## A Algorithm 1 Technical Details

**Counterfactual Stop Gradients** $\hat{\delta}_\emptyset$ is included in the objective function twice. To balance terms, the stop-gradient operator (assigning gradients to zero) can be used on $\hat{\delta}_\emptyset$ in the estimate of reafference. This may help to stabilise training, although we have not observed a substantial difference with/without stop-gradients in our experiments.

**Zeroing Reafference** The estimated reafferent effect for the null action should be zero. At the initial stages of training the estimate will be non-zero and small errors later may impact performance. A simple trick that may improve training time and overall performance is simply to set $\hat{\delta}_a = 0$ where $a_i = \emptyset$ in the mini-batch. If interpolation between actions is required (for example, with continuous actions) this may be a detriment to performance since some gradient information is lost and the additional freedom in the estimate at $\emptyset$ might impact neighbouring actions.

**Variance of Reafference Estimates** Since reafference is computed from two estimates the variance of its estimate is comparatively large. When reafferent effects are small, the additional epistemic uncertainty can have a proportionally significant impact on the estimate. This is a limitation of our approach, exploring it further is left as future work.

## B Environment & Training Details

### B.1 Cartpole

**Environment Details** This environment is a modified version of the `Carpole-v1` environment provided by OpenAI gym (Brockman et al. 2016). The environment is modified to contain the null-action, and is made more interesting by allowing the pole to fall further before resetting the environment. The agent may take one of three actions: $\{-\beta, 0, \beta\}$, each applies a force to the cart (0 applies no force). The agent's observation contains the cart position and velocity, and the poles angle and angular velocity. The forces on the cart/pole are not observed. The SCM for the agent's observation is given below:

$$\begin{aligned}
A_t &:= \pi(\theta) \\
Y_{t+1} &:= Y_t + dt\dot{Y}_t \\
\dot{Y}_{t+1} &:= \dot{Y}_t + dt\ddot{Y}_t \\
\theta_{t+1} &:= \theta_t + dt\dot{\theta}_t \\
\ddot{Y}_{t+1} &:= \gamma_t + M_p\ddot{\theta}_t cos(\theta_t)/(M_p + M_c) \\
\dot{\theta}_{t+1} &:= \dot{\theta}_t + dt\ddot{\theta}_t \\
\ddot{\theta}_{t+1} &:= (Gsin(\theta_t) - cos(\theta_t)\gamma_t)/C_t \\
\gamma_t &:= (A_t + M_p\dot{\theta}_t^2 sin(\theta_t))/(M_p + M_c) \\
C_t &:= L(4/3 - M_p cos(\theta_t)^2/(M_p + M_c)
\end{aligned}$$

Where $Y$ is the cart position, $\theta$ is the pole angle, $\dot{\square}$ indicates a derivative (velocity, acceleration). $M_p$ and $M_c$ are the masses of the pole and cart respectively, $L$ is the length of the pole, $G$ is the gravitational acceleration. $\gamma_t$ and $C_t$ are placeholder variables that do not need to be included in the causal graph. The agent observes $X_t = (Y_t, \dot{Y}_t, \theta_t, \dot{\theta}_t)$.

**Training Details** The model trained is a 4 layer MLP with tanh activation with approx. 500k parameters. The action is represented as a one-hot vector and is combined with the observation in the initial network layer. The model was trained for 100 epochs on 100k examples that were collected using a uniform random policy and stored. The Adam optimiser with a learning rate of 0.0005 was used.

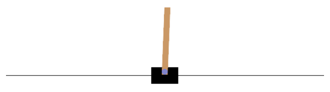

Figure 7: Cartpole environment.

**Semi-Implicit Euler Cartpole** The kinematics integrator used in the original environment updates $Y_t$ and $\theta_t$ given the previous values of the higher-order terms. Actions therefore only immediately affect the higher-order terms. In the semi-implicit Euler version of Cartpole the SCM is modified as follows:

$$\begin{aligned}
Y_{t+1} &:= Y_t + dt\dot{Y}_{t+1} \\
\theta_{t+1} &:= \theta_t + dt\dot{\theta}_{t+1}
\end{aligned}$$

With these modifications, actions now have an immediate effect on the cart position and pole angle. Experiment (i.1) was rerun with this new environment, changes in the angle and position were modelled correctly.

### B.2 Atari Freeway

**Environment Details** This environment is a modified version of the `FreewayDeterminstic-v4` provided by OpenAI gym. The agent's observation is an $3 \times 84 \times 84$ colour image, a cropped version of the original observation. The agent may take one of three actions: $\{$`forward`, `backward`, $\emptyset\}$, moving the chicken forward, backwards or keeping it stationary.

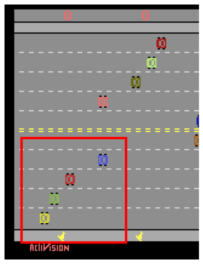

Figure 8: Freeway environment, the red rectangle shows the agent's observation used in our experiments.

**Training Details** The model trained follows the UNet architecture (Ronneberger, Fischer, and Brox 2015). We found that this architecture worked better than those without residual connections. Actions are embedded using a single linear layer, then introduced into the UNet by an element-wise product with the output of the encoder portion of the network. The network has approx. 33M parameters. It was trained for 50 epochs on 5k observations/actions collected using a uniform random policy and stored. The Adam optimiser with a learning rate 0.0005 was used.

This environment is not Markovian, which leads to some issues with prediction at certain points. Namely, when the chicken gets hit by a car, the agent's actions become ineffective and the agent moves backward some steps. The model will predict the usual reafferent effect if the forward/backward action is taken as there is no indication that the chicken has previously been hit in the current observation. This can be resolved by using frame stacking, or by introducing some kind of memory (e.g. with LSTM). The issue is not one that is relevant for demonstrating the effectiveness of our approach.

### B.3 Artificial Ape

**Environment Details** This environment was developed using the World of Bugs platform (Wilkins and Stathis 2022) in the Unity3D game engine. The agent can rotate its view left or right, or remain looking it its current direction (null-action). The view shows a scene from a first person perspective, the observation is an image $(1 \times 64 \times 64)$ pixels. A change in view due to action is the only reafferent effect. The agent sits atop a platform which similarly rotates left, right, or not at all, this rotation also changes the view of the agent. There are a number of chequered cubes that move perpendicular to the motion of the view (up/down). Both the change in view caused by the platforms rotation, and the movement of the cubes are exafferent effects on the agent's observation. In the first experiment we perform, the platform is kept stationary, and the goal is to disentangle to moving cubes from shifting view. In the second experiment, the platform is randomly rotating.

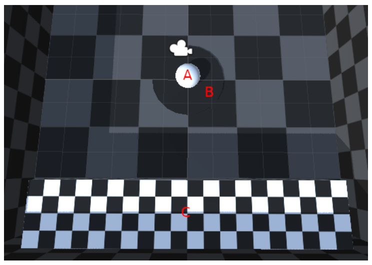

Figure 9: Artificial Ape environment from a birds-eye perspective. Objects are labelled, A is the agent, B is the platform, and C is the collection of moving cubes.

**Training Details** The model trained again uses the UNet architecture following that used in Freeway, this time with approx. 10M parameters. It was trained for 50 epochs on 5k observations/actions collected using a uniform random policy and stored. The Adam optimiser with a learning rate 0.0005 was used. The dataset used is publicly available[2].

## C Additional Experiments

### C.1 Adaptivity in Cartpole

Biological agent's experience changes in their motor system over their lifetime, for example, through growth or disease. The same action will consistently give rise to different effects at different times; any model of reafference needs to be capable of adapting to a changing motor system.

We perform a simple experiment so show that Alg. 1 is able to adapt to changes in the agent's body by continued training. This is demonstrated in the Cartpole environment, where the cart and pole is considered the body of the agent, by changing the length of the pole during training. Results are presented in Fig. 10. The model is able to quickly adapt to changes in the reafferent effect and properly recover both reafference and exafference after a short period of retraining.

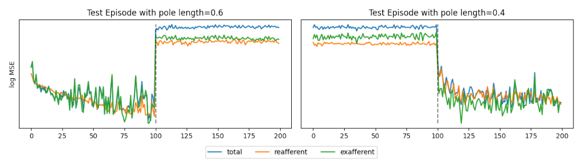

Figure 10: Demonstration of adaptivity in the Cartpole environment. Treating the cart and pole as the agent's body, the length of the pole $l$ is changed during training to simulate changes that a biological agent might undergo. Changing $l$ changes the dynamics of the system, both reafferent and exafferent effects change as a result and should be relearned by the agent. Each graph shows the performance on a test episode where estimates are compared to the ground truth for differing $l$. Training starts with $l = 0.6$ and is changed $l = 0.4$ after 100 epochs. During the phase where training $l$ matches test $l$ the error is minimised, otherwise the model performs poorly as is expected. The forward model is able to quickly adapt (see right) its estimates after a change in the agent's body.

### C.2 Doing Something in Cartpole

In environments where the do-nothing action is not available and where it is possible to intervene on the environments mechanism, it may still be possible to recover the reafferent effects. This situation is analogous to the bi-pedal robot example reviewed in the introduction and can be seen as a kind of inverse of our approach.

In the Cartpole environment, it turns out that the required interventions are $do(G = 0)$, $do(\dot{X} = 0)$, $do(\dot{\theta} = 0)$, these can be derived from the SCM (see section B.1), which is in turn derived from the dynamical equations that described the physical system (see (Florian 2007)). With knowledge of the SCM, the reafferent effect can be isolated by zeroing out additive terms that do not depend on the action. With

[2]https://github.com/BenedictWilkins/disentangling-reafference

these interventions, the reafferent effect can be estimated as the total effect. Training the reafferent forward model on the intervened mechanism, exafference is then estimated as any error when testing without intervention. This mechanism is closer to the view of reafference illustrated in Fig. 1.

Intervention on the environment mechanism can drastically impact how the environment evolves, some states may be difficult (or impossible) to reach using just the agent's action. This issue can be seen in the Cartpole environment, a random policy is unlikely to reach regions of the state space that are otherwise commonly visited without environment intervention. It can be somewhat avoided if the intervention is intermittent. Intermittence may be difficult to achieve in some environments, and the choice of when to intervene to obtain the most information regarding causal effects is highly non-trivial.

### C.3 Estimating Average Reafferent Effects

Estimating the Average Reafferent Effect (ARE) can be useful in some settings. Particularly for those in which the effect is similar (or constant) across observations. An example might be movement in a simple 2D world. If the action has the same effect in each state (e.g. move 1 unit in some direction) and is independent of the agent's position, then averaging over all positions will give a better estimate of this effect in small sample size regimes. Alg. 1 is used to estimate the ARE in a simple environment in which reafference is the same (on average) for each observation. The SCM is given as follows:

$$\begin{aligned} A &:= \pi(\{-1,0,1\}) \\ X &:= U_X \\ X' &:= X^2 + A \cdot U_{X'} \end{aligned}$$

where $\pi$ is a discrete uniformly random decision variable that selects actions from the set {-1,0,1}, and $U_X$ and $U_{X'}$ are continuous uniformly random variables on the interval [0,2]. A simple 2 layer MLP was trained using Alg. 1 to estimate the effects using 1000 sample observations. The ARE is estimated as 0.951 for $A = 1$, 0 for $A = 0$, and $-1.0151$ for $A = -1$. The average exafference effect is estimated as 0.31741. Results are shown in Fig. 11

### C.4 Artificial Ape without an Indicator

If no indicator is present in the Artificial Ape environment, this the one presented in experiment (iii.2) then aleatoric uncertainty is high and the estimates will be difficult to interpret. The distribution of exafferent view shifts is wide without an indicator and is being approximated by its expectation, leading to blurry results. It is not that the estimate would not be useful to an agent, only that it is difficult for us to interpret what has been learned. Results experiment (iii.2) without an indicator can be found in Fig. 13.

A similar experiment can be run in a 2D grid world in which the agent is *pushed* up/down at random, this is analogous to the random rotation of the platform in Artificial Ape. Since the environment is simpler it is easier to interpret. Additionally, as the environment is simple, the model can reach close to perfect performance meaning no epistemic artefacts. The result of this experiment is presented in Fig. 12.

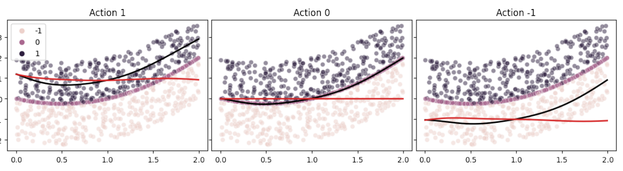

Figure 11: Example of estimating the ARE where action effects are the same (in expectation). The plots show the observed effects $x' - x$ (y-axis) plotted again the observation $x$ (x-axis). Each graph shows estimates for a particular action. The red lines shows the estimated reafferent effect for different values of $x$. It remains constant and so can be averaged over to get an estimate of the ARE that is useful. The black line shows the total effect, the upward trend is the exafferent effect.

## D Miscellaneous

### D.1 Empty Space Example Calculations

The average reafferent effect of action 1 in the example given in section 3.2 can be computed as follows:

$$\begin{aligned} \delta_{Y'}(1) &= \mathbb{E}[Y'(1)] - \mathbb{E}[Y'(0)] \\ &= [\mathbb{E}[A \cdot Z | do(A=1)] + \mathbb{E}[Y]] \\ &\quad - [\mathbb{E}[A \cdot Z | do(A=0)] + \mathbb{E}[Y]] \\ &= 1 \cdot Pr(Z=1) - 0 \cdot Pr(Z=1) \\ &= Pr(Z=1) = p_z \end{aligned}$$

The reafferent effect is similar, except as we are conditioning on $Z$, $Pr(Z = 1)$ is 1 when $Z = 1$ and 0 when $Z = 0$ leading to the stated result. Note that in the example there is no confounding since the agent is essentially performing a randomised trail, the action $A$ is independent.

## E Reproducability Checklist

**Compute Requirements** All experiments were run on a single 12 cpu core 32gb RAM machine with an NVIDIA GeForce RTX 2070 GPU. Experiments can quickly and easily be reproduced on a mid-high end personal computer.

**Code Dependencies** Code is written in python 3.8 and models are developed with pytorch and run with CUDA. Other dependencies are specified in the `setup.py` file.

**Randomness & Seeding Environments** We have not provided the seeds used to generated data in each environment (e.g. the seed for the random policy used) or for SGD, all results can be reproduced easily without them.

**Evaluation metrics** Our results are difficult to evaluate numerically. Our work is not a *delta*, so there are no measures of performance to compare to from previous/related work.

In cases where ground truth reafference/exafference is available (Cartpole and Freeway) any metric that compares

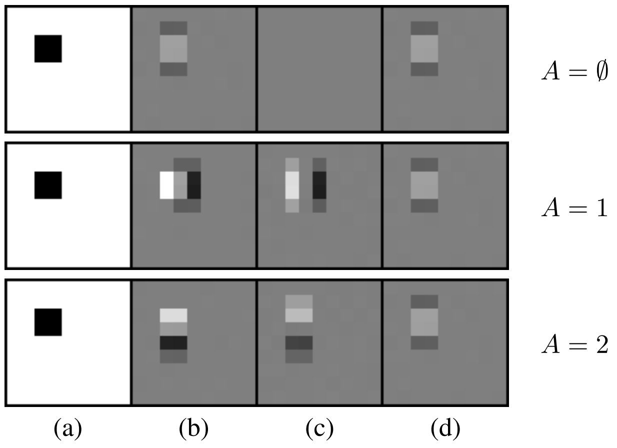

Figure 12: Disentangling in the 2D grid World environment. (a) shows the agent's observation, a 9x9 pixel image, (b) shows the estimated total effect, (c) shows the estimated reafferent effect and (d) shows the estimated exafferent effect. In this environment, the agent can take the actions `NORTH`, `EAST`, `SOUTH`, `WEST` or $\emptyset$ to move the agent (shown as a black square) in the given direction. At each step the agent is randomly pushed north/south with probability 0.25, or remains it its current location with probability 0.5. This leads to the exafferent effect seen with action $A = \emptyset$. For actions $A = 1$ (`EAST`) and $A = 2$ (`SOUTH`), the reafferent effect is modelled with this random perturbation in mind. Since the most likely outcome is an absence of noise, for $A = 1$ the most likely location the agent will end up in is eastward, with less chance of ending up north or south eastward (indicated by the fainter values). The case is similar for $A = 2$. The hope is that this gives some intuition about what is actually being learned by Alg. 1. See also Fig. 13.

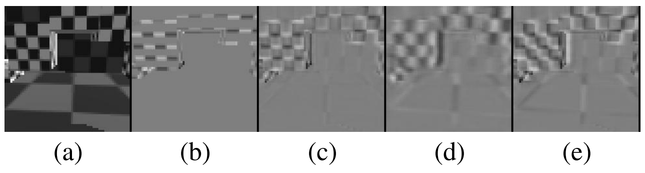

Figure 13: Disentangling in Artificial Ape with congruent reafference and exafference without an the platform as an indicator. (a) observation (b) ground truth total effect (c) estimated total effect (d) estimated reafferent effect (e) estimated exafferent effect. As in experiment (iii.2) in the main body of the paper, the agent and platform have rotated in opposite directions, leading to a cancellation in the total effect. The estimate of the total effect does not match the ground truth because of the aleatoric uncertainty in this environment. In cases such as these it might be better to use a generative model rather than approximating with expectations as is done here. This is left as a direction for future work.

estimates to the ground truth would not necessarily give a good indicator of a models ability to disentangle the two. Instead, it may just reflect whether the model was able to learn an accurate representation of the environment dynamics (this might happen if a model is not expressive enough to capture the true dynamics). We have tried to set up experiments in such a way that a single example of the disentanglement presented in the paper would give a good idea of the performance by *eye-balling*. Granted this is not ideal, but the samples presented in the paper are representative (i.e. not cherry picked for accuracy, only for clarity). Videos of the disentanglement for full runs through different environments are available in the supplementary files.

**Algorithm Stability & Hyper-Parameters** Our algorithm is very robust to hyper-parameter choice and given their limited number, we have not presented any detailed discussion of them in the main body of the paper. Our first selection of Adam with a learning rate (0.0005) was kept in all runs. Batch sizes were varied only to allow training with limited GPU memory, and did not have any noticeable on impact performance.

The most impactful hyper-parameter choice was the neural network architectures used. For Cartpole we settled on a simple MLP network (see B.1) after only a few runs which each take only a few minutes on our hardware. For Freeway and Artifical Ape, we tried various architectures, including a simple AlexNet like architecture, and after a handful of runs (less than 10) settled on the UNet architecture presented in the work (see supplementary code for the full architecture in code). The simpler architectures we tried did not well capture the environment dynamics, or were slower to train. A single training run took in the order of tens of minutes with our setup.

We tried using batch normalisation in a few runs to the detriment of the exafference estimates as they became nonsensical, this needs further investigation.

## References

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym.

Florian, R. V. 2007. Correct equations for the dynamics of the cart-pole system. 6.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 234–241. Springer.

Wilkins, B.; and Stathis, K. 2022. World of Bugs: A Platform for Automated Bug Detection in 3D Video Games. In *2022 IEEE Conference on Games (CoG)*, 520–523.