A Robotics Simulation

A.1 Physics Engine

Physics-based simulation has played a vital role in the field of robotics, enabling rapid prototyping, testing and experimentation. The accuracy and speed of simulation has scaled with available computation. Modern methods such as RL have leveraged this to simulate very large data sets that are otherwise impractical to collect from the real world, hence far more complex and general behaviours can be learnt. The majority of simulated robotics has focused on rigid-body dynamics and vision sensors. More recently, a range of specific environment suites have been introduced to bring simulation closer to reality or to facilitate research in an under-represented direction [1, 2, 3, 4, 5]. Collectively, these works highlight the importance of increasing the breadth and capabilities of simulation software available to researchers.

There are many choices for physics engines when developing a new suite. Here we choose the PyBullet [6] for its fast GPU rendering, support for deformable objects [7], fast and reliable kinematics and dynamics solvers, and its demonstrated sim-to-real success in robotics [8]. Importantly, PyBullet is open source and non-commercialised software which helps to improve accessibility and lower the barrier of entry for RL research. That said, the tools used here to build our tactile suite are available in other physics engines such as Mujoco, Gazebo, Nvidia-isaacgym and Unity ML agents.

A.2 Control

Throughout this work we use Cartesian velocity control where the control input is a desired velocity (twist) specified by a 6 DoF action constrained to the allowed modes of control of the task. A control rate specifies the frequency that new actions can be sent to the robot, with maximum velocity limits also imposed. For most of our experiments, we set a velocity control rate of 10 Hz, a maximum linear velocity of $10\,\mathrm{mm\,s^{-1}}$ and a maximum angular velocity of $5\,^{\circ}\,\mathrm{s^{-1}}$. When undergoing a series of predefined and random actions we notice no significant difference in Tool Centre Point (TCP) pose between simulation and the real robot.

Work-space coordinate frames are set in both simulation and reality, specific to each environment, with each action sent to the robot consisting of a Cartesian move relative to the work frame. Therefore, the learned policies can be transferred from sim-to-real without exactly replicating the simulated task; for example edges and surfaces can be placed in alternative locations provided the work frame is set correctly. As a consequence, the policy transfers even when there are notable differences in the simulation, such as mirrored arm configurations used in this work. Thus, in principle the policies could also be transferred to other robot arms, providing the same speed and frequency of control can be achieved.

B Reinforcement Learning Parameters

Near-default hyper-parameters are used in all training (full list in Table 4). Image-based observations use the Atari Nature [9] convolutional layers followed by two 256-node fully connected (FC) layers. State observations use only the FC layers. For tasks that require both image and state data, the state data is passed through two 64-node FC layers and the output concatenated with the flattened output of the convolutional layers, which is then passed through the final FC layers for action and value prediction. The convolutional weights are shared for all policy and value networks. Small random image translation augmentations help to improve performance and stabilise training, as proposed in [10, 11].

Table 4: RL and network hyperparameters.

	Param	Value
Feature Extractor	Input dim	[128, 128, C]
	Conv filters	[32, 64, 64]
	Kernel widths	[8, 4, 3]
	Strides	[4, 2, 1]
	Pooling	None
	Output dim	512
	State Encoder	[64, 64]
	Activation	*ReLU*
	Initialiser	*Orthoganal*
RL Net	Policy	[256, 256]
	Value	[256, 256]
	Activation	*Tanh*
PPO parameters	Learning Rate	3×10^{-4}
	n/ Envs	10
	Epoch steps	2048
	Batch size	64
	n/ Epochs	10
	Discount (γ)	0.95
	GAE lambda	0.9
	Clip range	0.2
	Entropy coeff	0.0
	VF coeff	0.5
	Max grad norm	0.5
	KL limit	0.1
	Optimiser	*Adam*

C Reinforcement Learning Environment Details

Table 5: Edge Follow environment description.

Observation	**• Env State:** {TCP pos, TCP lin vel, Goal pos, Edge ang} **• Tactile:** {Tactile Image} **• Vision:** {RGB Image} **• Vision + Tactile:** {RGB Image, Tactile Image}
Action Space	$\{x, y\}$
Reward	- (Euclidean distance from TCP to goal + perpendicular distance from TCP to edge)
Termination	• max episode length reached • euclidean distance from TCP to goal $< 1\text{cm}$
History	1 Frame.
Randomisation	• Edge randomly orientated through 360°. • Distance tip is embedded onto edge is randomly selected between 1.5mm and 3.5mm.

Table 6: Surface Follow environment description.

Observation	**• Env State:** {TCP pos, TCP orn, TCP lin vel, TCP ang vel, Goal pos, Target surface height, Target surface normal} **• Tactile:** {Tactile Image} **• Vision:** {RGB Image} **• Vision + Tactile:** {RGB Image, Tactile Image}
Action Space	$\{z, Rx, Ry\}$
Reward	-(z difference between TCP and local surface index + cosine difference between TCP normal and local surface normal)
Termination	• max episode length reached • euclidean distance from TCP to goal $< 1\text{cm}$
History	1 Frame.
Randomisation	• Surface randomly generated w/ OpenSimplex noise. • Direction of goal randomly selected from $[0°, 360°]$.

Table 7: Object Roll environment description.

Observation	**• Env State:** {TCP pos, TCP orn, TCP lin vel, TCP ang vel, Obj pos, Obj orn, Obj lin vel, Obj ang vel, Goal pos, Obj radius} **• Tactile:** {Tactile Image, Goal pos} **• Vision:** {RGB Image, Goal pos} **• Vision + Tactile:** {RGB Image, Tactile Image, Goal pos}
Action Space	$\{x, y\}$
Reward	-(euclidean distance from object to goal)
Termination	• max episode length reached • euclidean distance from object to goal $< 1\text{mm}$
History	1 Frame.
Randomisation	• Random starting position of object in TCP frame. • Random marble size between 5mm and 10mm diameter. • Random distance embedded into the sensor.

Table 8: Object Push environment description.

Observation	**• Env State:** {TCP pos, TCP orn, TCP lin vel, TCP ang vel, Obj pos, Obj orn, Obj lin vel, Obj ang vel, Goal pos, Goal orn} **• Tactile:** {Tactile Image, TCP pos, TCP orn, Goal pos, Goal orn} **• Vision:** {RGB Image, TCP pos, TCP orn, Goal pos, Goal orn} **• Vision + Tactile:** {RGB Image, Tactile Image, TCP pos, TCP orn, Goal pos, Goal orn}
Action Space	$\{y, Rz\}$
Reward	-(Euclidean distance from object to goal + cosine distance from object orn to goal orn + cosine distance from TCP normal to object normal)
Termination	• max episode length reached • Euclidean distance from object to final goal $<$ 2.5cm
History	1 Frame.
Randomisation	• Random trajectory of goals generated with OpenSimplex Noise.

Table 9: Object Balance environment description.

Observation	**• Env State:** {TCP pos, TCP orn, TCP lin vel, TCP ang vel, Obj pos, Obj orn, Obj lin vel, Obj ang vel} **• Tactile:** {Tactile Image} **• Vision:** {RGB Image} **• Vision + Tactile:** {RGB Image, Tactile Image}
Action Space	$\{x, y\}$
Reward	+1 per step
Termination	• max episode length reached • object tilts passed set angle ($35°$)
History	2 Frames.
Randomisation	• Random external force perturbation applied at start of episode.

D Full Reinforcement Learning Results

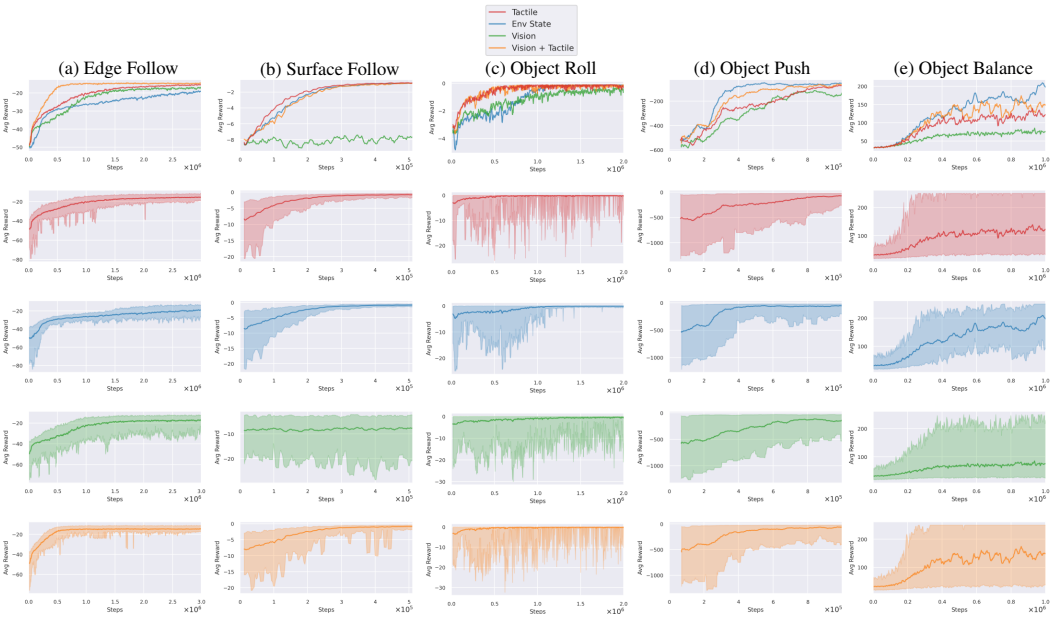

Figure 8: Full training results of reinforcement learning agents. Results are smoothed with a window size of 50 followed by averaging over 3 seeds. Shaded regions indicate maximum and minimum reward achieved over the 3 seeds (after smoothing).

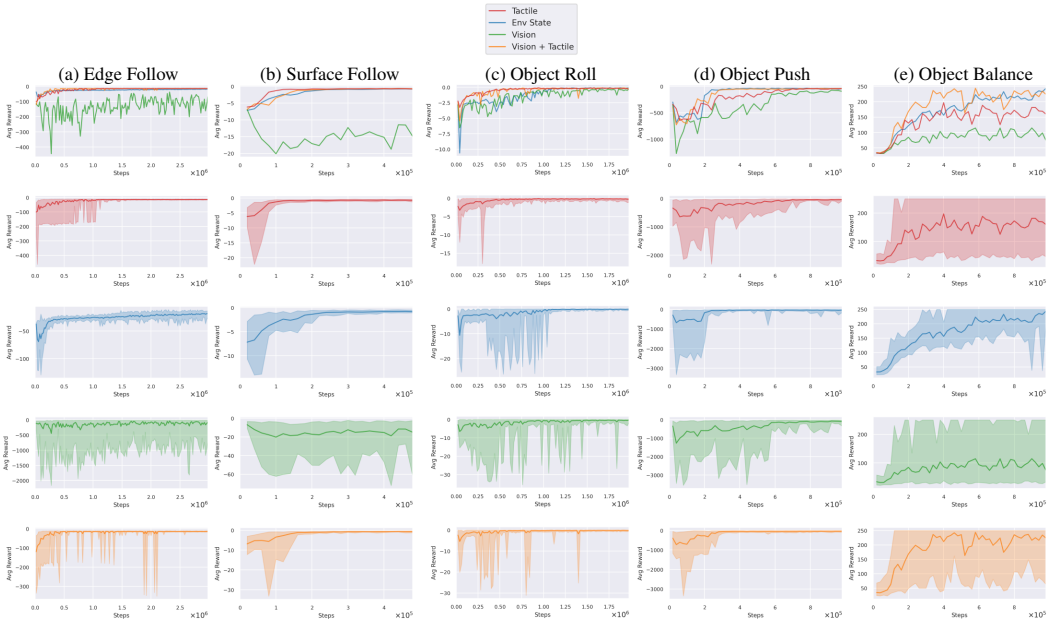

Figure 9: Full evaluation of reinforcement learning agents throughout training. 10 evaluation episodes occur every 20,000 steps, using deterministic actions. Results averaged over 3 seeds. Shaded regions indicate maximum and minimum reward achieved over the 3 seeds.

E Pix2Pix Architecture

Table 10: Pix2Pix architecture and parameters.

	Layer	Details			
Params	Batch Size	64			
	Learning Rate	0.0002			
	Image Norm	True			
	Image Trans	[2.5%, 2.5%]			
	Loss Weights	Wgan: 1.0, Wpix: 100.0]			
Generator	Input dim	[128, 128, 1]			
	Output dim	[128, 128, 1]			
		Input	**Output**	**Dropout**	**Norm**
	Down 1	1	64	None	False
	Down 2	64	128	None	True
	Down 3	128	256	None	True
	Down 4	256	512	0.5	True
	Down 5	512	512	0.5	True
	Down 6	512	512	0.5	True
	Down 7	512	512	0.5	True
	Up 1	512	512	0.5	True
	Up 2	1024	512	0.5	True
	Up 3	1024	512	0.5	True
	Up 4	1024	256	0.5	True
	Up 5	512	128	None	True
	Up 6	256	64	None	True
Discriminator	Input dim	[128, 128, 2]			
	Output dim	[16, 16, 1]			
		Input	**Output**	**Norm**	
	Disc 1	2	64	False	
	Disc 2	64	128	True	
	Disc 3	128	256	True	
	Disc 4	256	512	True	

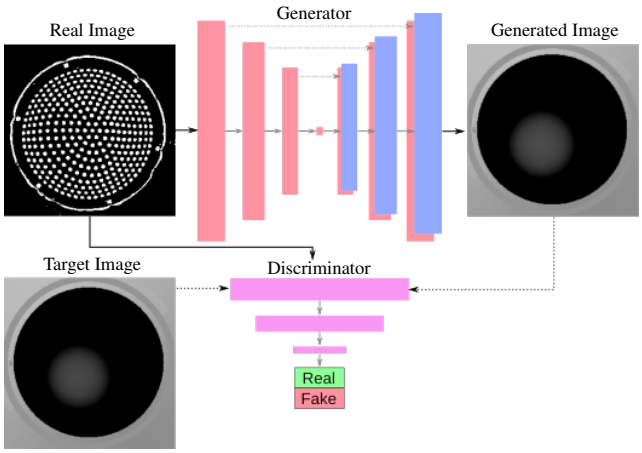

Figure 10: Real-to-sim translation of the tactile images uses a pix2pix-trained GAN. Real tactile images are processed by the generator to produce images that match the target simulated tactile images. The Discriminator is tasked with detecting whether an input tactile image pair is real or fake.

F Image Translation Data Collection

For the edge-following environment, we collect tactile images pressed onto a straight edge, varying the orientations, radial displacements and penetration of the sensor. A hemispherical sensor tip is used with the tool center point (TCP) located centrally at the end of the sensor. Relative to the TCP, the data is gathered over ranges: orientation $Rz \in [-179°, 180°]$, radial displacement $y \in [-6, 6]\text{mm}$, and penetration $z \in [3.5, 5.5]\text{mm}$.

For the surface-following and object-pushing environments, we collect tactile images pressed onto a flat surface, varying the orientations and penetration, also with a hemispherical tip. Relative to the TCP, data is gathered over ranges: orientation $\{Rx, Ry\} \in [-15°, 15°]$, and penetration $z \in [2, 5]\text{mm}$.

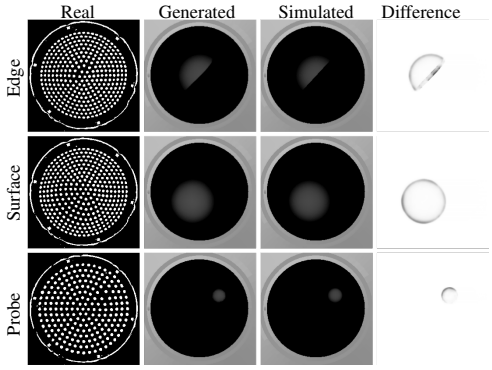

Figure 11: Image comparison between pairs of generated and simulated tactile images. Images are from validation sets for Real-to-Sim image translation. SSIM is used to create difference images.

For the object-rolling environment, we collect tactile images pressed onto a spherical probe stimulus, using a flat sensor tip appropriate to this environment. 9 spherical probe stimuli are used, ranging over $[2, 6]\text{mm}$ radius in 0.5mm increments. The sensor is positioned to contact the probe at random placements within a 15mm disk surrounding the centre of the tip. In this case, shear is not introduced into the data collection because rolling objects induce negligible motion-dependent shear.

F.1 Full SSIM Scores

We measure the SSIM scores across the validation sets collected for each task, each consisting of 2000 image pairs. For the Edge, Surface and Probe datasets respectively, we found mean scores of $[0.995, 0.992, 0.994]$, min scores of $[0.985, 0.982, 0.974]$, and max scores of $[0.999, 0.999, 0.999]$. Whilst high scores are expected due to the relatively sparse target images, this indicates strong performance in all cases.

G Supervised Learning Comparison

Table 11: Mean Absolute Error (MAE) for Task I: predicting polar angle (radians), Task II: predicting radial displacement (mm) and Task III*: predicting position of a probe (mm).

Approach	Task I	Task II	Task III*
Ours	**0.079**	**0.119**	**0.059**
Ding et al. [12]	0.254	0.45	0.73

*Indicates some difference between tasks as discussed below.

Whilst we were unable to find a sim-to-real reinforcement learning baseline that does not require specific hardware, we can compare to previous work simulating the TacTip sensor for supervised learning tasks. Ding et al. [12] used an elastic deformation approach to simulate the TacTip sensor. They focussed on supervised learning from simulated pin positions rather than tactile images, and could accurately predict edge position and orientation, and the (x, y) position of a pole pressed into the sensor. In this section we draw a comparison between the presented method on their supervised learning tasks.

The tasks are defined as follows: Task I predicts polar angle θ of an edge pressed into the sensor; Task II predicts radial displacement r of an edge pressed into the sensor; and Task III predicts the (x, y) location of the centre of a pole pressed into the sensor. More details are provided in [12] .

As this is not presented as a standardised benchmark, and we do not have access to the exact setup used, there are some differences in how these tasks have been carried out between the present work and [12]. In particular, we differ notably in Task III, where we use a flat TacTip sensor tip instead of the original hemispherical tip, because we are using the data, GANs and objects acquired for the object rolling task which needed a flat tip. That said, we expect there will only be small differences in accuracy between flat and curved TacTip tips.

For each task we first train a convolutional neural network to predict r, θ, x and y using a dataset of 10,000 simulated tactile images per task (taking $\sim$145 seconds to collect). We then collect a dataset of 2000 real sensor images and using GANs trained on edge or probe data, we translate from real-to-sim images to create the test datasets of generated images. The performance of the trained networks over the full generated datasets is then measured and compared with the results from [12].

As our simulated tactile data is comprised of images, we use a convolutional neural network (CNN). To avoid boundary issues over full rotations, we predict a sine/cosine encoding of angle. The CNN has 4 convolutional layers (each with kernel: 5, stride: 1, padding: 2, maxpool: 2) and 3 fully connected layers (dimensions = 1024, 512, output_dim). Batch normalization is applied only to the convolutional layers before the ELU [13] activation. Images of resolution 256×256-pixels were used.

Table 11 shows a large reduction in Mean Absolute Error (MAE) for both radial displacement and angle prediction in comparison with previous work simulating the TacTip [12]. A 3-fold improvement is found in predicting the angle θ (Task I) and a near 4-fold improvement when predicting radial distance (Task II). We also find a large decrease in MAE from $0.73\mathrm{mm}$ to $0.059\mathrm{mm}$ for position prediction (Task III), although as mentioned above there are some differences in setup.

References

[1] C. Chen, U. Jain, C. Schissler, S. V. A. Gari, Z. Al-Halah, V. K. Ithapu, P. Robinson, and K. Grauman. SoundSpaces: Audio-Visual Navigation in 3D Environments. In *ECCV*, volume 12351 LNCS, pages 17–36. Springer, 2019.

[2] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. RLBench: The Robot Learning Benchmark & Learning Environment. *RAL*, 5(2):3019–3026, 2019.

[3] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín. robosuite: A Modular Simulation Framework and Benchmark for Robot Learning. *arXiv:2009.12293*, 2020.

[4] Y. Hu, L. Anderson, T. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand. Difftaichi: Differentiable programming for physical simulation. In *Proc. of ICLR*, 2020.

[5] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme. NeuralSim: Augmenting Differentiable Simulators with Neural Networks. *arXiv:2011.04217*, 2020.

[6] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. Available: `http://pybullet.org`, 2016–2019.

[7] J. Matas, S. James, and A. J. Davison. Sim-to-real reinforcement learning for deformable object manipulation. In *CoRL*, pages 734–743. PMLR, 2018.

[8] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *CVPR*, pages 12627–12637, 2019.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[10] I. Kostrikov, D. Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv:2004.13649*, 2020.

[11] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. In *Advances in Neural Information Processing Systems*, volume 33, pages 19884–19895. Curran Associates, Inc., 2020.

[12] Z. Ding, N. F. Lepora, and E. Johns. Sim-to-Real Transfer for Optical Tactile Sensing. *ICRA*, pages 1639–1645, 2020.

[13] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.