Coarse-to-Fine Imitation Learning: Robot Manipulation from a Single Demonstration

Edward Johns [*]

Abstract— **We introduce a simple new method for visual imitation learning, which allows a novel robot manipulation task to be learned from a single human demonstration, without requiring any prior knowledge of the object being interacted with. Our method models imitation learning as a state estimation problem, with the state defined as the end-effector's pose at the point where object interaction begins, as observed from the demonstration. By modelling a manipulation task as a coarse, approach trajectory followed by a fine, interaction trajectory, this state estimator can be trained in a self-supervised manner, by automatically moving the end-effector's camera around the object. At test time, the end-effector is moved to the estimated state through a linear path, at which point the demonstration's end-effector velocities are simply repeated, enabling convenient acquisition of a complex interaction trajectory without actually needing to explicitly learn a policy. Real-world experiments on 8 everyday tasks show that our method can learn a diverse range of skills from just a single human demonstration, whilst also yielding a stable and interpretable controller.**

I. INTRODUCTION

The goal of imitation learning can be described as teaching a robot how to perform a novel task from human demonstrations, whilst minimising two criteria: (1) the amount of physical interaction required of the human, and (2) the amount of prior task knowledge required by the algorithm. Existing solutions typically fail in one or both of these. Methods that learn from demonstrations alone, such as behavioural cloning [1], require a large number of demonstrations. Methods that bootstrap from demonstrations with additional self-exploration, such as reinforcement learning [2], require environment resetting via manual intervention or task-specific apparatus. Methods that transfer knowledge from other similar tasks, such as meta-imitation learning [3], require prior knowledge of the task family. And methods that analytically model tasks in a manually-defined state space, such as dynamic movement primitives [4], require prior knowledge of the specific task. In this paper, we introduce a simple new method which addresses both these criteria: our method learns everyday robot manipulation tasks from just a single human demonstration, without requiring any prior knowledge of the object which is being interacted with.

Our method **models the robot's motion as a coarse-to-fine trajectory**, as shown in Figure 1. In typical object manipulation tasks, we observe that the end-effector first *approaches* the object's bottleneck in a *coarse* manner through free space, and then physically *interacts* with the object in a *fine* manner. We can thus model the approach analytically

[*] Edward Johns is with the Robot Learning Lab at Imperial College London. This work was supported by the Royal Academy of Engineering under the Research Fellowship scheme.

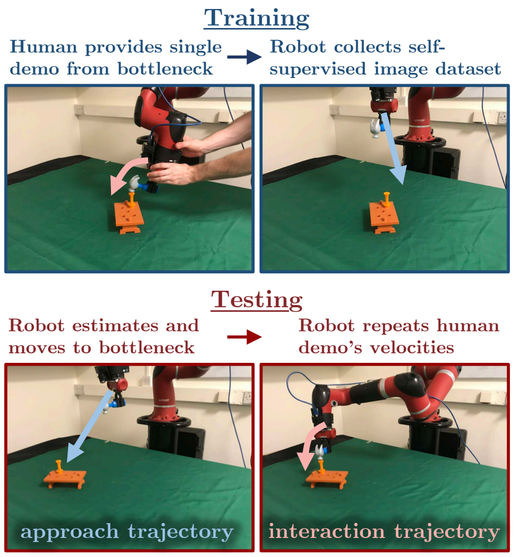
Training
Human provides single demo from bottleneck
Robot collects self-supervised image dataset
Testing
Robot estimates and moves to bottleneck
Robot repeats human demo's velocities
approach trajectory
interaction trajectory

Fig. 1: We model a robot manipulation task as a coarse *approach trajectory*, followed by a fine *interaction trajectory*. Training for the approach trajectory is self-supervised, and the interaction trajectory needs only a single demonstration. Here, we illustrate teaching the robot how to use a hammer.

with a simple linear path. And therefore, rather than requiring demonstrations in this space, a camera mounted to the end-effector can be moved around the object automatically to collect observations from multiple viewpoints, thus achieving self-supervised generalisation across the task space.

Building on this foundation, our method then **models imitation learning as a state estimation problem**. Typically, state estimation for robot manipulation requires prior knowledge of the object, such as via a pre-defined pose estimator [4] or fiducial markers [5], [6]. We show how this can be achieved with a novel object, by using the demonstration itself to define a state in 3D space. This state is defined as the pose of the object's *bottleneck*, which intuitively represents the end-effector's pose at the point where object interaction begins, as observed from the demonstration, which the robot should then attempt to reach during testing. A bottleneck pose predictor can then be trained using self-supervised learning, as introduced above. During testing, the robot estimates the bottleneck pose and moves the end-effector directly there. This then leads to the crux of our method: if the robot can reach the bottleneck pose with sufficient accuracy, we propose that the robot can then just repeat the original demonstration's end-effector velocities from the bottleneck onwards, with the simplest form of behavioural cloning. This enables convenient acquisition of a complex interaction trajectory without needing to explicitly learn a policy. And since the robot explores only in free space, and not during object interaction, we avoid the need for repeated task resetting which reinforcement learning suffers from.

In summary, we propose **Coarse-to-Fine Imitation Learning**, which combines the strengths of both analytical modelling and machine learning, for a data-efficient yet flexible framework. Furthermore, since machine learning is used only for pose estimation and not for policy learning, the controller itself is analytical, so we provide stability and interpretability in contrast to many recent methods in the field which adopt black-box, end-to-end policy learning. In real-world experiments, we study a number of different implementations of our method, and show how it can learn a diverse range of everyday tasks. A video is available at: **www.robot-learning.uk/coarse-to-fine-imitation-learning**.

II. Related Work

Whilst there is significant prior work on imitation learning using manually-defined low-dimensional state representations [4], [5], [6], these require task-specific prior knowledge, such as pre-defined pose estimators, or artificial experimental setups, such as fiducial markers. Therefore, in this section, we focus on methods where a robot can learn a new task from a human without requiring state-space engineering, particularly through visual observations.

Behavioural cloning methods use supervised learning to map observations to actions. This is often achieved through end-to-end learning, where a range of experimental setups have been designed to capture actions [1], [7], [8], [9], [10], [11]. Image representations can be regularised through keypoint-based architectures [1] or self-supervision [8]. Similar to our coarse-to-fine formulation, [12] combine planning with behavioural cloning. However, since all these methods explicitly learn an end-to-end policy, they require a large number of human demonstrations to achieve generalisation, whereas our method requires only a single demonstration.

Exploration-based methods use self-exploration of the environment to learn a policy, bootstrapping from the original demonstrations. This can be done by pre-training the policy [13], learning a residual policy [14], [15], pre-loading the replay buffer [2], adding a behavioural cloning loss to the actor [16], or restricting the exploration space [17], [18]. A reward function can be inferred from a goal image [19], inverse reinforcement learning [20], active learning [21], or contrastive learning [22]. Generative methods can attempt to reconstruct the demonstrations [23], [24], [25]. Exploration data can also be used to learn a dynamics model for trajectory optimisation, whose cost is defined by a demonstration [26], [27]. However, self-exploration methods can be unsafe, and additionally require environment resetting via human intervention or task-specific apparatus, whereas our method does not require any further exploration of object interaction.

Transfer learning methods aim to exploit prior knowledge of similar tasks, to learn a new task from one or a few demonstrations. This can be achieved by supervised learning across a task family [28], or by jointly learning a task embedding and policy [29]. Meta-learning an initialisation of network parameters for fast adaptation [30] can be deployed for behavioural cloning [3] and learning from observations of humans [31]. However, the efficient adaptation in these methods comes at the cost of requiring prior training on similar tasks to the new task being learned, whereas our method assumes no prior knowledge of the task family.

Our method is also related to **visual servoing**. [32], [33] attempt to align the robot's current image observation with a goal image, but require task-specific controllers to be manually defined for object interaction, whereas our method can learn from just a demonstration. [34] use visual servoing to directly track a demonstration, but require close initial alignment and only evaluate on tasks with very limited object interaction, whereas our method generalises across a wide task space and can facilitate complex interaction trajectories.

III. Method

A. Coordinate Frames

We first define the following coordinate frames: the robot's base R, the robot's end-effector E, and the object's bottleneck B, as shown in Figure 2. The bottleneck is a "virtual" frame and does not represent a physical body in the current scene, but represents where the end-effector should be at the point where interaction begins, such that all coarse trajectories converge at the bottleneck. B can considered to be fixed relative to the object. A homogeneous transformation matrix T_{EB} represents frame B expressed in frame E.

B. Coarse-to-Fine Trajectories

We observe that typical manipulation tasks begin with a coarse *approach trajectory*, followed by a fine *interaction trajectory*. During the approach, only the destination is important and not the specific trajectory, whereas during the interaction, the specific trajectory is crucial. We therefore execute the approach trajectory as a simple linear path using inverse kinematics. This does not require human demonstrations, and so we only require a demonstration of the interaction trajectory. From here, two questions now emerge. (1) During testing, what should the robot execute as the interaction trajectory? (2) During testing, where should the interaction trajectory begin? We now introduce two axioms for our assumed environment setup, which answer these.

The **first axiom** is that, for a controller which outputs actions relative to the end-effector's local frame, those actions should depend only on the relative pose between the end-effector and the object. Therefore, if we record the demonstration's local end-effector velocities from the bottleneck point onwards, then during testing, we can move the end-effector to the bottleneck and then repeat those velocities for the interaction trajectory. This answers our first question. Note that since the interaction trajectory is executed open loop, any error in reaching the bottleneck would result in a growing offset from the demonstration, during the interaction trajectory. Therefore, we minimise the length of

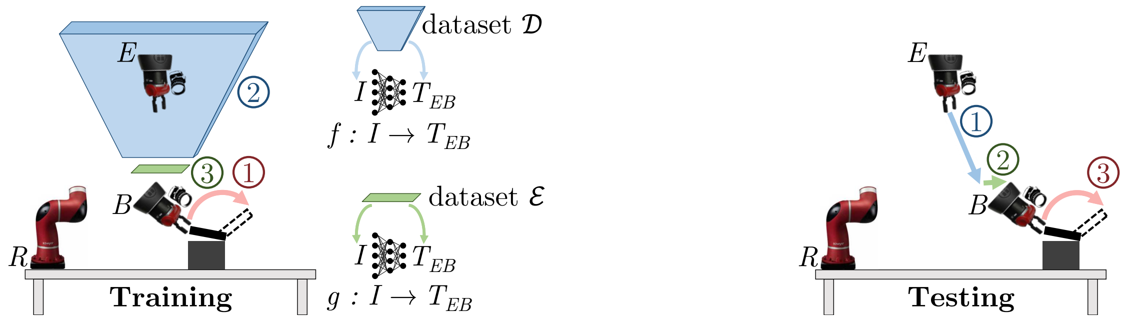

Fig. 2: Visualising the different stages of training and testing. During training, ① first a single human demonstration is provided from the bottleneck (e.g. here, opening a lid), ② then the robot moves its end-effector and camera around the bottleneck in the blue region, whilst collecting a dataset $\mathcal{D}$ of images I and bottleneck poses T_{EB} to train network $f(I)$, ③ then the robot collects a similar dataset $\mathcal{E}$ but in the smaller green region, to train network $g(I)$. During testing, with the object now moved, ① first the robot uses $f(I)$ to estimate the bottleneck pose T_{EB} and move towards it, ② then the robot makes a last-inch correction to its estimate using $g(I)$, ③ then the robot repeats the demonstration's end-effector velocities.

the interaction trajectory by beginning the demonstration as close to the object as possible, and define the bottleneck pose T_{RB} during training as the demonstration's initial end-effector pose T_{RE}. The demonstration should begin no lower than the object's highest point, to avoid collisions during the self-supervised data collection (see Section III-C).

The **second axiom** is that the camera's image is a function of the relative pose between the camera and the object. Since we mount the camera rigidly to the end-effector, and the bottleneck is fixed relative to the object, it follows that this image is also a function of the relative pose between the end-effector and the bottleneck, T_{EB}. Therefore, we can collect images and associated T_{EB} values by automatically moving the camera around the object, and then train a function which predicts T_{EB} from an image. Then, during testing, the robot can predict the pose of the bottleneck, move the end-effector there, and then execute the interaction trajectory. This answers our second question. Note that in reality, this axiom is an approximation due to observation noise, illumination effects, and variation in background, although these are not significant in our experimental setup and could potentially be mitigated by image augmentation.

C. *Self-Supervised Learning*

Data collection. We define $f(I)$ as the above function to predict T_{EB} from image I. After the demonstration, we collect dataset $\mathcal{D} = \{(I, T_{EB})_i\}$, with T_{EB} calculated by:

$$T_{EB} = T_{ER}\, T_{RB}, \tag{1}$$

where T_{ER} is calculated using forward kinematics, and T_{RB} is the bottleneck pose defined from the demonstration. Note that Equation 1 assumes that the object is in the same pose when T_{RB} is first defined, as when $\mathcal{D}$ is collected. Therefore, if the object moves during the human demonstration, then either it should be re-positioned, or alternatively, $\mathcal{D}$ can be collected before the human demonstration, by first manually setting the bottleneck pose with the end-effector, and later returning to this pose to start the demonstration. Now, since the approach trajectory is a simple linear path and does not require human demonstrations, we can collect $\mathcal{D}$ automatically, by moving the end-effector around in the space above the bottleneck (the blue region in Figure 2), to capture the appearance of the object from a range of viewpoints. We do this over a number of trajectories, where each first moves the robot to a random initial pose above the object, and then moves in a straight line at a constant speed towards a pose sampled from near the bottleneck, whilst adding (I, T_{EB}) pairs to $\mathcal{D}$. Given dataset $\mathcal{D}$, we then train $f(I)$ as a convolutional neural network using supervised learning. This is done by directly regressing the pose, and therefore we do not require any camera calibration, as long as the camera remains rigidly mounted to the end-effector.

Simplifying the prediction space. Now, since the specific motion during the approach trajectory is not important, we can constrain the space of bottleneck poses which the trajectory terminates at, and hence simplify the prediction space of the pose estimator $f(I)$. We assume a typical tabletop setup, where the object is limited to horizontal translation and rotation about the vertical axis. To capture only this rotation dimension, we constrain the end-effector poses to be "vertical" and pointing directly downwards, and allow end-effector rotation only about the vertical axis, both when collecting $\mathcal{D}$, and during the approach trajectory when testing. Given the known height of the bottleneck, this reduces the prediction space of $f(I)$ to three dimensions: two horizontal translations, and one rotation about the vertical. However, all this assumes that the bottleneck is indeed vertical, even though the end-effector's pose at the start of the demonstration may not be. Therefore, we define the bottleneck as having the same position as the demonstration's initial pose, but with a vertical orientation. We then record R, the 3D rotation between the bottleneck and this initial pose, which is later applied during testing at the end of the approach trajectory, to re-orientate with the demonstration. Note that during testing, the interaction trajectory can still be 6-DOF despite the bottleneck pose estimation being 3-DOF.

Last-inch correction. Together with collecting dataset $\mathcal{D}$ to train $f(I)$, we also collect a second dataset $\mathcal{E}$ to train a second network $g(I)$, to enable a "last-inch" correction to the bottleneck pose estimate. $\mathcal{E}$ is collected in a similar manner to $\mathcal{D}$, except that the end-effector's height is fixed at the bottleneck's height throughout data collection, such that each trajectory only moves horizontally with rotation about the vertical (the green region in Figure 2). This ensures that $g(I)$ specialises in predictions where the end-effector is already very close to the bottleneck, without images from the remainder of the task space consuming network capacity. During testing, at the end of the approach trajectory, $g(I)$ is then used to make one final estimate of the bottleneck pose. Algorithm 1 summarises the overall training procedure.

Algorithm 1: Training algorithm.

```
Collect one demo, record initial pose and velocities $\mathcal{U}$
Set bottleneck to initial pose with vertical orientation
Set $R$ to rotation between bottleneck and initial pose
Set $h$ to height of bottleneck
Initialise empty dataset $\mathcal{D}$
for each trajectory do
    Move robot to random starting pose
    Sample random target pose near to bottleneck
    while robot is above $h$ do
        Capture image $I$ and pose $T_{EB}$
        Store $(I, T_{EB})$ in $\mathcal{D}$
        Move robot towards target pose along linear path
Collect dataset $\mathcal{E}$
Train $f(I)$ on $\mathcal{D}$
Train $g(I)$ on $\mathcal{E}$
```

D. Sequential State Estimation

During the approach trajectory, we assume that the object is stationary. Therefore, every prediction of the bottleneck pose is a prediction of the same value. So rather than only using the prediction from the image at the current time step, we can also consider fusing together multiple sequential predictions. First, let us denote $\hat{x}_t$ as 3-dimensional vector representing the horizontal position and vertical orientation of the predicted bottleneck pose at time t, such that $\hat{x}_t \equiv f(I_t)$. We also assume a Gaussian uncertainty for this vector, represented as a standard deviation $\hat{\sigma}_t$, which will be discussed below. Then, let us denote $\bar{x}_t$ as the estimate returned by our sequential estimation method, which depends on predictions $\hat{x}_0 \dots \hat{x}_t$. This is also assigned a Gaussian uncertainty $\bar{\sigma}_t$. We now propose two simple methods to calculate $\bar{x}_t$: firstly, by averaging the individual predictions as a *Batch*, and secondly, by *Filtering* the individual predictions.

Batch. In batch estimation, $\bar{x}_t$ is re-estimated at each time step using all previous individual predictions, $\hat{x}_0 \dots \hat{x}_t$. Batch estimation then uses inverse-variance weighting [35] to combine each $\hat{x}$, which is the optimal estimator under Gaussian uncertainty σ. The estimate is calculated by:

$$\bar{x}_t = \frac{\sum_{\tau=0}^{\tau=t} \hat{x}_\tau / \hat{\sigma}_\tau^2}{\sum_{\tau=0}^{\tau=t} 1/\hat{\sigma}_\tau^2}. \tag{2}$$

Filtering. In estimation via filtering, $\bar{x}_t$ is re-estimated at each time step using the estimate from the previous time step, $\bar{x}_{t-1}$, and the prediction at the current time step, $\hat{x}_t$, together with the associated uncertainties:

$$\bar{x}_t = \frac{(\bar{x}_{t-1}/\bar{\sigma}_{t-1}^2) + (\hat{x}_t/\hat{\sigma}_t^2)}{(1/\bar{\sigma}_{t-1}^2) + (1/\hat{\sigma}_t^2)}, \tag{3}$$

$$\bar{\sigma}_t^2 = \frac{1}{(1/\bar{\sigma}_{t-1}^2) + (1/\hat{\sigma}_t^2)}, \tag{4}$$

where the initial $\bar{x}_0$ is set to $\hat{x}_0$, and the initial $\bar{\sigma}_0$ is set to the *Prior* uncertainty, defined below. This is a form of Bayes filter [36], and is derived from a Kalman filter with zero process noise, since we assume that kinematic errors are insignificant relative to the uncertainty in $f(I)$'s predictions.

Estimating Uncertainty. Both the *Batch* and *Filtering* sequential estimation methods require a Gaussian uncertainty $\hat{\sigma}$ associated with each prediction $\hat{x}$. In our experiments, we investigated three approaches to calculating $\hat{\sigma}$. The first, is to use Dropout to model $f(I)$ as an approximate Bayesian neural network [37], where we retain Dropout during inference, and set $\hat{\sigma}$ to the standard deviation of the predictions. The second, is to train a function which predicts $\hat{\sigma}$ given $\hat{x}$, in an attempt to capture any dependency between state and uncertainty. For example, it was observed that images captured at a greater distance from the object generally resulted in greater prediction uncertainty. To train this, we took the validation subset of $\mathcal{D}$ used for early stopping of $f(I)$, computed predicted poses for all images, and calculated their errors. Then we trained a small neural network to predict the errors from the predicted poses. The third, is to use $f(I)$'s validation error as a constant, prior uncertainty for all predictions. We refer to these methods in later experiments as *Dropout*, *Predicted*, and *Prior*, respectively.

E. Task Execution

For testing, the robot executes the approach trajectory by moving along a linear path towards the latest estimate of the bottleneck pose. This pose T_{RB} is calculated by:

$$T_{RB} = T_{RE}\ T_{EB}, \tag{5}$$

where T_{RE} is calculated using forward kinematics, and T_{EB} is the prediction from $f(I)$. When the end-effector reaches the bottleneck's height, $g(I)$ is used to calculate a final estimate of T_{RB}. The robot then moves to this pose, and executes the interaction trajectory by repeating the demonstration's end-effector velocities. Algorithm 2 summarises the overall testing procedure.

Algorithm 2: Testing algorithm.

```
Retrieve $\mathcal{U}$, $R$, $h$, $f(I)$, and $g(I)$ from Algorithm 1
while robot is above $h$ do
    Capture image $I$
    Predict bottleneck using $f(I)$
    Update bottleneck with sequential state estimation
    Move robot towards bottleneck along linear path
Predict bottleneck using $g(I)$
Move robot to bottleneck
Rotate robot by $R$
Execute velocities $\mathcal{U}$
```

IV. Experiments

A. Implementation Details

We used a basic convolutional neural network for both $f(I)$ and $g(I)$, consisting of four convolutional layers with ReLU and max-pool downsampling, followed by four fully-connected layers with ReLU and Dropout. A mean squared-error loss function was used, with coefficients to balance translation and rotation determined using [38], which was calculated using a small dataset and then fixed for all experiments. Early stopping was done with a 20% validation subset. Real-world experiments used a 7-DOF Sawyer robot operating at 30 Hz, with a camera capturing 64-by-64 RGB images. We found that as the camera is very close to the object by the end of the approach trajectory, higher resolution images were not necessary. The task space was a 40-by-40 cm region on a table below the robot. During training, the object was placed at the centre of the task space, and the robot's approach trajectories began at 50 cm above the table, with initial horizontal positions uniformly sampled from the task space, and initial orientations uniformly sampled over a range of 90 degrees. During each test, the object was placed randomly in the task space, and over a range of 90 degrees relative to its pose during training, and the robot's approach trajectories began at 50 cm above the centre of the task space. For the trajectories used to train $f(I)$, we uniformly sampled a target pose over ranges of 5 cm and 5° relative to the bottleneck, with ranges of 2 cm and 20° for $g(I)$.

B. Target Reaching

We first evaluate a range of methods for bottleneck pose estimation using $f(I)$, including the sequential methods described in Section III-D. We created a target reaching task to test the approach trajectory, where the end-effector was first manually moved to a random point above an object to define an artificial bottleneck, with the task then being to estimate that pose and move to it through the approach trajectory. We performed experiments with 7 random objects (different to those for the later imitation learning experiments), using 50 trajectories on each to collect $\mathcal{D}$. We tested each object in 5 different poses, each time setting the end-effector by eye as close to the artificial bottleneck as possible, to record the ground-truth bottleneck pose.

Methods. We implemented 11 different methods for estimating the bottleneck pose. *Oracle* uses the ground-truth bottleneck to reveal the unavoidable error in the position controller. *First Image* deploys $f(I)$ only for the first image in the trajectory, and uses that same estimate throughout the trajectory. *Best Image* continually deploys $f(I)$ for all images, and uses the estimate with the lowest uncertainty from all images captured so far (using *Dropout* or *Predicted*). *Visual Servoing* continually deploys $f(I)$ for all images, and uses the estimate only for the current image without performing sequential estimation. *Batch* and *Filtering* are described in Section III-D, and perform sequential estimation by fusing predictions from multiple images along the trajectory (using *Dropout*, *Predicted*, or *Prior*).

	Mean Error		Min Error		Max Error	
Estimation Method	Pos	Ori	Pos	Ori	Pos	Ori
Oracle	0.3	0.0	0.1	0.0	0.7	0.1
First Image	13.9	9.0	3.7	**1.3**	25.7	17.7
Best Image (Dropout)	<u>**4.9**</u>	6.4	**2.3**	2.5	<u>**8.2**</u>	**10.7**
Best Image (Predicted)	13.6	12.0	5.5	4.6	24.9	24.2
Visual Servoing	5.7	6.6	3.4	3.1	**8.9**	11.3
Batch (Prior)	**5.2**	**5.8**	<u>**1.9**</u>	2.0	9.8	<u>**10.4**</u>
Batch (Dropout)	7.5	**5.9**	2.9	<u>**1.2**</u>	12.7	12.1
Batch (Predicted)	5.8	6.4	2.3	2.2	10.2	11.3
Filtering (Prior)	**5.2**	<u>**5.7**</u>	**1.9**	1.8	**9.5**	**10.6**
Filtering (Dropout)	7.6	6.2	3.1	**1.7**	12.5	12.7
Filtering (Predicted)	5.8	6.9	2.5	2.3	11.0	12.8

TABLE I: Target reaching results for the approach trajectory, for different bottleneck pose estimation methods. Mean, min and max errors were calculated across 5 poses per object, then averaged across all 7 objects. Positions are in mm, orientations in degrees. The three best results per column (excluding *Oracle*) are in bold, with the best also underlined.

Results. Table I shows the results, which reveal three insights. Firstly, the best methods for fusing multiple predictions typically outperformed basic visual servoing. This demonstrates the effectiveness of classical, explicit state estimation over implicit state estimation via deep learning, and supports our proposal to model imitation learning as a state estimation problem. Secondly, methods which assume a constant uncertainty for all predictions (*Prior*) typically outperformed those which estimate uncertainty dynamically (*Dropout* and *Predicted*). Since the formulations in Section III-D assume Gaussian uncertainties, we can conclude that this assumption introduces a harmful bias since estimating Gaussian uncertainties is challenging with deep learning. Thirdly, filtering methods performed at least as well as batch methods despite discarding past data, which could be explained by the bias towards more recent images captured closer to the object, which are typically more discriminative.

C. Imitation Learning

The target reaching experiments evaluated the approach trajectory, and we now evaluate our full framework for imitation learning, by introducing human demonstrations and the interaction trajectory. We used 50 trajectories for collecting each of $\mathcal{D}$ and $\mathcal{E}$. A video is available at: **www.robot-learning.uk/coarse-to-fine-imitation-learning**.

Tasks. We chose 8 everyday tasks which reflect a range of different challenges, as shown in Figure 3. *Bottle* requires a lid to be placed onto a bottle. *Plate* requires a plate to be inserted into a specific slot in a rack. *Screwdriver* requires a toy screwdriver to be inserted into a screw in a toy aeroplane, and then twisted by 90 degrees. *Lid* requires a lid on a box to be lifted upright. *Knife* requires a wooden knife to be inserted into a knife rack. *Hammer* requires a toy nail to be knocked into a hole using a toy hammer. *Scoop* requires a small bag to be scooped up. *Plug* requires an electrical plug to be inserted into a socket. For each task, specific success criteria were defined based on the object configuration at the end of the robot's attempt, which were then judged by eye.

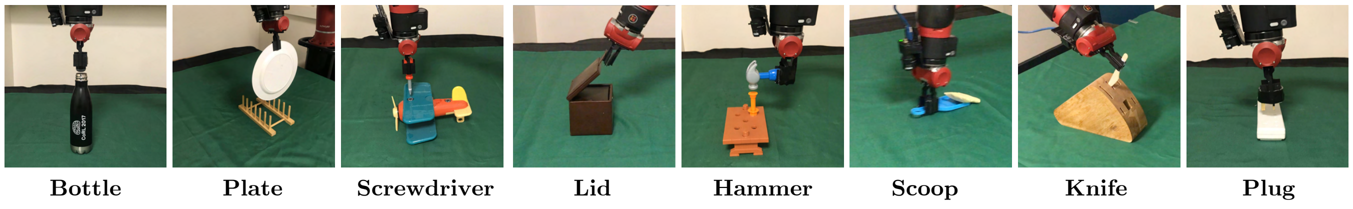

Fig. 3: The set of 8 everyday tasks we used for real-world imitation learning experiments.

Method	Bottle	Plate	Screwdriver	Lid	Hammer	Scoop	Knife	Plug	Average
Visual Servoing	65	25	20	65	35	55	**15**	5	35.6
Visual Servoing + Correction	**100**	**95**	**70**	**100**	40	95	10	10	65.0
Filtering	85	25	10	85	50	90	0	10	44.4
Filtering + Correction	**100**	80	60	**100**	**65**	**100**	10	**45**	**70.0**

TABLE II: Real-world imitation learning results, showing % success rate over 20 object poses for each task. The best result per column is in bold.

Methods. We tested two different methods for the coarse trajectory, each with and without the last-inch correction. First, we evaluated using only the current image to estimate the bottleneck pose at each time step, denoted *Visual Servoing*. Second, we evaluated using *Filtering* with *Prior* uncertainty, since on average this performed at least as well as any other method in the target reaching experiments. We also implemented a method based on residual reinforcement learning following [19], using the negative $\ell 1$ distance between the current image and the demonstration's final image as a dense reward, and with the baseline policy trained with behavioural cloning on the single demonstration. However, we found that even with hyper-parameter tuning, this was not able to solve any of the tasks. Whilst this method has previously performed well for insertion tasks using multiple demonstrations [19], we found that a single demonstration was not sufficient to constrain exploration, and the $\ell 1$ image distance was often not a meaningful signal for our tasks. We did not compare to behavioural cloning, as this requires more than one demonstration, nor meta-learning, as current methods require prior training on similar tasks.

Results. Table II shows the results for our imitation learning experiments, with 20 object poses tested per task. We see that for the full implementation, using both filtering and the last-inch correction, we are able to learn a range of everyday tasks with encouraging success rates, given that only a single demonstration is required. Three of the tasks were successfully executed in all 20 of the tested object poses. As with the target reaching experiments, sequential state estimation proves to be superior to basic visual servoing. We also see the benefit of the last-inch correction, which significantly increased success rates. Note that since the bottleneck pose estimation is self-supervised, our method can automatically improve in performance with more data collection, without requiring further human demonstrations.

We also notice a large variation in performance across the different tasks. Task difficulty can be defined by two orthogonal properties: first, the required precision to complete the task, and second, the suitability of the object's shape and texture for 2D image-based pose estimation. For example, the *Knife* task was very challenging because not only is the slot in the rack very thin, but the rack itself lacks texture, and appears in an image as just a region of relatively uniform colour, making orientation prediction particularly difficult. The *Plug* task was also challenging, since this is contact-rich and requires significant force to overcome resistance. However, our full method solved this in 9 out of 20 poses of the socket, which is surprisingly high considering only a simple velocity controller is used for contact-rich insertion. However, had the socket not been free to compliantly slide along the table during interaction, the success rates would likely have been lower. Another important observation is how well our method performed on the *Hammer* and *Scoop* tasks. These both require accurate motion at high velocities during object interaction, and are thus well suited to our method of directly replicating the demonstration's velocities. This interaction would be challenging to achieve via explicit policy learning, such as reinforcement learning.

V. Conclusions

We have proposed and evaluated Coarse-to-Fine Imitation Learning, which models a robot manipulation task as a coarse, approach trajectory, followed by a fine, interaction trajectory, and models imitation learning as a state estimation problem. We have shown that this can learn a range of novel, real-world, everyday tasks, from just a single human demonstration, whereas existing methods typically require either multiple demonstrations, repeated environment resetting, or prior task knowledge. A key component is the ability to simply repeat the original demonstration's velocities, enabling convenient acquisition of complex motion, without needing to explicitly learn a policy. Furthermore, the controller is analytical, stable, and interpretable, which typically cannot be said of many visual imitation learning methods today based on black-box, end-to-end policy learning. Opportunities for future work include improving the bottleneck pose estimator with 3D computer vision, and introducing closed-loop control during the interaction trajectory.

REFERENCES

[1] T. Zhang, Z. McCarthy, O. Jowl, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[2] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[3] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," in *Conference on Robot Learning (CoRL)*, 2017.

[4] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[5] P. Englert and M. Toussaint, "Learning manipulation skills from a single demonstration," *The International Journal of Robotics Research*, 2018.

[6] Y. Y. Tsai, H. Xu, Z. Ding, C. Zhang, E. Johns, and B. Huang, "DROID: Minimizing the reality gap using single-shot human demonstration," *IEEE Robotics and Automation Letters (RA-L)*, 2021.

[7] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[8] P. Florence, L. Manuelli, and R. Tedrake, "Self-supervised correspondence in visuomotor policy learning," *IEEE Robotics and Automation Letters (RA-L)*, 2019.

[9] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto, "Visual imitation made easy," in *Conference on Robot Learning (CoRL)*, 2020.

[10] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Conference on Robot Learning (CoRL)*, 2017.

[11] N. Di Palo and E. Johns, "SAFARI: Safe and active robot imitation learning with imagination," *arXiv preprint arXiv:2011.09586*, 2020.

[12] S. Paradis, M. Hwang, B. Thananjeyan, J. Ichnowski, D. Seita, D. Fer, T. Low, J. E. Gonzalez, and K. Goldberg, "Intermittent visual servoing: Efficiently learning policies robust to instrument changes for high-precision surgical manipulation," *arXiv preprint arXiv:2011.06163*, 2020.

[13] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *Robotics: Science and Systems (RSS)*, 2018.

[14] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[15] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual policy learning," *arXiv preprint arXiv:1812.06298*, 2018.

[16] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[17] B. Thananjeyan, A. Balakrishna, U. Rosolia, F. Li, R. McAllister, J. E. Gonzalez, S. Levine, F. Borrelli, and K. Goldberg, "Safety augmented value estimation from demonstrations (SAVED): Safe deep model-based RL for sparse cost robotic tasks," *IEEE Robotics and Automation Letters (RA-L)*, 2020.

[18] Y.-Y. Tsai, B. Xiao, E. Johns, and G.-Z. Yang, "Constrained space optimization and reinforcement learning for complex tasks," *IEEE Robotics and Automation Letters (RA-L)*, 2020.

[19] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solowjow, and S. Levine, "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[20] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International Conference on Machine Learning (ICML)*, 2016.

[21] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, "End-to-end robotic reinforcement learning without reward engineering," in *Robotics: Science and Systems (RSS)*, 2019.

[22] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[23] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[24] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. Freitas, and N. Heess, "Reinforcement and imitation learning for diverse visuomotor skills," in *Robotics: Science and Systems (RSS)*, 2018.

[25] Y. Schroecker, M. Vecerik, and J. Scholz, "Generative predecessor models for sample-efficient imitation learning," in *International Conference on Learning Representations (ICLR)*, 2019.

[26] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[27] M. Sieb, Z. Xian, A. Huang, O. Kroemer, and K. Fragkiadaki, "Graph-structured visual imitation," in *Conference on Robot Learning (CoRL)*, 2020.

[28] Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[29] S. James, M. Bloesch, and A. J. Davison, "Task-embedded control networks for few-shot imitation learning," in *Conference on Robot Learning (CoRL)*, 2018.

[30] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning (ICML)*, 2017.

[31] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine, "One-shot imitation from observing humans via domain-adaptive meta-learning," in *Robotics: Science and Systems (RSS)*, 2018.

[32] C. Yu, Z. Cai, H. Pham, and Q.-C. Pham, "Siamese convolutional neural network for sub-millimeter-accurate camera pose estimation and visual servoing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[33] E. Y. Puang, K. Peng Tee, and W. Jing, "KOVIS: Keypoint-based visual servoing with zero-shot sim-to-real transfer for robotics manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[34] M. Argus, L. Hermann, J. Long, and T. Brox, "Flowcontrol: Optical flow based visual servoing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[35] J. Hartung, G. Knapp, and B. K. Sinha, *Statistical meta-analysis with applications*. John Wiley & Sons, 2011.

[36] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, 2002.

[37] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning (ICML)*, 2016.

[38] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A convolutional network for real-time 6-DOF camera relocalization," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.