Physics-Based Object 6D-Pose Estimation during Non-Prehensile Manipulation

Zisong Xu, Rafael Papallas, and Mehmet R. Dogar

University of Leeds, LS2 9JT, UK
{sc19zx, r.papallas, m.r.dogar}@leeds.ac.uk

Abstract. We propose a method to track the 6D pose of an object over time, while the object is under non-prehensile manipulation by a robot. At any given time during the manipulation of the object, we assume access to the robot joint controls and an image from a camera. We use the robot joint controls to perform a physics-based prediction of how the object might be moving. We then combine this prediction with the observation coming from the camera, to estimate the object pose as accurately as possible. We use a particle filtering approach to combine the control information with the visual information. We compare the proposed method with two baselines: (i) using only an image-based pose estimation system at each time-step, and (ii) a particle filter which does not perform the computationally expensive physics predictions, but assumes the object moves with constant velocity. Our results show that making physics-based predictions is worth the computational cost, resulting in more accurate tracking, and estimating object pose even when the object is not clearly visible to the camera.

Keywords: Non-prehensile Manipulation, Object-Pose Estimation

1 Introduction

The dominant approach in robotic manipulation applications has been estimating the object pose from RGB(D) camera images [1]. Object pose is estimated using deep learning (e.g. DOPE [2]) or feature-based [3] methods.

However, during non-prehensile manipulation [4,5,18] in clutter, objects are often fully or partially obstructed from the camera view, by cluttering objects (e.g., Fig. 1, top-row) or by the robot hand (e.g., Fig. 1, bottom-row). This degrades the performance of vision-only pose estimation.

The key insight we have is that, during non-prehensile manipulation, physics-based predictions about objects' motion can be used to supplement the visual information, to better estimate object pose even when there are significant visual

This research has received funding from the UK Engineering and Physical Sciences Research Council under the grant EP/V052659/1. Associated code and data available at: https://github.com/ZisongXu/trackObjectWithPF For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) license to any Author Accepted Manuscript version arising.

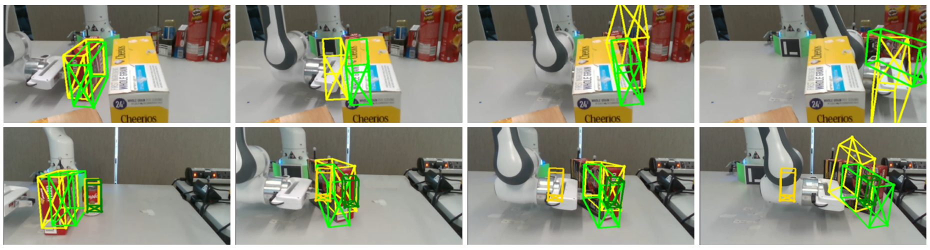

Fig. 1. Two example scenes. The estimated object pose of our method (PBPF) is shown as a green wireframe box. The pose from a vision-only system (DOPE) is shown as a yellow wireframe box. See the attached video or at: https://youtu.be/B9-5iTwgFnk.

occlusions. We propose a particle-filtering method that combines physics-based predictions (using the available robot joint information and a physics-engine) with the visual camera information to estimate the pose of manipulated objects. We show two example scenes in Fig. 1. In the top-row, the robot is pushing a red box. As the pushed object becomes visually obstructed by the box closer to the camera, the vision-only system fails to estimate the object's pose, whereas our method can generate good estimates. In the bottom-row, two objects are manipulated: a red box, which then pushes against a red can. As the push progresses, the robot hand visually obstructs the red box and the red box visually obstructs the can. Our method uses the physics-based interaction between the red box and the can (in addition to the robot-red box interaction) to estimate the poses of both objects.

A similar work that combines visual information with physics-based reasoning for 6D pose estimation in clutter is [11]. However, this work addresses static scenes (e.g. based on the physical stability of a pile of objects). Instead, our work uses physics predictions as the robot is manipulating/moving the objects. Other work also uses physics and combines it with contact/tactile sensing [13,7,12,17], as robot-object contact forces are tightly related to the pose and motion of the object. In our work, we do not assume we have contact sensing, but instead, we use the more commonly available camera sensing.

Other vision-based object tracking systems exist, sometimes using particle filtering similar to our work [15,16,19,20], but they either assume the object does not move [6] (but the camera may move as in [14]), or they [10] assume constant velocity. Recent work from [21] proposes a method to track the pose of a box undergoing a non-smooth impact with a surface (e.g. a box tossed onto a surface) using a camera. The work derives and uses a specific impact model for the box-surface system. The main difference of our work to the above is that, in addition to visual information, we use robot control information and its physical effects. To the best of our knowledge, this is the first work that integrates general physics-based predictions with pose estimation.

2 Problem Formulation

We use $x_t \in SE(3)$ to represent the full 6D pose of the object at time t. Our objective is to estimate x_t for all times t during manipulation, i.e. to track the object pose as it is being manipulated. At any time t, we assume we have access to:

- the controls u_t, which are the joint-space controls executed by the robot since the last time-step $t-1$;
- the observation z_t, which is an image from a camera looking at the scene.

Therefore, our problem can be formalized as: At any time t, given all controls since the beginning of the manipulation $\{u_0, u_1, \ldots, u_t\}$ and all observations $\{z_0, z_1, \ldots, z_t\}$, predict an estimate of the object pose, $\tilde{x}_t$.

We use x_t^* to represent the ground truth pose of the object. We define the *positional error* as the Euclidean distance between the positional components of $\tilde{x}_t$ and x_t^*. We define the *rotational error* as the (smaller) angle required to represent the relative rotation of $\tilde{x}_t$ and x_t^* around a certain axis: Given $\tilde{q}_t$ and q_t^* as the quaternion components of the estimated and ground truth poses, respectively, the rotational error is given by the amount of rotation suggested by $q_t^* * \tilde{q}_t^{-1}$. Our goal is to minimize positional and rotational errors.

Below, we discuss three different methods (two baselines and our proposed method): (i) Repetitive single-snapshot pose estimation (baseline, in Sec. 3), (ii) Physics-based particle filtering (proposed method, in Sec. 4), and (iii) Constant-velocity particle filtering (baseline, in Sec. 5).

3 Repetitive Single-Snapshot Pose Estimation

One commonly used method in robotic manipulation is to use a system that can estimate the pose of the object from a single-snapshot, i.e. using only the current camera image z_t. There are multiple systems developed to perform object pose estimation given an image, as mentioned in Sec. 1. In this paper, we use DOPE [2] as a state-of-the-art deep-learning-based pose estimation system. DOPE has been trained on the YCB objects [8] that we also use, and is publicly available. Therefore, the repetitive single-snapshot pose estimation method, in our case, corresponds to running DOPE at every time-step t during manipulation: $\tilde{x}_t = DOPE(z_t)$. This method treats every new observation as independent from the previous time steps and therefore does not use temporal continuity. The performance particularly degrades when the object is partially or fully obstructed in the camera view. Moreover, it does not use the control information.

4 Physics-Based Particle Filtering (PBPF)

Using a Bayesian approach, we estimate the probability distribution for x_t given all the previous controls and observations, $p(x_t \,|\, z_0, z_1, ..., z_t, u_0, u_1, ..., u_t)$, i.e., the *belief state*. In particle filtering [22], the belief state is represented using a set of *particles* sampled from this distribution, $\mathcal{X}_t := x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]}$, where each particle $x_t^{[m]}$ $(1 \leq m \leq M)$ is an instance of the state at time t. In our setting, each particle represents a possible pose for all the manipulated objects.

At any time t, if a single estimate $\tilde{x}_t$ is required (e.g. the estimates shown in Fig. 1), we use the mean of all the particles in $\mathcal{X}_t$ to compute $\tilde{x}_t$.

During particle filtering, at each time-step t, the previous set of particles $\mathcal{X}_{t-1}$ are updated using the current controls, u_t, and observation, z_t, to generate a new set of particles $\mathcal{X}_t$. This happens in two stages: the *motion update* (presented in Sec. 4.1), and the *observation update* (in Sec. 4.2).

4.1 Motion Update

During this first stage, for each particle $x_{t-1}^{[m]}$, we generate a new intermediate particle (shown as $\bar{x}_t^{[m]}$), by sampling $\bar{x}_t^{[m]} \sim p(x_t \mid x_{t-1}^{[m]}, u_t)$.

We assume access to a physics engine, f_θ, such that: $x_t = f_\theta(x_{t-1}, u_t)$. Here, θ refers to physical parameters, e.g. friction coefficient at the contacts, restitution of the contacts, and the mass of the object. The physics engine is deterministic, and therefore cannot directly be used instead of the probabilistic motion model. Therefore, we approximate sampling from the motion model $\bar{x}_t^{[m]} \sim p(x_t \mid x_{t-1}^{[m]}, u_t)$, by first sampling θ from a distribution representing our uncertainty about the physical parameters of the objects, $\theta_t^{[m]} \sim \mathcal{N}(\mu_\theta, \sigma_\theta^2)$, and then running the physics engine with the sampled parameter: $\bar{x}_t^{[m]} = f_{\theta_t^{[m]}}(x_{t-1}^{[m]}, u_t) + \epsilon$, with the addition of small Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma_f^2)$.

4.2 Observation Update

During this second stage of particle filtering, for each intermediate particle $\bar{x}_t^{[m]}$, we calculate an *importance factor* $w_t^{[m]}$ using the observation, $w_t^{[m]} = p(z_t \mid \bar{x}_t^{[m]})$. After the importance factors for all the particles are computed, they are used to *re-sample* the new set of particles $\mathcal{X}_t$, completing the particle filter update.

To compute $p(z_t \mid \bar{x}_t^{[m]})$, we propose to use the vision-only pose estimation system (e.g., DOPE) to predict the pose of the object according to z_t, and then to use the distance of $\bar{x}_t^{[m]}$ to this predicted pose to compute a probability value, $p(z_t \mid \bar{x}_t^{[m]}) \simeq \mathcal{N}(\bar{x}_t^{[m]}; DOPE(z_t), \sigma_{DOPE}^2)$, where $DOPE(z_t)$ is the pose predicted by DOPE, and the parameter σ_{DOPE}^2 represents the variance of DOPE errors for the object. This can be estimated beforehand for an object by collecting DOPE estimates for the object and comparing it to a ground truth pose.

4.3 Using Visibility Information

The observation model above assumes that a DOPE detection, $DOPE(z_t)$, is always available. However, for a given camera image z_t, DOPE may not output a pose estimate. At such time points, one option is to skip the observation update. This is what we do in the basic PBPF algorithm.

However, the lack of detection of the object is not a lack of information. The object is probably partially or fully occluded. Therefore, we also implemented an extension of our algorithm: *PBPF with Visibility* (PBPF-V), which is currently implemented only for single-object manipulation.

For this method, we first compute a *visibility score*, $v_t^{[m]}$, for each particle. To compute the visibility score, in the physics engine, we shoot R rays from the camera towards R points uniformly sampled on the manipulated object. Some of these rays hit the object, but some others may hit other occluding objects or robot links before hitting the manipulated object. We define the visibility of a particle as the ratio of rays hitting the particle to the number of all rays: $v_t^{[m]} = \frac{\text{number of rays hitting the particle}}{R}$.

A visibility score of 1 implies perfect visibility of the object by the camera, whereas a score of 0 implies the particle is completely occluded. Then, we can relate the importance factor of a particle, $w_t^{[m]}$, to its visibility $v_t^{[m]}$ based on whether a DOPE estimate is available or not at time t. The particular method to relate these values depends on a model of DOPE behaviour under occlusion. Our experiments with DOPE did **not** suggest that DOPE gradually becomes worse with more occlusion. Instead, after a certain amount of occlusion, DOPE usually loses the object completely. Therefore, instead of relating $w_t^{[m]}$ and $v_t^{[m]}$ (inversely) proportionally, we implemented a threshold-based model: If DOPE is not able to detect the object at time t, then we assign a fixed high importance factor value (W_H) to particles that have low visibility scores (i.e. $v_t^{[m]} < V_H$, where V_H is a fixed visibility threshold), and a fixed low importance factor value (W_L) to particles that have high visibility scores (i.e. $v_t^{[m]} > V_H$). Conversely, if a DOPE estimate is available at time t, then we compute the importance factor values according to the previous Section 4.2, but penalize the particles with low visibility ($v_t^{[m]} < V_L$, where V_L is a fixed low visibility threshold) by scaling their importance factor by $\alpha_W \in [0, 1]$.

5 Constant-Velocity Particle Filtering (CVPF)

Our first baseline method was presented in Sec. 3, which estimates the object's pose from a new image frame at every timestep. Our second baseline method, presented here, is a particle filter, similar to PBPF presented in Sec. 4, the only difference being in the *motion update* stage. While PBPF uses a physics engine in its motion model, here, instead we use a computationally cheap motion model, which simply assumes that the object moves with a constant velocity. In other words, during the motion update, we update a particle at time step t using the estimated pose of the object from the previous time steps $t-1$ and $t-2$. We calculate the "difference" between the estimated pose of the object at the previous time-steps: $\mathrm{d}x = \tilde{x}_{t-1} - \tilde{x}_{t-2}$ and then, we assume the object keeps moving with the same velocity, and use the motion update rule: $\bar{x}_t^{[m]} = x_{t-1}^{[m]} + \mathrm{d}x + \epsilon$ where ϵ is an extra noise term, similar to the parameter in 4.1.

There are no differences between PBPF and CVPF handling the pose information from DOPE.

6 Experiments And Results

We evaluated and compared the performance of our methods in eight different non-prehensile manipulation scenes. (All scenes can be seen in the attached video.) We used three objects from the YCB dataset [8], the CheezIt box, the Gelatin Jell-o and the Campbell Soup, a cylinder. DOPE had good performance of these objects when they were visible.

Cheezit Scene 1 (shown in Fig. 1-first row) and **Soup Scene 1** (shown in Fig. 2-fourth row): The robot pushes the object among the clutter, where the cluttering object obstructs the view of the camera. The pose of the obstructing object is fixed and known.

Cheezit Scene 2 (shown in Fig. 2-first row) and **Soup Scene 2** (shown in Fig. 2-fifth row): The robot pushes the object on a clear table. The hand can obstruct the camera view during pushing.

Cheezit Scene 3 (shown in Fig. 2-second row) and **Soup Scene 3** (shown in Fig. 2-sixth row): The robot pushes the object on a clear table, and the camera has a clear view of the object.

Cheezit Scene 4 (shown in Fig. 2-third row): The robot tilts the object up and then brings it down. The camera has a clear view. The pose of the supporting object is fixed and known.

Scene 5 (shown in Fig. 2-seventh row): The robot pushes multiple objects. These objects occlude each other during the push. We track all objects' poses simultaneously.

In Scenes 1, 2, and 3, obstacles in the clutter do not physically interact with the robot and the tracked object.

We have implemented our methods as below[1].

PBPF: The physics-based particle filtering method (Sec.4). (a) *Motion model parameters.* μ_θ and σ_θ: Mean friction coefficient of 0.1 and standard deviation of 0.3, with minimum capped at 0.001. Mean restitution of 0.9 and standard deviation of 0.2. The mean mass of 0.38 kg and the standard deviation of 0.5 with a minimum cap of 0.05 kg. σ_f: For position 0.005 m, for rotation 0.05 rad. As the physics model, f_θ, we used the Pybullet physics engine [9]. A Pybullet environment was initialized for each particle. The Pybullet environments for particles were parallelized over the 8 (16 virtual) CPU cores (11th Gen Intel(R) Core(TM) i9-11900@2.50GHz) of the computer. (b) *Observation model parameters.* σ_{DOPE}: For position 0.02 m and rotation 0.09 rad. (c) *Update frequency.* $\Delta t = 0.16s$. (d) *Number of particles.* $M = 70$. (e) *Initialization.* We use a Gaussian distribution to initialize 70 particles at $t = 0$. The mean pose is estimated by DOPE at $t = 0$. The standard dev. for initialization is 0.16 m and 0.43 rad.

[1] Experiments were performed on CPU: 11th Gen Intel(R) Core(TM) i9-11900@2.50GHz; GPU: NVIDIA GeForce RTX 3090; RAM: 128580 Mb

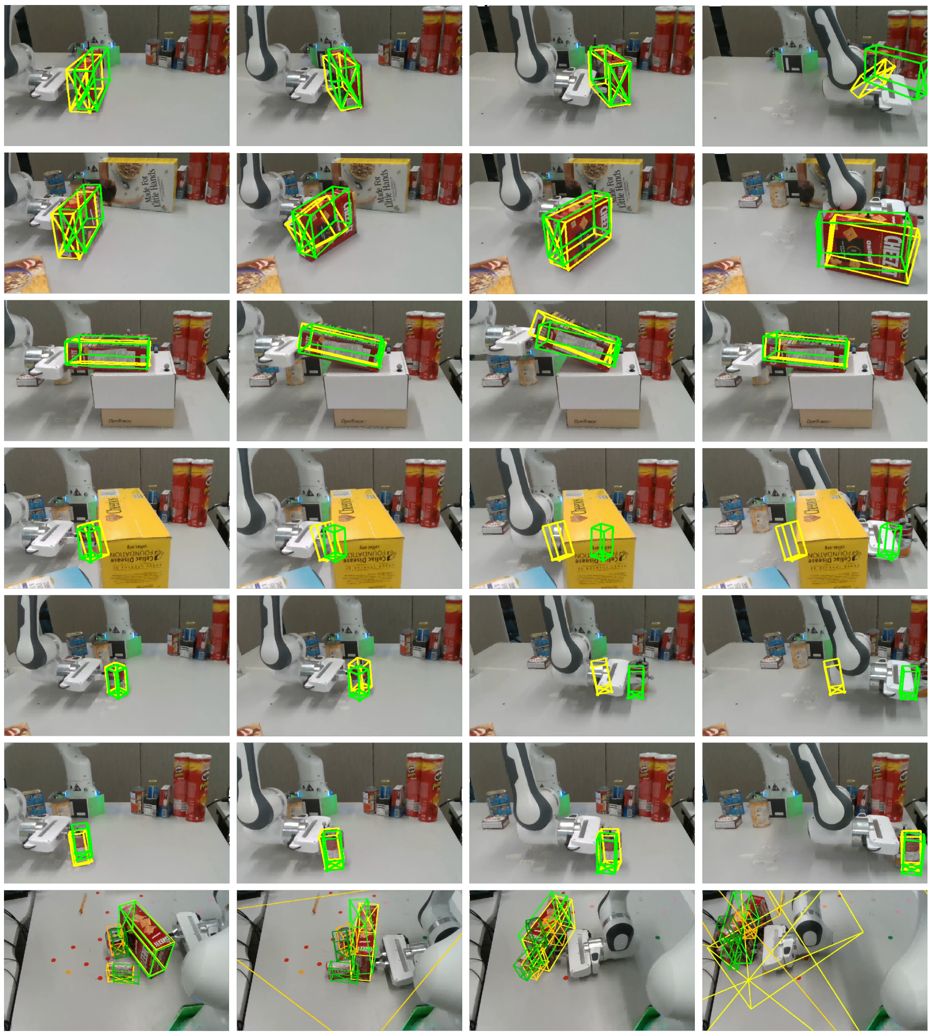

Fig. 2. Experimental Scenes. The images are from the tracking camera. The estimated object pose of our method (PBPF) is shown as a green wireframe box, while the estimated object pose from DOPE is shown as a yellow wireframe box.

PBPF-V: An extension of PBPF which also considers visibility (as described in Sec. 4.3). All parameters are the same with PBPF except that the number of particles $M = 150$ in Scene 5. Additional visibility parameters: For Cheezit and Gelatin, $W_H = 0.75$, $W_L = 0.25$, $V_H = 0.95$, $V_L = 0.5$, $\alpha_W = 0.33$. For Soup, $W_H = 0.6$, $W_L = 0.45$, $V_H = 0.8$, $V_L = 0.9$, $\alpha_W = 0.33$.

CVPF: The constant-velocity particle filtering method. All parameters were the same with PBPF, except the number of particles, M. Since CVPF is computationally cheaper, it could handle more particles $M = 150$.

Ground truth: We used a marker-based OptiTrack system to record "ground truth" poses, x_t^*, during manipulation.

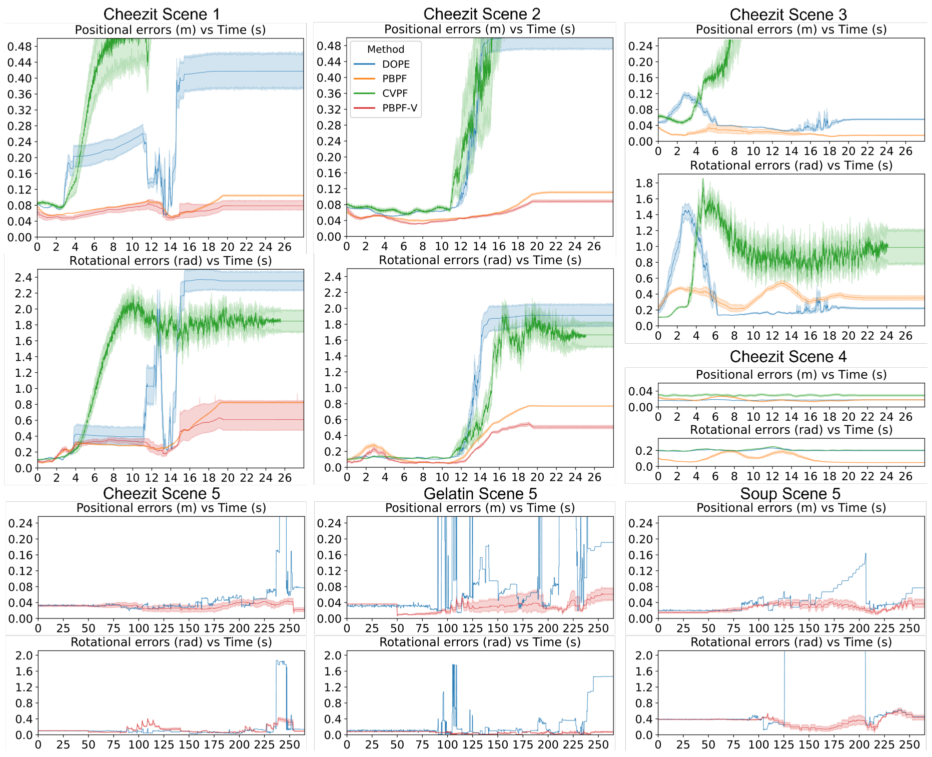

Fig. 3. Positional and rotational errors of different methods in four different scenes with the Cheezit object and the fifth scene shows the tracking of three different target objects, each with its own positional and rotational errors in different methods. Plots show mean values with the shadows indicating the 95% Confidence Interval. In all plots, the horizontal axes show time, and the vertical axes show the positional/rotational error. Similar plots for all scenes and all objects can be seen in our repository.

6.1 Experimental Procedure

In Scenes 1-4, we repeated 10 robot runs, and in Scene 5, we repeated 1 robot run (i.e. a total of 71 real robot executions). During each run, we recorded the robot controls, u_t, and the camera images, z_t. Since particle filtering is a sampling-based method, it can generate different results with the same input. For statistical accuracy, we evaluated each method 10 times on the data from each of the 10 runs in Scenes 1-4, giving 100 evaluations of each method for Scenes 1-4. In Scene 5, we evaluated each method 20 times on the data from the 1 real robot execution.

Table 1. Mean and standard deviation of the errors of different methods

	Scene 1				**Scene 2**				**Scene 3**				**Scene 4**			
	pos.err(m)		rot.err(rad)		pos.err(m)		rot.err(rad)		pos.err(m)		rot.err(rad)		pos.err(m)		rot.err(rad)	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Cheezit																
PBPF	0.081	0.020	0.484	0.272	0.070	0.030	0.426	0.309	**0.019**	0.024	**0.367**	0.187	0.018	0.008	**0.092**	0.067
PBPF-V	**0.063**	0.020	**0.396**	0.295	**0.058**	0.021	**0.294**	0.204	-	-	-	-	-	-	-	-
DOPE	0.289	0.218	1.345	1.168	0.293	0.248	1.046	1.027	0.052	0.033	0.375	0.447	**0.017**	0.002	0.201	0.013
CVPF	2.359	7.421	1.472	0.951	0.407	0.678	0.878	0.952	25.14	152.8	0.879	0.952	0.029	0.012	0.207	0.022
Soup																
PBPF	**0.043**	0.023	1.133	0.364	0.049	0.032	0.851	0.249	**0.026**	0.021	**0.405**	0.455	-	-	-	-
PBPF-V	0.045	0.027	**1.077**	0.395	**0.046**	0.028	**0.724**	0.262	-	-	-	-	-	-	-	-
DOPE	0.215	0.117	1.174	0.372	0.206	0.112	0.776	0.222	0.071	0.140	0.438	0.482	-	-	-	-
CVPF	0.298	0.153	1.884	0.794	0.240	0.126	1.702	0.845	6.528	52.09	1.174	0.982	-	-	-	-

At each time step of each evaluation, we computed the positional and rotational errors of the mean estimate, against the ground truth. When computing errors for DOPE, if DOPE did not output any pose at a certain time step (e.g., because of occlusions), we used the latest pose reported before that time step.

6.2 Results

We present the overall (averaged over all runs and all time-steps) positional and rotational errors from Scene 1-4 in Table. 1, and from Scene 5 in Table. 2. Compared to DOPE and CVPF, PBPF performs significantly better in Scenes 1, 2, and 3. In Scene 4, where DOPE has a good view and overall good detection, PBPF and DOPE perform similarly. CVPF performs worse out of all the methods. In Scene 5, we only ran the PBPF-V algorithm used for comparison with DOPE. PBPF-V performs significantly better than the baseline method.

When there is occlusion (Scenes 1 and 2), PBPF-V performs better than basic PBPF for the Cheezit object. For the Soup object, the difference between PBPF and PBPF-V is not significant. This is probably due to the Soup being a much smaller object, for which DOPE's behaviour under occlusion is much more difficult to model accurately.

We also present the positional and rotational errors of each method over the course of manipulation in Scenes 1-4 for Cheezit in Fig. 3. (Results for Soup are similar, and are in our repository.) We can see that PBPF and PBPF-V produce better and more stable estimates of the object position. In Fig. 3, we also show the results for each object in Scene 5. PBPF-V significantly outperforms DOPE.

It is worth noting that even with a good field of view, DOPE can be noisy, because it uses a single visual snapshot, which becomes unstable when the light, angle, and other factors change. This can lead to physically unrealistic poses, e.g.

Table 2. Mean and standard deviation of the errors when tracking multi-object (Scene 5)

	Cheezit				**Gelatin**				**Soup**			
	pos.err(m)		rot.err(rad)		pos.err(m)		rot.err(rad)		pos.err(m)		rot.err(rad)	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
PBPF-V	**0.031**	0.011	**0.126**	0.087	**0.031**	0.023	**0.056**	0.030	**0.027**	0.013	**0.346**	0.139
DOPE	0.058	0.108	0.158	0.337	0.179	0.665	0.272	0.410	0.045	0.029	1.193	1.192

target object can be inside the robot arm, or two target objects can penetrate. In contrast, our method (PBPF) outputs physically feasible poses, effectively avoiding the unrealistic scenarios above.

References

1. Correll, N., Bekris, K.E., Berenson, D., et al., “Analysis and observations from the first amazon picking challenge,” T-ASE, 2016.
2. Tremblay, J., To, T., Sundaralingam, B., Xiang, Y., et al., “Deep object pose estimation for semantic robotic grasping of household objects,” arXiv preprint, 2018.
3. Collet, A., Martinez, M. and Srinivasa, S.S., “The moped framework: Object recognition and pose estimation for manipulation,” IJRR, 2011.
4. Ruggiero, F., Lippiello, V. and Siciliano, B., 2018. “Nonprehensile dynamic manipulation: A survey,” RA-L, 2018.
5. Papallas, R., Cohn, A.G. and Dogar, M.R., 2020. “Online replanning with human-in-the-loop for non-prehensile manipulation in clutter—a trajectory optimization based approach”. RA-L, 2020.
6. Choi, C. and Christensen, H.I., “Probabilistic object tracking using a range camera,” in IROS, 2013.
7. Schmidt, T., Hertkorn, K., Newcombe, R., Marton, Z., Suppa, M. and Fox, D., 2015, May. “Depth-based tracking with physical constraints for robot manipulation”. ICRA, 2015.
8. Calli, B., Walsman, A., Singh, A., et al., “Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols,” arXiv preprint, 2015.
9. Coumans, E. and Bai, Y., 2016. “Pybullet, a python module for physics simulation for games, robotics and machine learning”.
10. Deng, X., Mousavian, A., Xiang, Y., et al., “Poserbpf: A rao–blackwellized particle filter for 6-d object pose tracking,” T-RO, 2021.
11. Mitash, C., Boularias, A. and Bekris, K., 2022. “Physics-based scene-level reasoning for object pose estimation in clutter”. IJRR, 2022.
12. Pfanne, M., Chalon, M., Stulp, F. and Albu-Schäffer, A., 2018. “Fusing joint measurements and visual features for in-hand object pose estimation”. RA-L, 2018.
13. Koval, M.C., Dogar, M.R., Pollard, N.S. and Srinivasa, S.S., 2013, November. “Pose estimation for contact manipulation with manifold particle filters”. IROS, 2013.
14. Wen, B., Mitash, C., Ren, B. and Bekris, K.E., “SE (3)-tracknet: Data-driven 6d pose tracking by calibrating image residuals in synthetic domains”. IROS, 2020.
15. Pauwels, K. and Kragic, D., 2015, September. “Simtrack: A simulation-based framework for scalable real-time object pose detection and tracking,” in IROS, 2015.
16. Pauwels, K., Ivan, V., Ros, E., et al. “Real-time object pose recognition and tracking with an imprecisely calibrated moving rgb-d camera,” in IROS, 2014.
17. Zhong, S., Fazeli, N. and Berenson, D., 2022. “Soft tracking using contacts for cluttered objects to perform blind object retrieval”. RA-L, 2022.
18. Papallas, R. and Dogar, M.R., 2022, October. “To ask for help or not to ask: A predictive approach to human-in-the-loop motion planning for robot manipulation tasks”. in IROS, 2022.
19. Ge, R. and Loianno, G., “Vipose: Real-time visual-inertial 6d object pose tracking,” in IROS, 2021.
20. Lu, S., Wang, R., Miao, Y., et al. “Online object model reconstruction and reuse for lifelong improvement of robot manipulation,” in ICRA, 2022.

21. Jongeneel, M., Bernardino, A., van de Wouw, N. and Saccon, A., 2022, June. “Model-Based 6D Visual Object Tracking with Impact Collision Models”. In ACC, 2022.
22. Thrun, S., “Probabilistic robotics”. Communications of the ACM, 2002.