Physics-Based Dexterous Manipulations with Estimated Hand Poses and Residual Reinforcement Learning

Guillermo Garcia-Hernando[1,2], Edward Johns[1] and Tae-Kyun Kim[1,3]

Abstract— **Dexterous manipulation of objects in virtual environments with our bare hands, by using only a depth sensor and a state-of-the-art 3D hand pose estimator (HPE), is challenging. While virtual environments are ruled by physics, e.g. object weights and surface frictions, the absence of force feedback makes the task challenging, as even slight inaccuracies on finger tips or contact points from HPE may make the interactions fail. Prior arts simply generate contact forces in the direction of the fingers' closures, when finger joints penetrate virtual objects. Although useful for simple grasping scenarios, they cannot be applied to dexterous manipulations such as in-hand manipulation. Existing reinforcement learning (RL) and imitation learning (IL) approaches train agents that learn skills by using task-specific rewards, without considering any online user input. In this work, we propose to learn a model that maps noisy input hand poses to target virtual poses, which introduces the needed contacts to accomplish the tasks on a physics simulator. The agent is trained in a residual setting by using a model-free hybrid RL+IL approach. A 3D hand pose estimation reward is introduced leading to an improvement on HPE accuracy when the physics-guided corrected target poses are remapped to the input space. As the model corrects HPE errors by applying minor but crucial joint displacements for contacts, this helps to keep the generated motion visually close to the user input. Since HPE sequences performing successful virtual interactions do not exist, a data generation scheme to train and evaluate the system is proposed. We test our framework in two applications that use hand pose estimates for dexterous manipulations: hand-object interactions in VR and hand-object motion reconstruction in-the-wild. Experiments show that the proposed method outperforms various RL/IL baselines and the simple prior art of enforcing hand closure, both in task success and hand pose accuracy.**

I. INTRODUCTION

Capturing and transferring human hand motion to anthropomorphic hand models in physics-embedded environments, is the cornerstone of applications that require realistic interactions in VR/AR. To capture hand motion in such applications, most previous works resort to expensive and intrusive motion capture (mocap) systems, such as gloves [1], exoskeletons and controllers [2]. In this work, we aim to avoid such systems and explore a solution that allows us to perform dexterous manipulation actions by only using an estimate of the human hand pose.

Hand pose estimators (HPEs) typically produce 3D locations of keypoints of a human hand model. Given the difference between the human hand and the hand model, the design of a function mapping an input hand pose to the model's parameters is needed, a process known as inverse kinematics or motion/pose retargeting. Designing a function that produces a visually similar output is relatively straightforward, and hand-engineered [4], [5], data-driven [6], [7] or hybrid [5], [8] solutions are available. However, when interacting with the simulated physical environment, visual resemblance between input and target is not enough, given that one needs to consider both contact physics between the hand and object and input noise coming from the hand pose estimator as shown in Fig. 1. Commercial solutions [4], [9] circumvent these problems simply by ignoring physics laws and 'attracting' the hand towards the object.

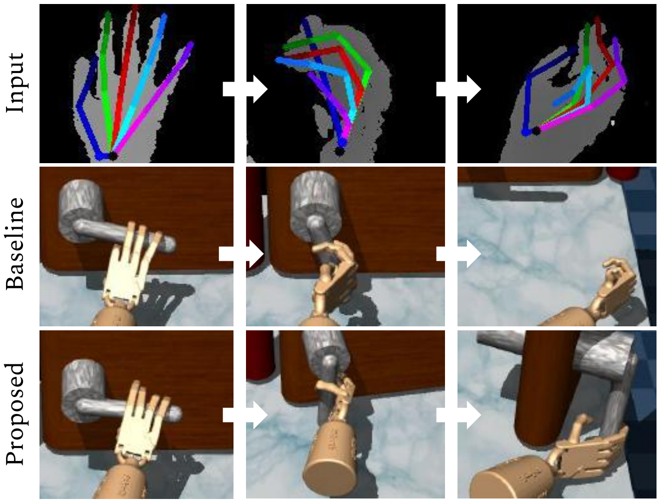

Fig. 1. Mapping an estimated hand pose from a user, to a physically accurate virtual hand model is challenging. Simple pose retargeting functions fail due to the domain gap, contact physics, pose prediction errors and noise. Our method observes both the imperfect mapped hand pose from the user input, middle row, and the state of the simulation and produces a small residual correction that completes the task. To train our system, we generate input hand poses, top row, with a new data generation scheme that builds upon a mocap dataset [1] and a large public hand pose dataset [3]. Note that the depth camera is pointing to the human hand from the ground.

Other approaches model the underlying contact physics by establishing relationships between the virtual penetration of the hand on the object [10]. Such solutions, despite being effective for some simple grasping actions, do not produce physically realistic motion in the target domain. Also, the inferred contact force will depend directly on the noisy pose estimate, making it difficult to apply the precise forces and subtle movements required in some dexterous tasks.

Related to our work, [11], [12] track and reconstruct 3D hand-object interactions using simple physics constraints such as contact and mesh penetration. In contrast, we generate complete physics-aware sequences using a physics simulator, which can actually succeed in the task of interest. Related to us, and aiming to generate physically plausible sequences from vision, [13], [14] use RL for full body

[1]Imperial College London, United Kingdom. [2]Niantic, Inc., United Kingdom. [3]KAIST, South Korea. This work was part of Imperial College London-Samsung Research project, supported by Samsung Electronics.

poses. Different to [13], which aims to teach an agent to autonomously perform by observing a single reconstructed and filtered video, our work aims to correct noisy user hand poses ‘as they come’, and *assist* the user in a similar setting to shared autonomy [15]. In contrast to [14], which aims to estimate the ego-pose of the humanoid by indirectly observing from their character point of view, we directly observe user’s hand motion and assist in achieving the task while generating virtual poses similar to the visual input.

We propose a system, illustrated in Fig. 2, that observes an imperfect user input and refines it in order to accomplish the manipulation task. We define the user input, Section III-A, as an estimated hand pose mapped by an inverse kinematics or pose retargeting function. To achieve this, we introduce a residual agent that acts on top of the user input in Section III-B. We assume that the user input is similar to the optimal action –modern HPEs present average joint errors in the range of 7 to 15 mm [16]– and only require a correcting stage to produce the correct kinematics. In order to automatically learn this correction without making any assumptions on the underlying contact physics, we train the residual agent using reinforcement learning (RL) in a model-free setting [17], [18] within an accurate physics simulator [19]. To avoid unnatural motion typically present under RL framework [1], our system builds upon recent work in adversarial imitation learning (IL) [20], [21], that uses a discriminator to encourage the policy to produce actions similar to trajectories from a dataset captured using a mocap glove [1]. Unlike prior arts [4], [9], [10], our method enables dexterous manipulations e.g. in-hand pen manipulation or picking a coin. The proposed residual agent is also learned by the 3D hand pose estimation reward, improving HPE accuracy when the physics-guided corrected target poses are re-mapped to input space. These objectives are presented in Section III-B.1.

To train such a framework, we need continuous intended action sequences of noisy estimated hand poses, as well as some successful manipulation actions obtained by mocap data. It is difficult to collect such HPE sequences in an online fashion, because users tend to stop their motions in the middle of the tasks when they fail. We first explore generating noisy input sequences by adding random noise to the ground-truth mocap data. To circumvent the gap between the synthetic noise and the real structured noise coming from HPE, we propose, in Section III-C, a data generation approach which, given a dataset of successful manipulation sequences in the virtual space [1], finds a ground-truth hand pose and depth image that is most likely to have generated such action, by querying a public large scale hand pose dataset [3]. Using this pipeline we conduct experiments on two potential applications of our framework. The first one, *Experiment A*, appears in Section IV-A and it studies a typical VR scenario where the user interacts with the environment with their bare hands in mid-air and a hand pose estimator. In the second one, *Experiment B* in Section IV-B, we aim to reconstruct in a physics simulator hand-object RGBD sequences captured in-the-wild with the use estimated hand poses and initial object pose estimates. In various experiments, our proposed method outperforms RL/IL baselines, and some relevant arts.

II. RELATED WORK

3D hand pose estimation consists of estimating the 3D locations of hand keypoints given an image. A main part of the success in the field comes from the use of depth sensors [6], [22], [23] and deep learning [24], [25], [16], while recent successful approaches exploit single RGB images as input [26], [27]. Note that most current hand pose estimators only output 3D joint locations than angles, making the mapping between locations and angles not trivial; however there is some promising work on estimating 3D hand meshes that could make this problem easier [28], [29].

Vision-based teleoperation. Traditionally, teleoperation has been limited to mapping the human hand to the (physical or virtual) robot hand by using contact devices such as tracking sensors [30], exoskeletons [31] and gloves [32]. Some vision-based approaches exist [33], [34], [35], [5], [8], [36] but are limited to simple grasping actions. [5] proposes a retargeting method between depth images and a robotic hand model, however the mapping function is purely based on hand appearance ignoring objects. [8] combines inverse kinematics with a PSO function that encourages contact between object and hand surfaces. We share with [8] the aim of achieving realistic interactions, but simply forcing contact is not enough for dexterous actions such as in-hand manipulation. [36] introduces a HPE tailored to a robot hand model. Given that our framework is HPE-agnostic, both works are complementary and could to produce a solid system if combined. In the VR and graphics community, perhaps the simplest approach for tackling such problems, and as adopted by commercial products such as Leap Motion [4] or Hololens [9], is to recognize the ongoing hand gesture, e.g. swipe or pinch, and then trigger a prerecorded output [37], [38], [39]. However, such approaches produce artificial motion that often deviates significantly from the user input. Similarly, the interaction engine by Leap Motion [4] recognizes the gesture and ‘attracts’ the object to the hand producing an artificial ‘sticking’ effect. Our method corrects the user input slightly, but only enough to achieve the task, and importantly it respects the laws of physics. Other works use a priori information about the hand and the scene, by synthesizing a grasp from a predefined database [40], [41], [42], [43], [44], limited to a specific set of objects and interactions, and very sensitive to uncertainty about the environment. Some works attempt to model the contact physics [45], [46], [47], [48], [49], [10] to infer contact forces between the hand and objects, by measuring, for example, the penetration of the user hand into the object mesh. The main problem of such approaches is that the computed contact force relies on high-precision hand pose estimation, and the method tends to apply forces that do not necessarily transfer to the real world without unexpected consequences.

Physics-based pose estimation. [11] uses a physics simulator within an optimization framework to refine hand poses, following earlier generative and discriminative model fitting work [50], [51], [52], [53]. [12] presents an end-to-end deep learning model that exploits a contact loss and mesh penetration penalty similar to [54], [51], [11], [55], for plausible hand-object mesh reconstruction. These estimators are subject to simple physical constraints such as contact and mesh penetration and deal with single-shot images. In [14], physically-valid body poses are estimated and forecasted from egocentric videos using RL. Their aim is to estimate the ego-pose of the humanoid by indirectly observing from their character point of view using similar rewards as [13], discussed below.

Motion retargeting and reinforcement learning. Our problem shares similarities with full body motion retargeting [56], particularly with methods that consider accurate physics on the target space and train control policies using RL [57], [58], [59], [60], [13]. [58], [59], [60] propose an RL approach to learn skills from a reference mocap motion. [13] extends such work to deal with reference motion from a body pose estimation step that is cleaned and post-processed to mimic the motions, as in [58]. The main difference of our work is that we perform online predictions given a noisy user input instead of learning to mimic a skill in an offline fashion. For this reason, we embrace the noisy nature of our problem and propose the residual learning guided by the hand pose estimation reward and the noisy data generation scheme.

Robot dexterous manipulation and reinforcement learning. For attempting to learn robotic manipulation skills without user input, and using both RL and IL, we highlight three recent works [21], [1], [18]. We share with [21] a similar adversarial hybrid loss, however our model has significantly more degrees of freedom.We build upon [1]'s simulation framework, using their dataset of glove demonstrations, and extend the environments to deal with vision-based hand pose estimation. We share with [18] the ambition of learning physically accurate dexterous hand manipulations, but more in physics embedded VR space using user's hand via state-of-the-art hand pose estimator.

Residual policy learning. We discuss two recent papers proposing a similar residual policy idea [61], [62]. We share with these works the residual nature of our policy and the idea that improving an action, instead of learning from scratch, significantly helps the exploration problem of RL and tends to produce more robust policies. The main difference from our work is that our residual action works on top of a user input instead of a pre-trained policy, i.e. our policy observes the action taken by the user and the world and then acts accordingly, instead of just observing the state of the world, which could lead to a discrepancy between the user's intention and the agent. Other differences include the nature of the problem, the complexity of the action space, the combination with adversarial IL, and a problem setting similar to shared autonomy [15].

III. PROPOSED FRAMEWORK

A. Inverse kinematics: from human hand pose to virtual pose

Given the user's estimated hand pose x_t, which consists of the 3D locations of 21 joints of a human hand [3] on a given visual representation ϕ_t at time step t, we aim to obtain a visually similar hand posture z_t in our virtual model. This requires estimating parameters a_t, defined as the actuators or *actions* of the virtual hand model which determine the target angle between hand joints with the help of PID controllers.

Inverse kinematics (IK) refers to the task of computing rotations a_t such that the virtual hand pose z_t is equivalent to the user's hand pose x_t. Note that z_t belongs to a different domain to x_t, but it can be measured by carefully placing sensors in the virtual hand model. This mapping from pose to rotations, κ, can be manually designed or automatically learned, for example with a supervised neural network when input-output pairs are available, and can be written as:

$$a_t = \kappa(x_t(\phi_t)). \tag{1}$$

For simplicity, we often refer to $\kappa(x_t(\phi_t))$, in the action space, as the *user input*, in contrast to user's estimated hand pose x_t, in the pose space. IK is inherently an ill-posed problem, since depending on how different the virtual and human models are, the target pose z_t can potentially be reached by multiple a_t's, or there may not be a solution at all. This problem becomes even more aggravated when the input x_t is noisy, which is the nature of a hand pose estimator. We describe our residual approach to deal with this imperfect input next.

B. Residual Hand Agent

We now describe how to train the residual controller, which acts upon the output of the above IK function. Due to both the imperfect mapping between the human kinematics and virtual kinematics, and the noise introduced by the hand pose estimator, we assume that the user input $\kappa(x_t(\phi_t))$ produces actions that are *close* to optimal, but not sufficiently good to succeed in the task of interest. As an additional requirement for optimal action predictions, the temporal nature of our sequences means that a small early mistake can later have a catastrophic effect due to compounding errors that propagate to subsequent simulation stages. The residual controller introduces a *residual* action f_t, which is a function of $\kappa(x_t(\phi_t))$, the current simulation state s_t and the visual representation ϕ_t, which can be either an image or extracted visual features. Those terms are combined as follows:

$$a_t = \kappa(x_t(\phi_t)) - f_t(s_t, \kappa(x_t(\phi_t)), \phi_t). \tag{2}$$

In order to not deviate from the user input significantly, we limit the residual action f to be within a certain zero-centered interval. We formulate the learning of the residual policy as a RL problem, where an agent interacts with a simulated environment by following a policy $\pi_\theta(f|s, \kappa, \phi)$ parametrized by θ, which in our case is a neural network.

The state s includes the current information tailored to every task of the simulation environment, such as the relative positions between the target object and the virtual hand model, the model's velocity, etc. At each time step t the agent observes s_t, $\kappa(x_t(\phi_t)$ and ϕ_t, samples an action f_t from π_θ, and an action a_t is applied to the environment. The environment moves to the next state s_{t+1} sampled from the environment dynamics, which we assume to be unknown. A scalar reward r_t quantifies how good or bad this transition was, and thus our goal is to find an optimal policy that maximizes the expected return, defined as $J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\theta)}\left[\sum_{t=0}^{T}\gamma^t r_t\right]$, where $p_\theta(\tau)$ is the distribution over all possible trajectories $\tau = (s_0, \kappa(x_0), \phi_0, f_0, s_1, ...)$ following the policy π_θ. The term $\sum_{t=0}^{T}\gamma^t r_t$ represents the total return of a trajectory for a horizon of T time steps and a discount factor $\gamma \in [0, 1]$. In our problem, T is variable

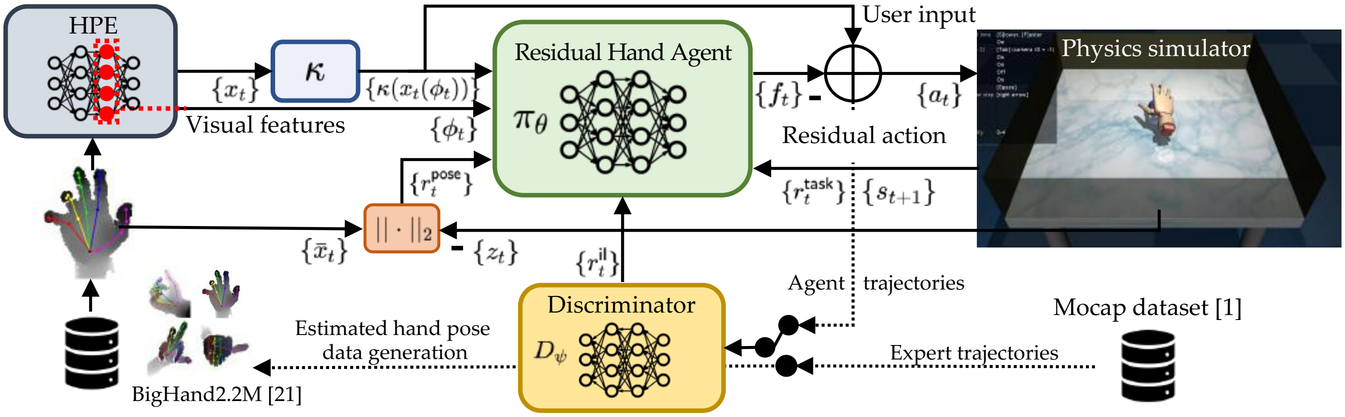

Fig. 2. **Framework training overview.** During training, the residual agent performs actions that aim to correct the user input, and receives feedback from both the simulator and a discriminator. The discriminator indicates how much the actions resemble expert human actions from a mocap dataset [1], whilst the simulator allows us to generate several samples of rich physics simulation and to measure the resemblance between input and virtual poses. To train our framework in the absence of ground-truth pairs hand poses and actions, we can generate estimated hand poses by finding images on a large hand pose dataset [3] that are likely to have generated the actions from the mocap dataset. Once we find these samples, we pass them through a hand pose estimator and an inverse kinematics or pose retargeting function to generate user input. Algorithm details can be found in the Appendix.

depending on the length of the hand pose input sequence. State and rewards details can be found in the Appendix.

To optimize θ several methods can be used, however in this work we focus on a popular policy gradient approach proximal policy optimization (PPO) [17] due to its recent success on learning dexterous policies without user input [18]. This approach optimizes J over θ to maximize the return. The gradient of the expected return $\nabla_\theta J(\theta)$ is estimated with trajectories sampled by following the policy, and learns both a policy network and a value function, which estimates the expected return when following the policy.

1) Reward function: The total reward function r_t that guides the framework learning process is defined as:

$$r_t = \omega^{\text{task}} r_t^{\text{task}} + \omega^{\text{il}} r_t^{\text{il}} + \omega^{\text{pose}} r_t^{\text{pose}}, \tag{3}$$

where ω^{task}, ω^{il} and ω^{pose} are weighting factors.

a) Task-oriented reward: r_t^{task}: it is tailored for each environment and guides the policy towards desirable behaviours in terms of task accomplishment, with short-term rewards such as getting close to the object of interest, and long-term rewards such as opening the door (see Appendix).

b) Imitation learning reward: r_t^{il}: Policies learned with only RL tend to produce unnatural behavior: they are effective to accomplish the task of interest, but produce actions that a human would never do [1]. To encourage action sequences that more closely resemble expert data, we add the following adversarial IL reward function similar to [21]:

$$r_t^{\text{il}} = (1 - \lambda)\log(1 - D_\psi(s_t, a_t)), \tag{4}$$

where D_ψ is a score quantifying how good an action is, given by a discriminator with parameters ψ. To include this objective in our framework, we use a min-max objective [20]:

$$\min_\theta \max_\psi \mathbb{E}_{\pi_E}[\log D_\psi(s,a)] + \mathbb{E}_{\pi_\theta}[\log(1 - D_\psi(s,a))], \tag{5}$$

where π_E denotes an expert policy generated from demonstration trajectories. This objective encourages the policy π_θ to produce actions f_t that correct the user input $\kappa(x_t(\phi_t))$, generating pairs of (s_t, a_t) that are similar to those of an expert. In our framework, we obtain $\mathcal{D} = (s_i, a_i)_{i=1\dots N}$ from [1], which used a data glove and a tracking system [32] to capture noise-free sequences.

c) 3D hand pose estimation reward: r^{pose}: The reward terms introduced above can lead to virtual poses z_t, that diverge from the pose depicted on the user input image, particularly if the hand pose estimator fails due to object occlusion. If we have access to annotated ground-truth hand poses $\bar{x}_t$ during training, we can introduce an additional reward that encourages the policy network to produce actions that visually resemble the user pose and is defined as:

$$r_t^{\text{pose}} = -\sum_j^{21} ||z_t^j - \bar{x}_t^j||_2, \tag{6}$$

where z_t^j and $\bar{x}_t^j$ denote the 3D position of the j-th joint of the human and model respectively.

C. Data generation scheme

If we examine Eq. 2 we observe that, to train our residual policy, we need a dataset of estimated hand poses $\{x_t\}$ depicting natural hand motion that would produce a successful interaction if the system was perfect. We could think of recording hand pose sequences by asking users to perform the action 'as if it was successful', but given the temporal dependency of the problem we would be acquiring data somewhat different from the true distribution.

Our idea consists of using a mocap dataset which contains successful sequences of state-action pairs and find hand images that could have produced these actions by querying a 3D hand pose dataset. For this approach to work, a dense and exhaustive 3D hand pose dataset in terms of articulations and relative camera-hand viewpoints is needed. We use BigHand2.2M [3] as hand pose dataset and the dataset introduced in Rajeswaran et al. [1] as mocap dataset. We first measure the virtual poses $\{z_t\}$ generated by the actions by placing virtual sensors and a virtual camera. Given the sequences of virtual poses, we retrieve the closest ground-truth poses in a 3D hand pose dataset. We tried different representations and query functions for retrieval, but got the best results by retrieving similar viewpoints and later refining by the distance of aligned and palm-normalised joint coordinates. Once the matches are found, we retrieve their associated image and compute estimated hand poses by passing the images through a 3D hand pose estimator.

IV. EXPERIMENTS[1]

A. Performing dexterous manipulations in a virtual space with estimated hand poses in mid-air

In this experiment we evaluate our framework when we have access to a glove-recorded mocap dataset [1] with successful expert trajectories and we use our data generation scheme. As HPE we use [63] and to train and retrieve images with particular poses we use BigHand2.2M dataset [3], which was designed to densely capture articulation and viewpoint spaces in mid-air and in an object-free setup. Because of the absence of object occlusions in BigHand2.2M, we drop r^{pose} and do not feed visual features to the policy network. We first evaluate our framework in a controlled setting where we add synthetic noise to expert demonstrations and then we evaluate it with real structured hand pose estimation noise.

Hand model: We use the ADROIT anthropomorphic platform [1], consisting of 24 degrees-of-freedom (DoF) joint angle rotations of Shadow dexterous hand, plus 6 DoF defining the 3D position and orientation of the hand.

Simulator and tasks: We use the MuJoCo physics simulator [19] and the four dexterous manipulation scenarios defined in [1]: door opening, in-hand manipulation, tool use and object relocation. In 'door opening' the task is to undo the latch and swing the door open. In 'in-hand manipulation' the task is reposition a blue pen to match the orientation of a target pose (green pen). 'Tool use': the task consists of picking up a hammer and drive the nail into a board. 'Object relocation' aims to move a blue ball to a green target location. Each task is considered successful if the target is achieved with a certain tolerance. There are about 24 mocap trajectories per task and we split them in equal training-test sets.

Policy network: π is a (64, 64) MLP and the residual policy is limited to 20% of the action space. The action is modeled as Gaussian distribution with a state-dependent mean and a fixed diagonal covariance matrix. We use the same architecture for value function and discriminator.

Baselines. In this experiment we evaluate the following:

Inverse kinematics (IK): The action applied is based solely on user's input and we specify below its nature.

Reinforcement learning (RL): The agent observes both the user input and the state in a non-residual way [17] without access to demonstrations. Two versions: 'RL - no user' with only task reward and 'RL + user reward' with additional reward term encouraging following the user input.

Imitation learning (IL): The agent observes both the user input and the state and it has access to demonstrations during the adversarial learning process based on GAIL [20].

Hybrid learning: We combine the above baselines in a similar way to our proposed algorithm without residual.

Implementation details, states and rewards definitions, and training parameters can be found in the Appendix.

1) Overcoming random noise on demonstrations: The aim of this experiment is to verify whether our framework can deal with noisy observations and produce useful residual actions. In this scenario we have total control on the amount and nature of the noise allowing us to dissect the results. In this experiment the user inputs are the expert successful actions recorded using a mocap glove from [1] on the 'opening door' environment, thus we can assume they are free of noise. We synthesize noise by adding a zero-mean Gaussian noise with standard deviation σ radians to each actuator, on top of the user input in both training and test trajectories. Note that errors in a single actuator propagate through the linked joints by forward kinematics.

After training a policy for a certain σ, we show its generalization to other values of noise on test sequences in Table I. We observe that our residual agent is able to recover meaningful motion up to a σ_{test} of 0.20 rad when similar magnitudes have also been observed in training. The noisy user input can succeed a significant amount of times alone provided that small changes, or changes in the right direction, may not affect the overall success.

TABLE I
OUR APPROACH FOR DIFFERENT NOISE LEVELS IN TRAINING/TEST

	σ_{test}					
σ_{train}	0.00	0.01	0.05	0.10	0.15	0.20
0.01	71.00	70.00	52.00	26.00	9.00	1.00
0.05	100.0	90.00	83.00	50.00	24.00	4.00
0.10	91.00	89.00	87.00	**87.00**	56.00	**26.00**
0.15	**100.0**	**96.00**	**92.00**	80.00	**57.00**	19.00
0.20	71.00	74.00	75.00	71.00	47.00	20.00
User input:	80.00	86.30	74.00	33.80	9.20	2.70

In Table II we show the performance of different baselines for a fixed σ value of 0.05 rad for each environment. Two results are reported: the task success on noisy sequences generated from demonstration sequences that are only seen during training, and the accuracy on an independent test set. Some of the baselines do not succeed on some environments, being consistent with the results reported by [1]. Furthermore, when training our residual policy, it converges significantly faster than other baselines (see Fig. 3 (a)). For instance, our policy converges after 3.8M and 5.2M samples for door opening and in-hand manipulation, compared to 7.9M and 13.8M for RL baseline. In our RL framework for our approach and baselines, 5M samples with network updates are generated in about 12 hours on a single core machine with a GTX 1080Ti, while RL alone baselines require only. The reason for this faster convergence is the help in exploration that the user input brings to the learning process [62]. For the last scenario, 'object relocation', none of our baselines nor our approach is able to correct the user input and degrade its performance. We hypothesize that the low result of PPO propagates to our algorithm and using other optimization could help to recover the user input [1].

To conclude this experiment we perform an ablation study to evaluate the impact of each RL and IL components of our approach. Combining both leads to accomplishing the task while keeping a motion that resembles the human experts more closely (see Fig. 3 (b) and video). In terms of task success, RL alone achieves 75.9% while IL alone 36.5%.

2) Overcoming structured hand pose estimation and mapping errors: In this experiment, we aim to verify that our algorithm can also deal with the structured noise injected

[1] Appendix can be found at the end of this document and videos in the project page: https://sites.google.com/view/dexres

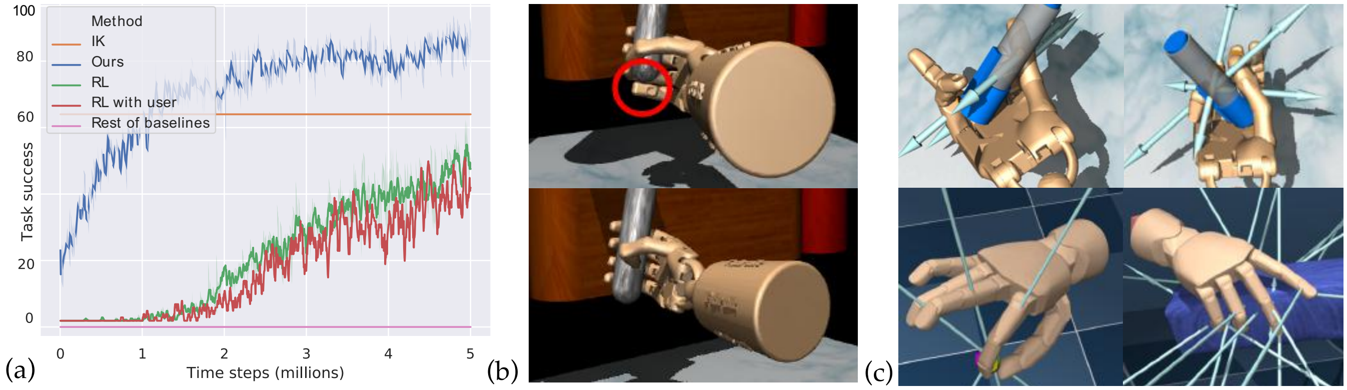

Fig. 3. Training curves on ‘opening door’ for our approach and baselines (a). (b) Qualitative ablation study on reward function (top) Our agent with only task reward r_t^{task} and (bottom) adding r_t^{il} on the same input sequence with equal weights. (c) Resulting contact forces for in-hand manipulation, ‘give coin’ and ‘pour juice’. For in-hand manipulation, approaches maximizing contact cannot accomplish the task.

TABLE II

BASELINES FOR A FIXED AMOUNT OF NOISE ON TOP OF USER INPUT.

	Door opening		Tool use		In-hand man.		Object rel.	
Method	Train	Test	Train	Test	Train	Test	Train	Test
IK	64.00	74.00	50.00	56.00	67.67	69.92	77.00	83.00
RL-no user	75.00	59.00	51.00	44.00	43.61	38.34	0.00	0.00
IL-no user	0.00	0.00	0.00	0.00	4.00	6.77	0.00	0.00
Hybrid-no res.	0.00	0.00	0.00	0.00	4.00	0.00	0.00	0.00
RL+user reward	69.92	62.40	6.01	9.02	48.12	27.81	0.00	0.00
Hybrid+user rew.	0.00	0.00	56.39	33.08	9.02	7.51	0.00	0.00
Ours	**81.33**	**83.00**	**61.00**	**58.00**	**90.97**	**87.21**	**49.62**	**16.54**

TABLE III

BASELINE ON STRUCTURED HAND POSE ERROR ON GROUND-TRUTH (GT) AND ESTIMATED HAND POSES (EST.)

	Door opening		In-hand man.	
Method (Training set)	GT	Est.	GT.	Est.
IK	49.62	27.81	0.00	20.30
RL - no user (GT)	98.49	76.69	13.53	25.56
RL - no user (Est.)	66.16	71.42	13.53	0.00
RL + user reward (GT)	0.00	0.00	**45.86**	32.33
RL + user reward (Est.)	0.00	0.00	0.00	12.03
Ours (Experiment A.1)	57.14	38.34	10.52	0.00
Ours (GT poses)	83.45	42.10	10.52	32.33
Ours (Est. poses)	**85.95**	**70.67**	20.33	**57.14**

via the hand pose estimator and the mapping function. We generate the training data using our strategy described in Section III-C. After creating the dataset, we also need to design a mapping function from the hand pose to the virtual model. Leveraging our data sampling strategy, we create pairs of data (x_t, a_t). Other settings remain the same.

Supervised IK baseline: We use these pairs to train a function $\kappa(x_t)$ in a supervised setting. Our IK network is a (64, 64) MLP trained with a regression loss. In Table III we observe that this function alone is not enough although it can achieve moderate success on the ‘door opening’ scenario when ground-truth (GT, not noisy) poses are used.

In Table III the results of both the best performing baselines in the previous experiment and our algorithm are depicted. We show results for both ground-truth poses and estimated (Est.) hand poses passed through IK. We observe that our approach can achieve the task even when the IK output is poor (‘in-hand’) and offers solid performance when we observe better inputs (door). Using RL with user-augmented reward improves the IK baseline on ‘in-hand’, however it struggles when noise from hand pose estimator is added. ‘RL-no user’ performs well on ‘door-op’, however in this baseline the virtual hand does not follow the user input and acts independently, similarly to triggering a prerecorded sequence. In Fig. 4 we show qualitative results on ‘in-hand’ and in Fig. 3 (c) generated contact forces. Applying our models trained on the previous experiment did not perform well due to the different noise nature between both experiments, motivating our data generation scheme.

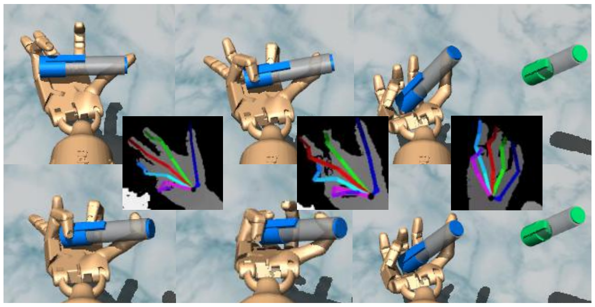

Fig. 4. Qualitative results on ‘in-hand manipulation task’. (Middle) estimated hand pose (Top) IK result (Bottom) Our result. Depth images are retrieved using our data generation scheme.

B. Physics-based hand-object sequence reconstruction

In this experiment we test our framework on the challenging task of transferring hand-object interactions from the real visual domain to a physically accurate simulation space. As a testbed, we use the First-Person Hand Action Benchmark (F-PHAB) [64], providing hand-object interaction sequences with hand and object pose annotations. We select two different manipulation tasks covering two extreme cases of power and precision grasps: ‘pour juice from a carton to a glass’ and ‘give coin to other person’. Each task contains 24 annotated video sequences from 6 different users and we use the 1:1 split of [64] for train-test data partition.

We recreate the real environment on the virtual space by placing a virtual object that we initialise with the 6D ground-truth pose. For the coin environment, we also place a target box that simulates ‘the hand of the other person’. We build the environments on MuJoCo and use the MPL hand model that consists of 23 DoF + 6 DoF [32] and ω^{pose} value of 0.01. As 3D hand pose estimator we use DeepPrior++ [65], extracting visual features $\phi_t \in \mathbb{R}^{1024}$ from the FC2 layer; and trained on the full dataset following the same 1:1

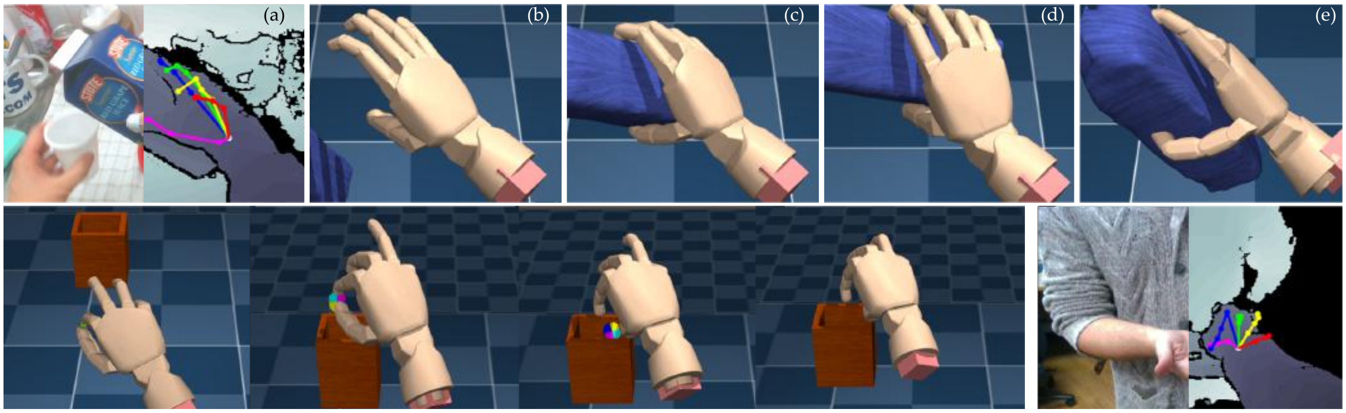

Fig. 5. Qualitative results. Top row: a frame belonging to 'pouring juice' action from F-PHAB dataset and its reconstruction for different methods from a fixed camera viewpoint. (a) RGB/depth image and estimated 3D hand pose. (b) IK function κ [8] on HPE. (c) Closing hand baseline on top of κ. (d) Our approach without visual features and pose reward. (e) Our full approach, it produces a hand posture closer to the one depicted by the reference visual hand motion. Bottom row: Qualitative result on a 'give coin' sequence. The task is achieved when the coin is placed on the other person's hand (red box).

protocol which yields an average test joint error of 14.54 mm. Note that in this setup we do not have access to expert demonstrations, thus we cannot compute r^{il} nor use our data generation scheme. The rest of network architectures and parameters are the same as in previous experiment.

Baselines: In this experiment we implement two baselines, an IK function κ following [8] and a 'closing' baseline that acts on top of κ and attempts to tighten the grasp or generate more contact forces similar to [10].

Metrics: We use three different criteria to measure performance. First, 'task success' measures the percentage of the times that the interaction is successful on test sequences. $\mathbf{E}_{\text{pose}}$ measures the 3D hand pose error, in mm, by reprojecting z_t to the input RGBD image space and comparing to ground-truth annotations, which gives us a notion on how similar the virtual posture looks compared to the actual visual pose. $\bar{T}$ measures the average length (in percentage over the total length) of the sequence before the simulation becomes unstable and the task is not completed successfully.

In Table IV we show quantitative results on 'pour juice' and 'give coin' actions. We observe that our approach is able to accomplish the task while keeping a hand posture similar to the visual input (qualitative results are shown in Fig. 5) and perform better than all baselines at train and test time. We observe that introducing the pose reward encourages a virtual pose closer to the visual input. Note that the virtual model is fixed in terms of bone lengths and kinematics and thus the reprojected pose will have an error offset even if the mapping was perfect. We show a successful example with generated contact forces in Fig. 3 (c)). We observe a significant gap between training and test results that is even more severe in the 'give coin' scenario where all the baselines show poor results in both training and test sets. Slight inaccuracies make the light and thin coin fall and thus failing in the task. We suspect that there are two main reasons for this. First, hand pose estimation errors are more severe than in the previous experiment and propagate through the hand model. Second the small number of training sequences may lead our network to overfit to the training set to some extent. This effect could be relieved by recording more training data or some data/trajectory augmentation technique. Note that the results on training sequences are still meaningful in the problem of offline motion reconstruction [13].

TABLE IV
HAND-OBJECT RECONSTRUCTION OF SEQUENCES IN-THE-WILD

	Training			Test		
Method (Pour Juice)	$\bar{T}\uparrow$	$\mathbf{E}_{\text{pose}}\downarrow$	Success ↑	$\bar{T}\uparrow$	$\mathbf{E}_{\text{pose}}\downarrow$	Success ↑
IK [8]	18.0	26.95	16.0	24.8	33.22	5.0
Closing hand	85.4	24.78	55.0	47.0	35.46	38.0
Ours w/o pose reward	97.4	26.82	84.0	52.0	37.88	47.0
Ours	**98.2**	25.43	**93.0**	**59.6**	**33.15**	**65.0**
Method (Give coin)	$\bar{T}\uparrow$	$\mathbf{E}_{\text{pose}}\downarrow$	Success ↑	$\bar{T}\uparrow$	$\mathbf{E}_{\text{pose}}\downarrow$	Success ↑
IK [8]	9.2	24.90	0.0	11.5	25.93	0.0
Closing hand	55.4	28.44	25.0	70.2	33.70	28.57
Ours	**95.5**	**24.3**	**80.0**	**92.1**	**25.30**	**83.3**

V. CONCLUSION AND FUTURE WORK

We presented a framework that can perform dexterous manipulation skills by simply using a hand pose estimator without the need of any costly hardware. A residual agent learns within a physics simulator how to improve the user input to achieve a task while keeping the motion close to the input and expert recorded trajectories. We showed that our approach can be applied on two applications that require accurate hand-object motion while using noisy input poses.

We believe this paper can inspire future work and it can be extended in several different ways. For instance, making the full framework end-to-end, where the gradients propagate from the simulator to the hand pose estimator, is a promising direction for physics-based pose estimation. For the second application, it would also be interesting to add a 6D object pose estimator in the loop [66]. Besides, generating synthetic data to close the training loop has also potential, for instance by fitting a realistic hand model in a similar way as in [67] on top of mocap data or already trained policies [1]. This could also help to narrow, to some extent, the training-test gap found in our experiments and make possible the deployment of the system to receive poses in a stream in a VR system and may prompt additional challenges. The study of RL generalization to both in-the-wild scenarios and other tasks is an open research problem. New results in these areas would benefit the present work, because how to scale up the number of tasks in the current framework is not clear.

References

[1] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *RSS*, 2018.

[2] T. Zhang, Z. McCarthy, O. Jow, D. Lee, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *ICRA*, 2018.

[3] S. Yuan, Q. Ye, B. Stenger, S. Jain, and T.-K. Kim, "Bighand2. 2m benchmark: Hand pose dataset and state of the art analysis," in *CVPR*, 2017.

[4] "Leap motion sdk," https://www.leapmotion.com.

[5] S. Li, X. Ma, H. Liang, M. Görner, P. Ruppel, B. Fang, F. Sun, and J. Zhang, "Vision-based teleoperation of shadow dexterous hand using end-to-end deep neural network," in *ICRA*, 2019.

[6] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect." in *BMVC*, 2011.

[7] S. Khamis, J. Taylor, J. Shotton, C. Keskin, S. Izadi, and A. Fitzgibbon, "Learning an efficient model of hand shape variation from depth images," in *CVPR*, 2015.

[8] D. Antotsiou, G. Garcia-Hernando, and T.-K. Kim, "Task-oriented hand motion retargeting for dexterous manipulation imitation," in *ECCVW*, 2018.

[9] "Microsoft hololens," https://www.microsoft.com/en-us/hololens.

[10] M. Höll, M. Oberweger, C. Arth, and V. Lepetit, "Efficient physics-based implementation for realistic hand-object interaction in virtual reality," in *IEEE VR*, 2018.

[11] D. Tzionas, L. Ballan, A. Srikantha, P. Aponte, M. Pollefeys, and J. Gall, "Capturing hands in action using discriminative salient points and physics simulation," *IJCV*, 2016.

[12] Y. Hasson, G. Varol, D. Tzionas, I. Kalevatykh, M. J. Black, I. Laptev, and C. Schmid, "Learning joint reconstruction of hands and manipulated objects," in *CVPR*, 2019.

[13] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine, "Sfv: Reinforcement learning of physical skills from videos," *SIGGRAPH Asia*, 2018.

[14] Y. Yuan and K. Kitani, "Ego-pose estimation and forecasting as real-time pd control," in *ICCV*, 2019.

[15] S. Reddy, S. Levine, and A. Dragan, "Shared autonomy via deep reinforcement learning," in *RSS*, 2018.

[16] S. Yuan et al., "Depth-based 3d hand pose estimation: From current achievements to future goals," in *CVPR*, 2018.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv*, 2017.

[18] OpenAI et al., "Learning dexterous in-hand manipulation," *IJRR*, 2020.

[19] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IROS*, 2012.

[20] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *NIPS*, 2016.

[21] Y. Zhu et al., "Reinforcement and imitation learning for diverse visuomotor skills," in *RSS*, 2018.

[22] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun, "Hand pose estimation and hand shape classification using multi-layered randomized decision forests," in *ECCV*, 2012.

[23] D. Tang, H. Jin Chang, A. Tejani, and T.-K. Kim, "Latent regression forest: Structured estimation of 3d articulated hand posture," in *CVPR*, 2014.

[24] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," *ToG*, 2014.

[25] M. Oberweger, P. Wohlhart, and V. Lepetit, "Hands deep in deep learning for hand pose estimation," in *CVWW*, 2015.

[26] C. Zimmermann and T. Brox, "Learning to estimate 3d hand pose from single rgb images," in *ICCV*, 2017.

[27] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt, "Ganerated hands for real-time 3d hand tracking from monocular rgb," in *CVPR*, 2018.

[28] J. Romero, D. Tzionas, and M. J. Black, "Embodied hands: Modeling and capturing hands and bodies together," in *SIGGRAPH Asia*, 2017.

[29] A. Boukhayma, R. de Bem, and P. H. Torr, "3d hand shape and pose from images in the wild," in *CVPR*, 2019.

[30] I. Cerulo, F. Ficuciello, V. Lippiello, and B. Siciliano, "Teleoperation of the schunk s5fh under-actuated anthropomorphic hand using human hand motion tracking," *IEEE RAS*, 2017.

[31] C. W. Borst and A. P. Indugula, "Realistic virtual grasping," in *VR*, 2005.

[32] V. Kumar and E. Todorov, "Mujoco haptix: A virtual reality system for hand manipulation," in *IEEE RAS*, 2015.

[33] J. Kofman, S. Verma, and X. Wu, "Robot-manipulator teleoperation by markerless vision-based hand-arm tracking," *Optomechatronics*, 2007.

[34] J. Romero, "From human to robot grasping," Ph.D. dissertation, KTH Royal Institute of Technology, 2011.

[35] G. Du, P. Zhang, J. Mai, and Z. Li, "Markerless kinect-based hand tracking for robot teleoperation," *I.J. of Adv. Robotic Systems*, 2012.

[36] A. Handa, K. Van Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox, "Dexpilot: Vision based teleoperation of dexterous robotic hand-arm system," in *ICRA*, 2020.

[37] V. Buchmann, S. Violich, M. Billinghurst, and A. Cockburn, "Fingartips: gesture based direct manipulation in augmented reality," in *GRAPHITE*, 2004.

[38] M. Moehring and B. Froehlich, "Effective manipulation of virtual objects within arm's reach," in *IEEE VR*, 2011.

[39] D. Yim, G. N. Loison, F. H. Fard, E. Chan, A. McAllister, and F. Maurer, "Gesture-driven interactions on a virtual hologram in mixed reality," in *ISS Companion*, 2016.

[40] H. Rijpkema and M. Girard, "Computer animation of knowledge-based human grasping," in *SIGGRAPH*, 1991.

[41] G. ElKoura et al., "Handrix: animating the human hand," in *SIGGRAPH*, 2003.

[42] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *ICRA*, 2003.

[43] Y. Li, J. L. Fu, and N. S. Pollard, "Data-driven grasp synthesis using shape matching and task-based pruning," *IEEE TVCG*, 2007.

[44] M. Prachyabrued and C. W. Borst, "Virtual grasp release method and evaluation," *I.J. of Human-Computer Studies*, 2012.

[45] P. Kry et al., "Interaction capture and synthesis," in *SIGGRAPH*, 2006.

[46] C. K. Liu, "Dextrous manipulation from a grasping pose," in *SIGGRAPH*, 2009.

[47] Y. Ye et al., "Synthesis of detailed hand manipulations using contact sampling," *SIGGRAPH*, 2012.

[48] W. Zhao, J. Zhang, J. Min, and J. Chai, "Robust realtime physics-based motion control for human grasping," *TOG*, 2013.

[49] J.-S. Kim and J.-M. Park, "Physics-based hand interaction with virtual objects," in *ICRA*, 2015.

[50] H. Hamer, K. Schindler, E. Koller-Meier, and L. Van Gool, "Tracking a hand manipulating an object," in *CVPR*, 2009.

[51] N. Kyriazis and A. Argyros, "Physically plausible 3d scene tracking: The single actor hypothesis," in *CVPR*, 2013.

[52] N. Kyriazis and A. Argyros, "Scalable 3d tracking of multiple interacting objects," in *CVPR*, 2014.

[53] G. Rogez, J. S. Supancic, and D. Ramanan, "Understanding everyday hands in action from rgb-d images," in *ICCV*, 2015.

[54] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints," in *ICCV*, 2011.

[55] A. Tsoli and A. A. Argyros, "Joint 3d tracking of a deformable object in interaction with a hand," in *ECCV*, 2018.

[56] T. Geijtenbeek, M. Van De Panne, and A. F. Van Der Stappen, "Flexible muscle-based locomotion for bipedal creatures," *TOG*, 2013.

[57] N. Heess et al., "Emergence of locomotion behaviours in rich environments," *arXiv*, 2017.

[58] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," in *SIGGRAPH*, 2018.

[59] N. Chentanez, M. Müller, M. Macklin, V. Makoviychuk, and S. Jeschke, "Physics-based motion capture imitation with deep reinforcement learning," in *ACM MIG*, 2018.

[60] L. Liu and J. Hodgins, "Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning," *TOG*, 2018.

[61] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *ICRA*, 2019.

[62] T. Silver, K. Allen, J. Tenenbaum, and L. P. Kaelbling, "Residual policy learning," *arXiv*, 2018.

[63] Q. Ye, S. Yuan, and T.-K. Kim, "Spatial attention deep net with partial pso for hierarchical hybrid hand pose estimation," in *ECCV*, 2016.

[64] G. Garcia-Hernando, S. Yuan, S. Baek, and T.-K. Kim, "First-person hand action benchmark with rgb-d videos and 3d hand pose annotations," in *CVPR*, 2018.

[65] M. Oberweger and V. Lepetit, "Deepprior++: Improving fast and accurate 3d hand pose estimation," in *ICCVW*, 2017.

[66] C. Sahin, G. Garcia-Hernando, J. Sock, and T.-K. Kim, "A review on object pose recovery: from 3d bounding box detectors to full 6d pose estimators," *Image and Vision Computing*, 2020.

[67] A. Armagan et al., "Measuring generalisation to unseen viewpoints, articulations, shapes and objects for 3d hand pose estimation under hand-object interaction," in *ECCV*, 2020.

Appendix

1. Learning algorithm box

The pseudocode for the training process of our framework can be found in Algorithm 1.

Algorithm 1: Framework training with data generation scheme, PPO and GAIL

```
$\theta, v, \psi \leftarrow$ initialize policy, value function and discriminator weights
reset $\leftarrow$ true
while not done do
    Generate policy rollouts:
    for step$= 1, ..., m$ do
        if reset then
            $\tau \leftarrow$ Randomly select one expert sequence of length $T$
            $\{x_t, \phi_t\} \leftarrow$Generate hand pose and visual features sequences with Algorithm 2 and $\tau$
            Generate $\kappa(x_t(\phi_t))$ and augment translation actions from $\tau$ with noise
            Initialize environment to state $s_0$ from $\tau$
        end if
        $f_t \sim \pi_\theta(f_t|s_t, \kappa(x_t(\phi_t)), \phi_t)$
        $a_t \leftarrow \kappa(x_t(\phi_t)) - f_t$
        Apply $a_t$ and simulate forward one step
        $s_{t+1} \leftarrow$ end state
        $r_t \leftarrow \omega^{\text{task}} r_t^{\text{task}} + \omega^{\text{il}} r_t^{\text{il}} + \omega^{\text{pose}} r_t^{\text{pose}}$
        Record $(s_t, a_t, r_t, s_{t+1})$ into memory
    end for
    Update $\theta$, $v$ and $\psi$ with their respective gradients and recorded batches by following [17] and [20]
end while
```

2. Learning algorithm parameters

We use the same learning parameters for both our method and the baselines. Our approach and Hybrid/IL baselines have three networks: policy network, value function network and discriminator. RL baselines have only policy and value networks. Policy updates are performed after a batch of $m = 4096$ samples has been collected and minibatches of size 256 are sampled for each gradient step. We use Adam optimizer with learning rate for the policy and value function of $3 \cdot 10^{-4}$ and 10^{-4} for the discriminator network. After a batch of data is collected 15 parameter updates are performed for both policy and value networks and 5 for the discriminator. PPO parameters [17] are as follows: 0.995 for the temporal discount, 0.97 for the GAE parameter and 0.2 for the clipping threshold. The initial policy standard deviation is set to 0.01. λ is set to 0.5 in all the experiments involving hybrid reward. Networks updates are performed on a single GPU (NVIDIA GTX 1080Ti) using TensorFlow and data samples are generated solely on CPU. For the supervised mapping function $\kappa(x_t)$ used in Experiment Section 4.1.2, we use a 2 layer MLP (64-64), Adam update 10^{-3}, 32 batch size and 10^6 iterations. We optimize this mapping on the same GPU as above and using TensorFlow.

3. Details on data generation scheme

If we are given an expert action $a_t \sim \pi_E$ and we observe Eq. 2, it holds that $f_\theta(s_t, \kappa(x_t(\phi_t)), \phi_t) \approx 0$. Given that a_t has been recorded using a data glove, we can assume that the depicted pose from the virtual pose z_t has a similar posture to x_t. We set a camera viewpoint in the simulation to be placed in a similar place as the depth sensor in the real space and compute the relative viewpoint v_t, i.e. elevation and azimuth angles, between this camera viewpoint and a normal vector from the z_t's palm. Note that both poses belong to different domains and thus are not directly comparable. To deal with this, we normalize z_t to have all the joint links to be unit vectors, and rotate the palm to be aligned to a certain plane obtaining $\hat{z}_t$. We follow the same process on a hand pose dataset containing pairs of depth images and ground-truth hand pose annotations. We then query the dataset to first retrieve all the ground-truth hand poses that have the same viewpoint v_t, and retrieve the nearest neighbor by computing the Euclidean distance between $\hat{z}_t$ and the normalized set of ground-truth candidates. We then retrieve the associated depth image and compute the estimated hand pose. We do not consider the translation of the hand in the image given that this seriously limits the number of candidate poses, however we deal with this by generating different positions by adding noise on the ground-truth translations, which also allows us to generate diverse realistic sequences, thus making our training more diverse. See Algorithm 2.

Algorithm 2: Data generation scheme

```
Input: $\tau = \{s_t, a_t\} \in \mathcal{D}$ sequence of expert demonstrations of length $T$
$s_0 \leftarrow$ sample initial state from $\tau$
while $t < T$ do
    Apply $a_t$ to the environment
    $z_t \leftarrow$ read simulation sensors
    $v_t \leftarrow$ compute relative viewpoint between $z_t$'s palm and simulator camera
    $\hat{z}_t \leftarrow$ normalize and align $z_t$
    $\hat{x}_t \leftarrow$ query dataset with $v_t$ and $\hat{z}_t$
    $x_t, \phi_t \leftarrow$ apply hand pose estimator to $\hat{x}_t$'s associated image
end while
Output: $\{x_t\}$, $\{\phi_t\}$ sequences of estimated hand poses and visual features
```

4. Experiment A: Experiment Details

Experiment A.1: details and additional results

MoCap user demonstrations are sampled from the dataset provided in [1]. For each task we are given 24 MoCap demonstrations and we split them in equal training and test sets. We use the original environments from Rajeswaran *et al.* [1] with the modification of adding user input and making the action residual. We keep the simulator, physics parameters as in the original environments[2] and states and task rewards are described in Section 5.

[2] https://github.com/aravindr93/hand_dapg

Training and test data is generated by adding random Gaussian noise to these demonstrations. Reported train/test results are after averaging results for 100 policy rollouts (augmented with noise) and three random seeds on learning algorithm. We provide additional learning curves for the rest of the tasks curves on Figure 8. We also show the learning curves for different levels of training noise (used to generate Table 1 of the main paper) in Figure 6.

Experiments A.2: Details and additional results

We use the same training/test split as in previous experiment. BigHand2.2M dataset is queried for one user ($2 \cdot 10^5$ samples) that has not been seen by the hand pose estimator in training. As a hand pose estimator we used the approach of [63] trained on the rest of the subjects of BigHand2.2M. We report the following average hand pose estimation errors (Euclidean distance between annotation and estimation) for the different tasks, which are consistent with state-of-the-art results for depth images: 9.90 mm (door op.), 7.74 mm (in-hand man.), 8.76 mm (tool) and 7.42 mm (relocation).

For the dataset generation scheme the order of the query, i.e. first viewpoint or posture, will have an impact in the result. We empirically found that the best results were obtained with the following order: first query candidates based on azimuth, then proceed with altitude and end with pose distance.

In Figure 7 we show learning curves for our approach and other baselines, we observe that our training converges faster and to a higher task success than the baselines. In Table V we expand the Table III from the main paper that we had to reduce for space reasons. Train and test split is generated as in previous experiment and the x_t is generate following our data generation scheme. Noise of intensity 0.05 rad is added only to the translation and rotation actuators of the arm from the demonstrations, the rest of the user action (24) comes from $\kappa(x_t)$. Results are shown on generated test sequences for both ground-truth hand poses (GT) and estimated hand poses (Est.) on 100 policy rollouts for three different learning random seeds as in previous experiments. We observe that our approach outperforms all the baselines, except for the relocation environment, which is consistent with our result in A.1 and discussed in the main paper.

5. Experiment A: states, rewards and tasks

The state space and the task rewards are the same as in the work of [1] with the addition of $\kappa(x)$ to the state and an user following reward (only on + user reward baselines). The action space is the same as in [1] for each task.

Door opening

The user has to undo the latch before the door can be opened. The latch has a significant dry friction and a bias torque that forces the door to be closed. The success measure is defined as $door_{joint} > 1.0$ at the end of the interaction. The state is defined as

$$s_{\text{door}} = [hand_{joints}; palm_{pos}; door_{handle\ pos,latch,hinge}]$$

and the reward as:

$$\begin{aligned} r_{\text{door}} =& 10\mathbf{I}(door_{pos} > 1.35) + 8\mathbf{I}(door_{pos} > 1.0) \\ &+ 2\mathbf{I}(door_{pos} > 1.2) - 0.1||door_{pos} - 1.57||_2. \end{aligned}$$

Tool Use: Hammer

We consider using a hammer to drive in a nail. The user hand has to pick up the hammer from the ground, move it over to the nail and hammer in with a significant force to get the nail to move into the board. The nail has dry friction capable of absorbing up of 15N of force. There are more than one steps needed to perform this task, which require accurate grasping and positioning. The success measure is defined as $||nail_{pos} - nail^{goal}_{pos}||_2 < 0.01$ at the end of the interaction. The state is defined as:

$$\begin{aligned} s_{\text{hammer}} = [&hand_{joints,velocity}; palm_{pos}; hammer_{pos,rot}; \\ &nail^{goal}_{pos}; nail_{impactforce}] \end{aligned}$$

and the reward as:

$$\begin{aligned} r_{\text{hammer}} =& 75 * \mathbf{I}(||nail^{goal}_{pos} - nail_{pos}||_2 < 0.10) + \\ & 25 * \mathbf{I}(||nail^{goal}_{pos} - nail_{pos}||_2 < 0.02) - \\ & 10||nail^{goal}_{pos} - nail_{pos}||_2. \end{aligned}$$

In-hand Manipulation: Repositioning a pen

The user goal is to reposition the blue pen to a desired target orientation in-hand, visualized by the green pen. The base of the hand is fixed. The pen is highly underactuated and requires careful application of forces by the hand to reposition it. Most actions lead to catastrophic failure like dropping the object. The success measure is $||pen_{rot} - pen^{goal}_{rot}||_{cosine} > 0.95)$ at the end of the interaction. The state is defined as

$$s_{\text{pen}} = [hand_{joints}; pen_{pos,rot}; pen^{goal}_{pos,rot}]$$

and the reward as:

$$\begin{aligned} r_{\text{pen}} =& 50(\mathbf{I}(||pen^{goal}_{pos} - pen_{pos}||_2 < 0.075) \otimes \\ & \mathbf{I}(||pen_{rot} - pen^{goal}_{rot}||_{cosine} > 0.95)). \end{aligned}$$

Object relocation

The user goal is to use the hand to pick up the blue ball and move it to the green target location. The success measure is $||object_{pos} - object^{goal}_{pos}||_2 < 0.05)$ at the end of the interaction. The state is defined as:

$$s_{\text{relo}} = [hand_{joints}; palm_{pos}, object_{pos}; object^{goal}_{pos}]$$

and the reward is defined as:

$$\begin{aligned} r_{\text{relo}} =& 10\mathbf{I}(||object_{pos} - object^{goal}_{pos}||_2 < 0.1) + \\ & 20\mathbf{I}(||object_{pos} - object^{goal}_{pos}||_2 < 0.05). \end{aligned}$$

User reward (+ user reward baselines only)

We design a reward function to encourage the agent to follow user action and add it to the reward functions presented above for the RL+user reward and Hybrid+user reward baselines. It is defined as:

$$r_{user} = -0.1||a - \kappa(x)||_2.$$

6. Experiment B: Experiment Details

Networks and hyperparameters are the same as described above. As hand pose estimator we use DeepPrior++ [65] trained with the protocol described in [64]. Visual features ϕ are extracted from the last full connected layer with dimension 1024. The environments are implemented in MuJoCo using an extended version of the MPL model [32] by [8], we query the policy at 30 Hz and the simulator at 200 Hz. We use the inverse kinematics function from [8] to do the first mapping between estimated hand poses and hand model. We control the 23 actuators of the hand model, while the global rotation and orientation of the model in the virtual space are predicted by the hand pose estimator. As action representation, PD controllers are used to compute joint torques with default gain parameters [32]. In this experiment we found that using early termination (i.e. resetting the environment when the object falls far away) helped to make the training converge faster. 25% of the action space is used as residual domain.

Pouring juice action

This task consists of holding a juice bottle and making the pour action by following the user input. The environment has only one object consisting of the juice bottle. The task is considered successful if at the end of the clip the virtual model is holding the bottle. There are 24 sequences of about 100 frames each and we use 12 for training and 12 for test. The state is defined as:

$$s_{\mathsf{juice}} = [hand_{joints}, hand_{vel}, object_{pos} - hand_{pos}, object_{rot} - hand_{rot}, contact],$$

where *contact* is a function that measures the normalized distance between contact points between finger tips and object surface similar to the PSO fit function of [8]. We place a sensor to measure contacts at each finger tip, and thus *contacts* has a value for each fingertip (5-D). The input to the policy network is 1104 (this number includes the state, the visual features and the user input).

$$r_{\mathsf{juice}} = \mathbf{I}(\sum contacts > 0)(1 - \sum contacts) - ||object_{pos} - hand_{pos}||_2$$

Closing hand baseline: This baseline is implemented by enforcing contact between object surface and finger tips by first reading user input and moving the actuators towards the object. We use this baseline as user input to our residual policy.

Give coin action

The task consist of the user placing a coin on other user's hand. The virtual environment has three components: hand model, coin object and box. The box represents the other user's hand and the task is to place the coin within the limits of the box. There are two big challenges in this task: holding the small coin and carefully placing it in the box. In the original dataset there are 25 sequences, however we had to discard five of them due to simulator instabilities.

The state is defined as:

$$s_{\mathsf{coin}} = [hand_{joints}, hand_{vel}, coin_{pos} - index^{tip}_{pos}, coin_{pos} - thumb^{tip}_{pos}, box_{pos}, box_{pos} - index^{tip}_{pos}, box_{pos} - thumb^{tip}_{pos}, box_{pos} - coin_{pos}, coin_{rot} - hand_{rot}, contact],$$

where in this case *contacts* is limited to index and thumb finger tips. The input to the policy network is 1119.

The reward for this task consists of two parts that depend on the two main stages of the action: holding the coin and carefully placing it in the box. The reward function for this task is defined as follows:

$$r_{\mathsf{coin}} = \mathbf{I}(d(box_{pos}, index^{tip}_{pos}) > 0.12 \wedge d(box_{pos}, coin_{pos}) > 0.05) \times (1 - \sum contacts) - d(coin_{pos}, thumb^{tip}_{pos}) - d(coin_{pos}, index^{tip}_{pos})) + \mathbf{I}(d(box_{pos}, index^{tip}_{pos}) \leq 0.12 \vee d(box_{pos}, coin_{pos}) \leq 0.05) \times (1.5\mathbf{I}(d(box_{pos}, coin_{pos}) < 0.05) + \mathbf{I}(d(box_{pos}, coin_{pos}) < 0.08) - d(box_{pos}, index^{tip}_{pos})),$$

where $d(\cdot, \cdot)$ represents the Euclidean distance between two bodies. The first two lines of the equation represent the 'approaching box' phase where contact between index, thumb and coin are encouraged. The last two lines give a high reward when the coin is within the limits of the box and penalizes when the coin is far from the target (e.g. the coin fell outside the box).

Closing hand baseline: In this task we move the index and thumb fingers towards the coin to make a 'pinch' gesture. Once the hand is near the box, the fingers release the coin making also a subtle wrist movement. We use this baseline as user input to our residual policy.

7. Qualitative results

Videos can be visualised on the project webpage: https://sites.google.com/view/dexres

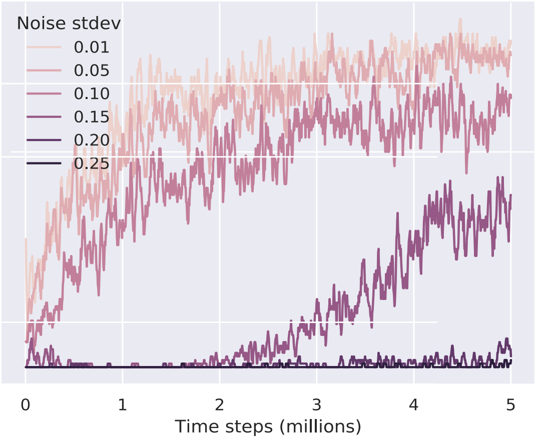

Fig. 6. Training curves for different levels of noise in training. These models were used to generate Table I in the main paper.

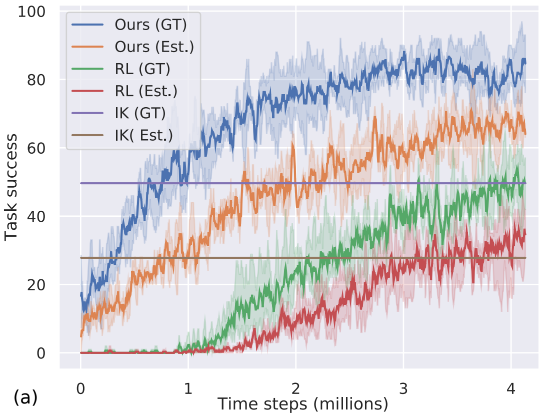

Fig. 7. Experiment A.2: training curves (door).

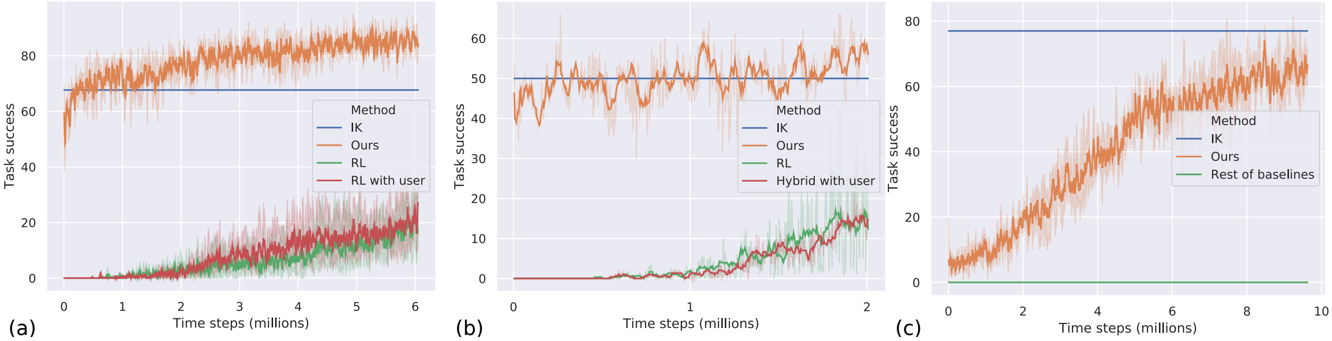

Fig. 8. Experiment A.1: Additional learning curves learning curves for our approach and the baselines that performed the best on **Table II**. **(a)** In-hand manipulation task **(b)** Tool use (hammer) and **(c)** Object relocation. Plots are generated for three different random seeds and confidence interval (95%)

TABLE V

EXPERIMENT A.2: BASELINE COMPARISON FOR ALL BASELINES AND TASKS (EXPANDS TABLE III)

	Door opening		In-hand man.		Tool use (hammer)		Object relocation	
Method (Training set)	GT	Est.	GT	Est.	GT	Est.	GT.	Est.
IK	49.62	27.81	0.00	20.30	66.16	68.42	82.70	90.22
RL - no user (GT)	98.49	76.69	13.53	25.56	34.59	29.32	0.00	0.00
IL - no user (GT)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Hybrid - no user (GT)	0.00	0.00	20.30	9.02	39.84	37.59	0.00	0.00
RL - no user (Est.)	66.16	71.42	13.53	0.00	58.65	54.89	0.00	0.00
IL - no user (Est.)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Hybrid - no user (Est.)	0.00	0.00	12.03	10.52	53.38	47.37	0.00	0.00
RL + user reward (GT)	0.00	0.00	**45.86**	32.33	3.76	3.76	0.00	0.00
Hybrid + user reward (GT)	0.00	0.00	0.00	12.03	58.64	29.32	0.00	0.00
RL + user reward (Est.)	0.00	0.00	0.00	12.03	12.78	4.51	0.00	0.00
Hybrid + user reward (Est.)	0.00	0.00	0.00	0.00	54.13	68.00	0.00	0.00
Ours (Experiment A.1)	57.14	38.34	10.52	0.00	60.15	30.82	21.80	29.32
Ours (GT poses)	83.45	42.10	10.52	32.33	78.00	25.56	34.00	12.78
Ours (Est. poses)	**85.95**	**70.67**	20.33	**57.14**	**78.94**	**71.42**	34.00	35.00