right\|^2 \tag{3}$$

  - is solution unique?
- Regularizations:

$$\left(\boldsymbol{x}-\boldsymbol{x}_{\mathsf{rest}}\right)^\top Q_x\left(\boldsymbol{x}-\boldsymbol{x}_{\mathsf{rest}}\right) \tag{4}$$

$$\left(\boldsymbol{u}-\boldsymbol{u}_{\mathsf{rest}}(\boldsymbol{x})\right)^\top Q_u\left(\boldsymbol{u}-\boldsymbol{u}_{\mathsf{rest}}(\boldsymbol{x})\right) \tag{5}$$

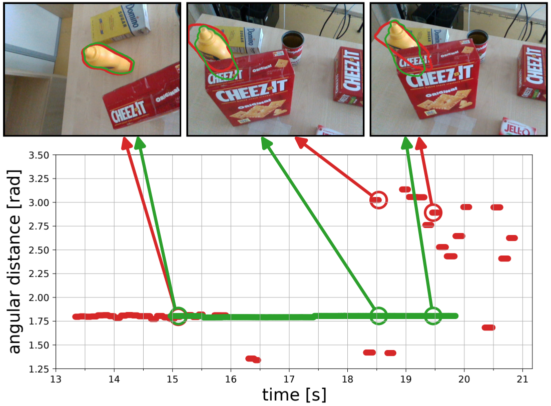
OLT with control for tracking



Temporal consistency

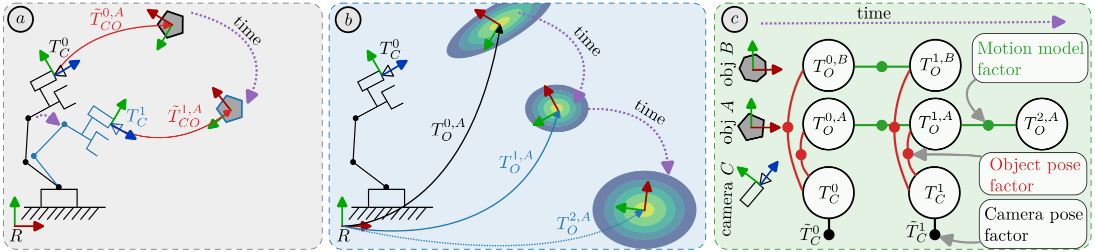
Temporal consistency for 6D pose estimation

Temporal consistency



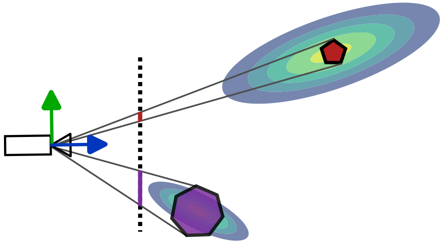
- Use smoothing and mapping with CosyPose to achieve temporal consistency
- Probabilistic smoothing
  - occlusions
  - jumps
- Bachelor Thesis of Vojtěch Přibáň, published in IEEE RA-L journal

Approach



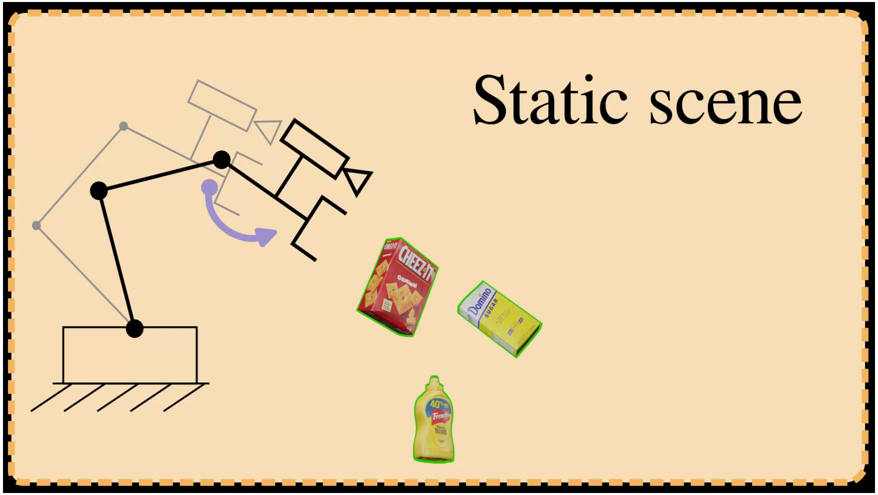
$$\chi^* = \arg\min_{\chi} \underbrace{\sum_{k=\tau-H}^{\tau} \left\| \boldsymbol{r}_C^k \right\|_{\boldsymbol{\Sigma}_C}^2}_{\text{camera pose factors}} + \underbrace{\sum_{i=1}^{N} \sum_{k=\tau-H}^{\tau} \delta^{k,i} \left\| \boldsymbol{r}_O^{k,i} \right\|_{\boldsymbol{\Sigma}_O}^2}_{\text{object pose factors}} + \underbrace{\sum_{i=1}^{N} \sum_{k=\tau-H+1}^{\tau} \left\| \boldsymbol{r}_M^{k-1:k,i} \right\|_{\boldsymbol{\Sigma}_M}^2}_{\text{motion model factors}}$$

Covariance model

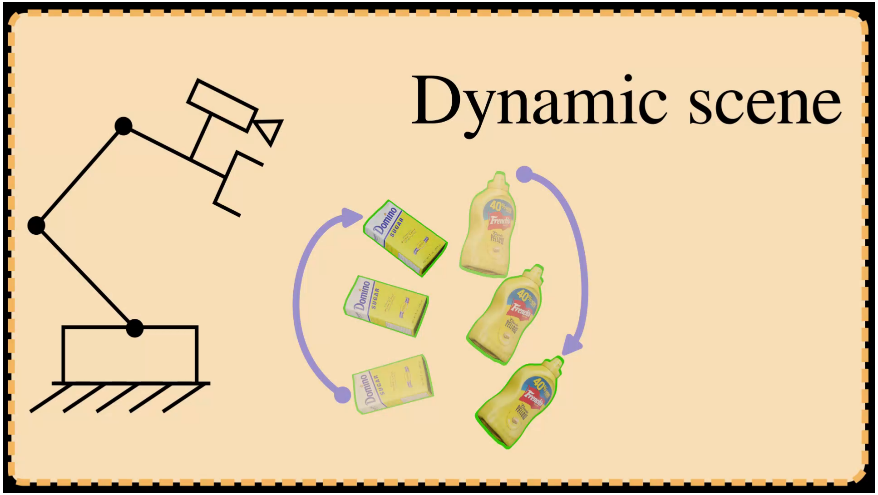


Decoupled	Visibility dependent	frame C'	recall	precision
✓	✓	✓	**0.571**	**0.609**
✓	×	✓	0.570	0.608
✓	✓	×	0.531	0.574
×	✓	N/A	0.483	0.549
×	×	N/A	0.498	0.542

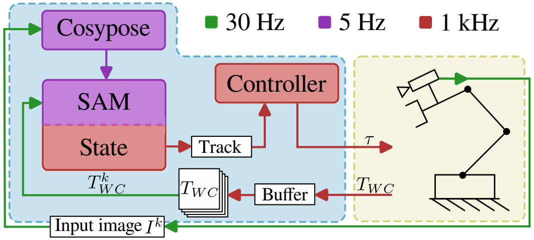
Qualitative static objects tracking

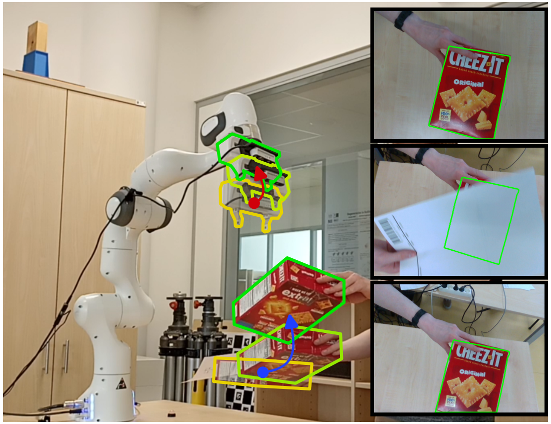


Qualitative dynamic objects tracking

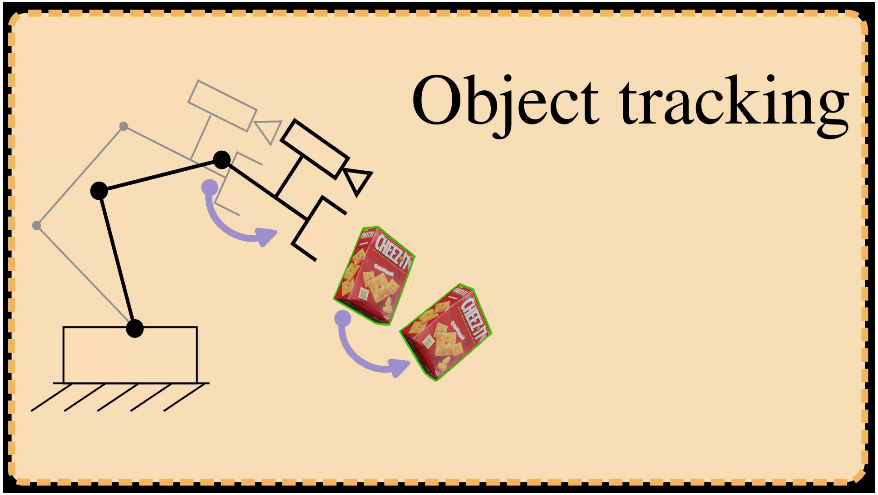


Robot control architecture





Qualitative robot tracking

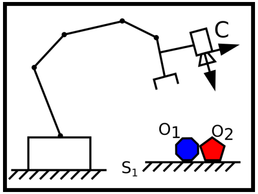


Geometrical consistency

Geometrical consistency for 6D pose estimation

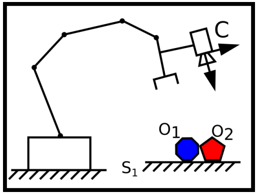
Geometrical consistency for object pose estimation from images

- Image based robotic manipulation



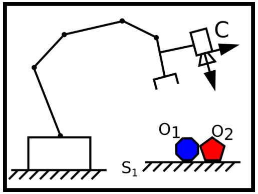
Geometrical consistency for object pose estimation from images

- Image based robotic manipulation
- Pose estimation from single RGB image



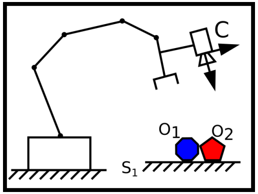
Geometrical consistency for obje
images

- Image based robotic manipulation
- Pose estimation from single RGB image



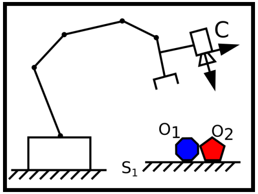
Geometrical consistency for object pose estimation from images

- Image based robotic manipulation
- Pose estimation from single RGB image
- Physical consistency



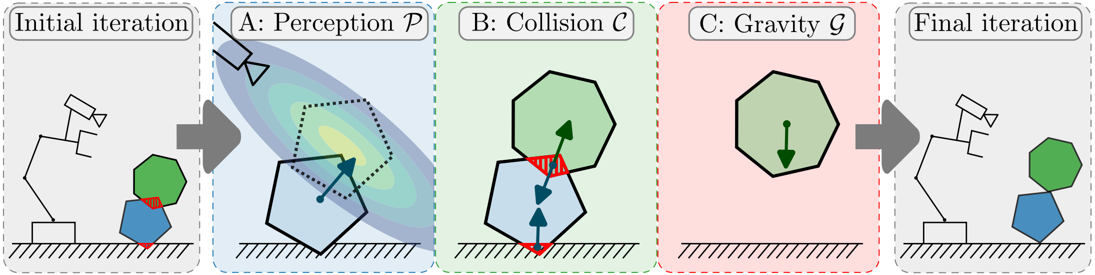
Geometrical consistency for obje images

- Image based robotic manipulation
- Pose estimation from single RGB image
- Physical consistency
- Bachelor Thesis of Martin Malenický

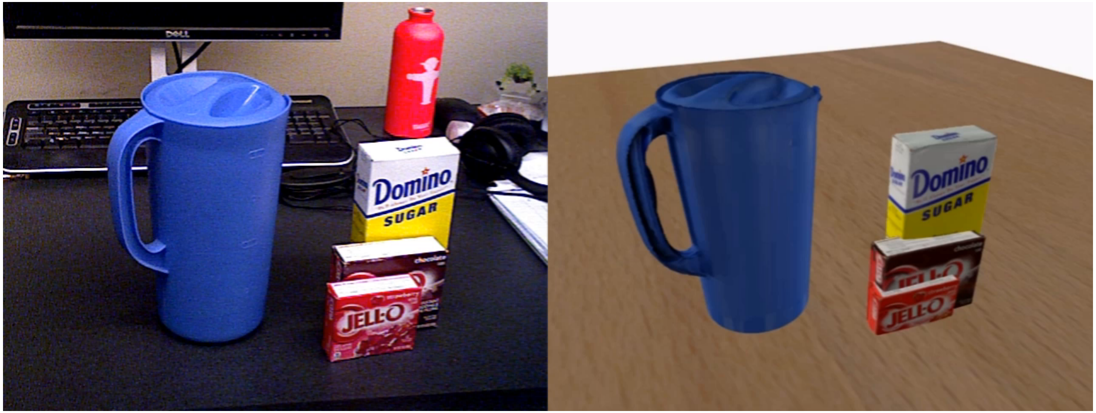


Approach

- Gradient descent optimization with derived analytical gradients



Visualization of optimization



Quantitative experiments

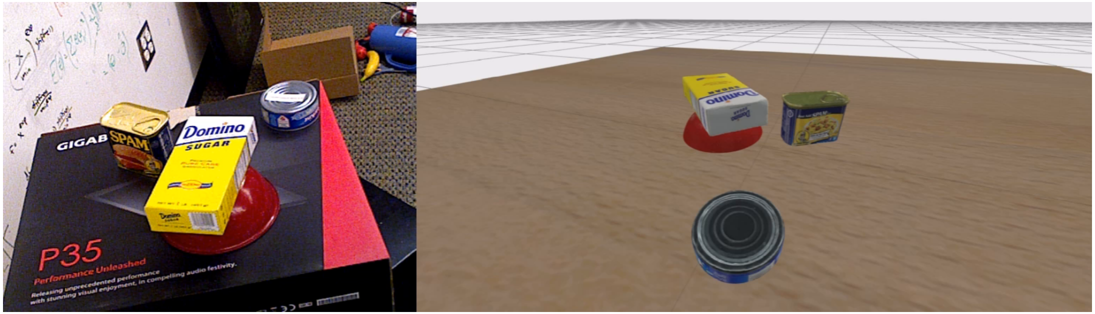
- Real BOP datasets:
  - YCB-V
  - HOPE-Video
  - T-LESS
- Synthetic datasets:
  - YCB
  - T-LESS

Quantitative experiments

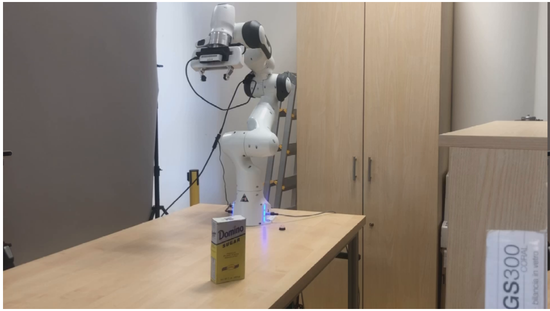
- Real BOP datasets:
  - YCB-V
  - HOPE-Video
  - T-LESS
- Synthetic datasets:
  - YCB
  - T-LESS

	real datasets	synthetic datasets
MegaPose	0.71	0.76
Ours	0.80	0.94
Ours improovement [%]	**12.7**	**23.7**

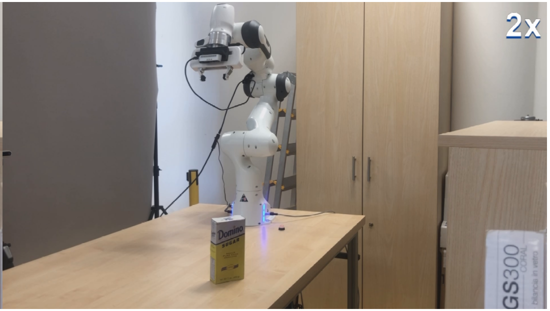
Visualization of optimization



Grasping example



MegaPose



Ours

From OC to RL

From OC to RL

Optimal control - Model Predictive Control

- Find optimal control sequence $\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_T$ to minimize cost function J

Optimal control - Model Predictive Control

- Find optimal control sequence $\boldsymbol{u}_0, \boldsymbol{u}_1, \dots, \boldsymbol{u}_T$ to minimize cost function J
  - $\boldsymbol{u}^* = \underset{\boldsymbol{u}_0,\dots,\boldsymbol{u}_{T-1}}{\arg\min}\ J(\boldsymbol{x}_0, \dots, \boldsymbol{x}_T, \boldsymbol{u}_0, \dots, \boldsymbol{u}_T)$ s.t. $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
  - $\boldsymbol{x}_t$ is state of the system at time t
  - $\boldsymbol{u}$ is control (torque, velocity, ...)
  - $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is dynamics/simulation of the system

Optimal control - Model Predictive Control

- Find optimal control sequence $\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_T$ to minimize cost function J
  - $\boldsymbol{u}^* = \underset{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{T-1}}{\arg\min} \; J(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T, \boldsymbol{u}_0, \ldots, \boldsymbol{u}_T)$ s.t. $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
  - $\boldsymbol{x}_t$ is state of the system at time t
  - $\boldsymbol{u}$ is control (torque, velocity, $\ldots$)
  - $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is dynamics/simulation of the system
- Cost function:
  - $J = \sum_{t=0}^{T-1} l(\boldsymbol{x}_t, \boldsymbol{u}_t) + l_T(\boldsymbol{x}_T)$
  - l is cost function at time t
  - l_T is terminal cost function
  - T is time horizon

Optimal control - Model Predictive Control

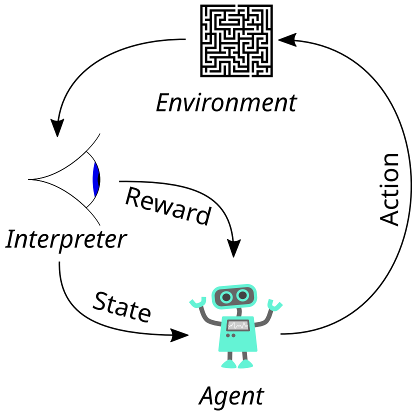
- Find optimal control sequence $\boldsymbol{u}_0, \boldsymbol{u}_1, \dots, \boldsymbol{u}_T$ to minimize cost function J
  - $\boldsymbol{u}^* = \underset{\boldsymbol{u}_0,\dots,\boldsymbol{u}_{T-1}}{\arg\min}\; J(\boldsymbol{x}_0, \dots, \boldsymbol{x}_T, \boldsymbol{u}_0, \dots, \boldsymbol{u}_T)$ s.t. $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
  - $\boldsymbol{x}_t$ is state of the system at time t
  - $\boldsymbol{u}$ is control (torque, velocity, ...)
  - $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is dynamics/simulation of the system
- Cost function:
  - $J = \sum_{t=0}^{T-1} l(\boldsymbol{x}_t, \boldsymbol{u}_t) + l_T(\boldsymbol{x}_T)$
  - l is cost function at time t
  - l_T is terminal cost function
  - T is time horizon
- Use numerical optimization to solve the minimization problem
  - dynamics (f) and costs (l, l_T) needs to be differentiable

Optimal control - Model Predictive Control

- Find optimal control sequence $\boldsymbol{u}_0, \boldsymbol{u}_1, \dots, \boldsymbol{u}_T$ to minimize cost function J
  - $\boldsymbol{u}^* = \underset{\boldsymbol{u}_0,\dots,\boldsymbol{u}_{T-1}}{\arg\min} \; J(\boldsymbol{x}_0, \dots, \boldsymbol{x}_T, \boldsymbol{u}_0, \dots, \boldsymbol{u}_T)$ s.t. $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
  - $\boldsymbol{x}_t$ is state of the system at time t
  - $\boldsymbol{u}$ is control (torque, velocity, ...)
  - $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is dynamics/simulation of the system
- Cost function:
  - $J = \sum_{t=0}^{T-1} l(\boldsymbol{x}_t, \boldsymbol{u}_t) + l_T(\boldsymbol{x}_T)$
  - l is cost function at time t
  - l_T is terminal cost function
  - T is time horizon
- Use numerical optimization to solve the minimization problem
  - dynamics (f) and costs (l, l_T) needs to be differentiable
  - what if we do not have gradient of dynamics/costs?

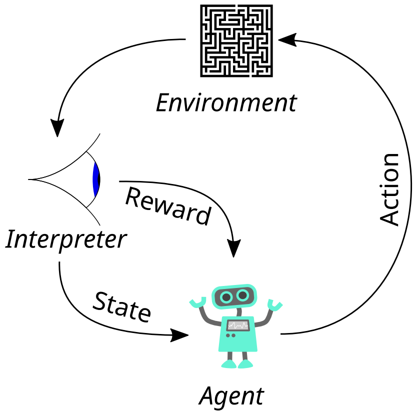
Reinforcement learning

- Modeled as Markov Decision Process



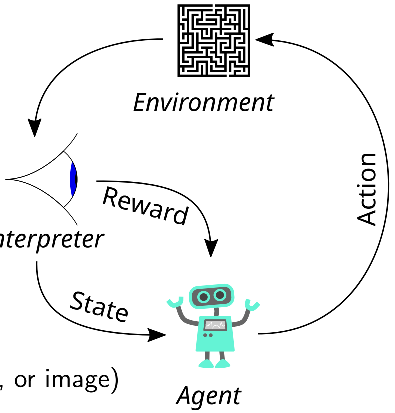
Reinforcement learning

- Modeled as Markov Decision Process
- Agent interacts with environment
- Agent receives reward for each action/state

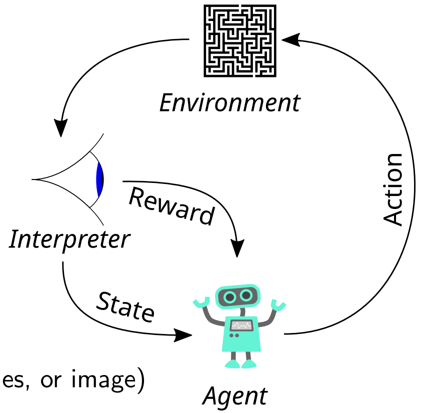


Reinforcement learning

- Modeled as Markov Decision Process
- Agent interacts with environment
- Agent receives reward for each action/state
- Goal is to find policy that maximizes reward in time
- Stochastic policy: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
  - $\boldsymbol{a}$ is action (e.g. torque)
  - $\boldsymbol{s}$ is state of the system (e.g. joint angles and velocities, or image)
  - π_θ is policy parameterized by θ



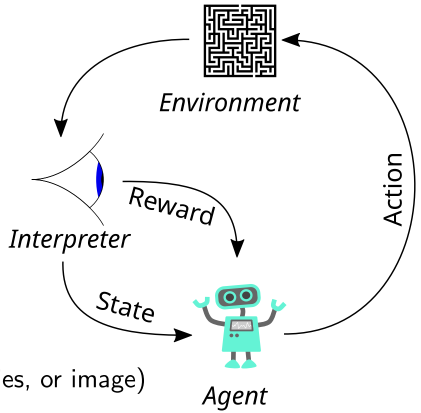
Reinforcement learning



- Modeled as Markov Decision Process
- Agent interacts with environment
- Agent receives reward for each action/state
- Goal is to find policy that maximizes reward in time
- Stochastic policy: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
  - $\boldsymbol{a}$ is action (e.g. torque)
  - $\boldsymbol{s}$ is state of the system (e.g. joint angles and velocities, or image)
  - π_θ is policy parameterized by θ
- Instantaneous reward: $r(\boldsymbol{s}, \boldsymbol{a})$
- Expected return of the policy: $R = \mathbb{E}_{\boldsymbol{a}_t \sim \pi_\theta(\boldsymbol{s}_t)} \left[\sum_t r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]$ s.t. $\boldsymbol{s}_{t+1} \sim f(\boldsymbol{s}_t, \boldsymbol{a}_t)$

Reinforcement learning

- Modeled as Markov Decision Process
- Agent interacts with environment
- Agent receives reward for each action/state
- Goal is to find policy that maximizes reward in time
- Stochastic policy: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
  - $\boldsymbol{a}$ is action (e.g. torque)
  - $\boldsymbol{s}$ is state of the system (e.g. joint angles and velocities, or image)
  - π_θ is policy parameterized by θ
- Instantaneous reward: $r(\boldsymbol{s}, \boldsymbol{a})$
- Expected return of the policy: $R = \mathbb{E}_{\boldsymbol{a}_t \sim \pi_\theta(\boldsymbol{s}_t)} \left[\sum_t r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]$ s.t. $\boldsymbol{s}_{t+1} \sim f(\boldsymbol{s}_t, \boldsymbol{a}_t)$
- Goal: $\arg\max_\theta R$
- Compare to MPC: $\arg\min_{\boldsymbol{u}_1,\dots,\boldsymbol{u}_T} J$ s.t. $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$



Policy gradient

- Policy π_θ is parameterized by θ
- Is used to sample action $\boldsymbol{a}$ given state $\boldsymbol{s}$: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$

Policy gradient

- Policy π_θ is parameterized by θ
- Is used to sample action $\boldsymbol{a}$ given state $\boldsymbol{s}$: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
- Gradient descent algorithm: $\theta_{t+1} = \theta_t + \alpha \nabla_\theta R(\pi_\theta)$
  - θ parameterizes policy π_θ
  - α is learning rate

Policy gradient

- Policy π_θ is parameterized by θ
- Is used to sample action $\boldsymbol{a}$ given state $\boldsymbol{s}$: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
- Gradient descent algorithm: $\theta_{t+1} = \theta_t + \alpha \nabla_\theta R(\pi_\theta)$
    - θ parameterizes policy π_θ
    - α is learning rate
    - $\nabla_\theta R(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_t \nabla_\theta \log \pi_\theta(\boldsymbol{s}_t) r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right]$
    - expectation over trajectories τ sampled by following policy π_θ

Policy gradient

- Policy π_θ is parameterized by θ
- Is used to sample action $\boldsymbol{a}$ given state $\boldsymbol{s}$: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
- Gradient descent algorithm: $\theta_{t+1} = \theta_t + \alpha \nabla_\theta R(\pi_\theta)$
  - θ parameterizes policy π_θ
  - α is learning rate
  - $\nabla_\theta R(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_t \nabla_\theta \log \pi_\theta(\boldsymbol{s}_t) r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right]$
  - expectation over trajectories τ sampled by following policy π_θ
  - in practise expectation is approximated by sampling a lot of trajectories (millions)
  - why we need stochastic policy?

Policy gradient

- Policy π_θ is parameterized by θ
- Is used to sample action $\boldsymbol{a}$ given state $\boldsymbol{s}$: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
- Gradient descent algorithm: $\theta_{t+1} = \theta_t + \alpha \nabla_\theta R(\pi_\theta)$
  - θ parameterizes policy π_θ
  - α is learning rate
  - $\nabla_\theta R(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_t \nabla_\theta \log \pi_\theta(\boldsymbol{s}_t) r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right]$
  - expectation over trajectories τ sampled by following policy π_θ
  - in practise expectation is approximated by sampling a lot of trajectories (millions)
  - why we need stochastic policy?
- Can we apply millions of trajectories to real robot?

Policy gradient

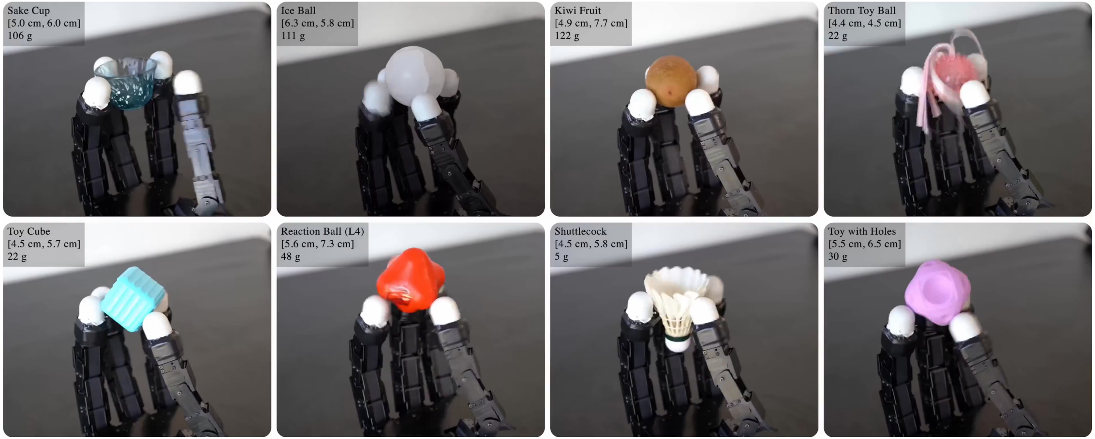
- Policy π_θ is parameterized by θ
- Is used to sample action $\boldsymbol{a}$ given state $\boldsymbol{s}$: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
- Gradient descent algorithm: $\theta_{t+1} = \theta_t + \alpha \nabla_\theta R(\pi_\theta)$
  - θ parameterizes policy π_θ
  - α is learning rate
  - $\nabla_\theta R(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_t \nabla_\theta \log \pi_\theta(\boldsymbol{s}_t) r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right]$
  - expectation over trajectories τ sampled by following policy π_θ
  - in practise expectation is approximated by sampling a lot of trajectories (millions)
  - why we need stochastic policy?
- Can we apply millions of trajectories to real robot?
- We need fast and accurate simulation of robots
  - Gazebo
  - NVIDIA Isaac Sim

Example of RL

Example of RL

Example of RL

Example of RL



Reward shaping

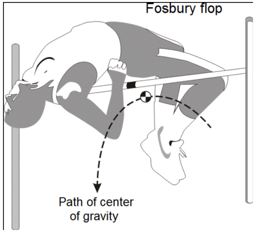
- Finding solution to RL problem is hard
  - sparse reward
  - local minima
  - long training time

Reward shaping

- Finding solution to RL problem is hard
  - sparse reward
  - local minima
  - long training time
- Reward shaping
  - add additional reward to the original reward
  - additional reward is designed to guide learning and avoid local minima
  - engineering work
- Is there a better solution?

Reward shaping

- Finding solution to RL problem is hard
  - sparse reward
  - local minima
  - long training time
- Reward shaping
  - add additional reward to the original reward
  - additional reward is designed to guide learning and avoid local minima
  - engineering work
- Is there a better solution? Learning from demonstration.
- Example from high-jump (Fosbury flop - 1968 gold medal)



Offline reinforcement learning - Learning from demonstration

- Collect data from real robot guided by the operator
- Pre-Train policy on collected data
- Optionally, fine-tune policy in simulation/ on real robot
- How to pre-train policy?

Offline reinforcement learning - Learning from demonstration

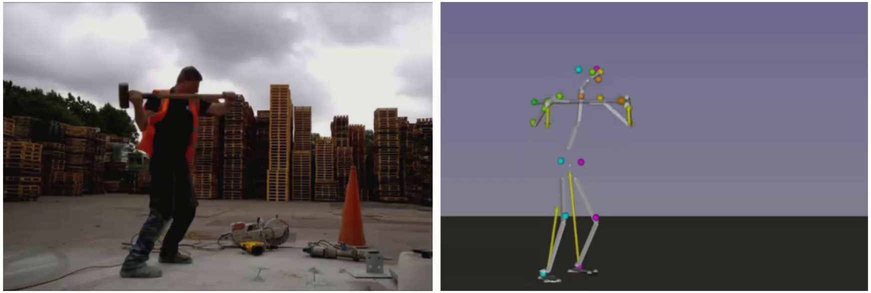
- Collect data from real robot guided by the operator
- Pre-Train policy on collected data
- Optionally, fine-tune policy in simulation/ on real robot
- How to pre-train policy?
  - behavior cloning - supervised learning
  - $\arg\min_{\theta} \sum_{i=1}^{N} (\pi_\theta(s_i) - a_i)^2$
  - diffusion policy - supervised learning

Learning from video

- Instructional videos are widely available on YouTube
- Can we learn from them?

Learning from video

- Instructional videos are widely available on YouTube
- Can we learn from them?
- Depends on the task/video, e.g. if human is visible
  - we can extract human pose from video
  - we can extract the manipulated object pose
  - we can extract interaction forces



Learning tool manipulation from instructional video

Learning to Use Tools by Watching Videos

Input: instructional video from YouTube

Output: tool manipulation skill transferred to a robot

Summary

- 6D pose estimation
  - Object detection
  - CosyPose
  - MegaPose
  - FocalPose
  - RoboPose
- 6D pose tracking
- Object localization and tracking for control
- Temporal/Geometrical consistency for pose estimation
- Reinforcement learning