Learning Affordances in Object-Centric Generative Models

Yizhe Wu [1 *] **Sudhanshu Kasewa** [1 *] **Oliver Groth** [1 *] **Sasha Salter** [1] **Li Sun** [2] **Oiwi Parker Jones** [1] **Ingmar Posner** [1]

Abstract

Given visual observations of a reaching task together with a stick-like tool, we propose a novel approach that learns to exploit task-relevant object affordances by combining generative modelling with a task-based performance predictor. The embedding learned by the generative model captures the factors of variation in object geometry, e.g. length, width, and configuration. The performance predictor identifies sub-manifolds correlated with task success in a weakly supervised manner. Using a 3D simulation environment, we demonstrate that traversing the latent space in this task-driven way results in appropriate tool geometries for the task at hand. Our results suggest that affordances are encoded along smooth trajectories in the learned latent space. Given only *high-level* performance criteria (such as task success), accessing these emergent affordances via gradient descent enables the agent to manipulate learned object geometries in a targeted and deliberate way.

1. Introduction

The advent of deep generative models (e.g. Burgess et al., 2019; Greff et al., 2019; Engelcke et al., 2019) with their aptitude for unsupervised representation learning casts a new light on learning *affordances* (Gibson, 1977). This kind of representation learning raises a tantalising question: Given that generative models naturally capture factors of variation, could they also be used to expose these factors such that they can be modified in a task-driven way? We posit that a task-driven traversal of a structured latent space leads to *affordances* emerging naturally along trajectories in this space. This is in stark contrast to more common approaches to affordance learning where it is achieved via direct supervision or implicitly via imitation (e.g. Tikhanoff et al., 2013; Myers et al., 2015; Liu et al., 2018; Grabner et al., 2011; Do et al., 2018). The setting we choose for our investigation is that of tool synthesis for reaching tasks as commonly investigated in the cognitive sciences (Ambrose, 2001; Emery & Clayton, 2009).

In order to demonstrate that a task-aware latent space encodes useful affordance information we require a mechanism to train such a model as well as to purposefully explore the space. To this end we propose an architecture in which a task-based performance predictor (a classifier) operates on the latent space of a generative model (see Figure 1). During training the classifier is used to provide an auxiliary objective, aiding in shaping the latent space. Importantly, however, during test time the performance predictor is used to guide exploration of the latent space via activation maximisation (Erhan et al., 2009; Zeiler & Fergus, 2014; Simonyan et al., 2014), thus explicitly exploiting the structure of the space. While our desire to affect factors of influence is similar in spirit to the notion of disentanglement, it contrasts significantly with models such as β-VAE (Higgins et al., 2017), where the factors of influence are effectively encouraged to be axis-aligned. Our approach instead relies on a high-level auxiliary loss to discover the direction in latent space to explore.

Our experiments demonstrate that artificial agents are able to *imagine* an appropriate tool for a variety of reaching tasks by manipulating the tool's task-relevant affordances. To the best of our knowledge, this makes us the first to demonstrate an artificial agent's ability to imagine, or synthesise, images of tools appropriate for a given task via optimisation in a structured latent embedding. Similarly, while activation maximisation has been used to visualise modified input images before (e.g. Mordvintsev et al., 2015), we believe this work to be the first to effect deliberate manipulation of factors of influence by chaining the outcome of a task predictor to the latent space, and then decoding the latent representation back into a 3D mesh. Beyond the application of tool synthesis, we believe our work to provide novel perspectives on affordance learning and disentanglement in demonstrating that object affordances can be viewed as *trajectories* in a structured latent space as well as by providing a novel architecture adept at deliberately manipulating interpretable factors of influence.

[*]Equal contribution [1]Applied AI Lab, University of Oxford, UK [2]Visual Computing Group, Univeristy of Sheffield, UK. Correspondence to: Yizhe Wu or Sudhanshu Kasewa <{ywu,su}@robots.ox.ac.uk>.

Workshop on Object-Oriented Learning at ICML 2020. Copyright 2020 by the author(s).

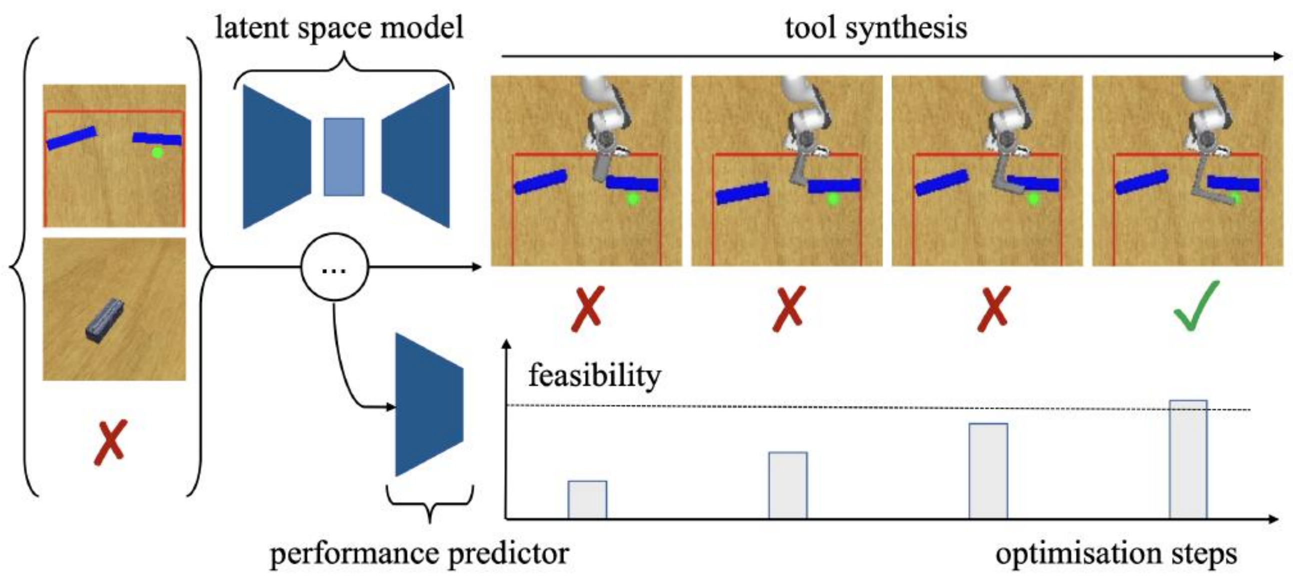


Figure 1. Tool innovation for a reaching task. Our model is trained on data-triplets {task observation, tool observation, success indicator}. The goal is to determine if the given tool can reach the green goal while avoiding blue barriers and remaining behind the red boundary. When the tool cannot satisfy these constraints, our approach (via the performance predictor) *imagines* how one may augment it in order to solve the task. We are interested in what these augmentations, during *tool synthesis*, imply about the learned object representations.

2. Related Work

An *affordance* describes a potential action to be performed on an object (e.g. a doorknob *affords* being turned) (Gibson, 1977). In computer vision and robotics, affordances are commonly learned in a supervised fashion where models discriminate between discrete affordance classes or predict masks for image regions which afford certain types of human interaction (e.g. Stoytchev, 2005; Kjellström et al., 2010; Tikhanoff et al., 2013; Mar et al., 2015; Myers et al., 2015; Do et al., 2018). Most works in this domain learn from object shapes which have been given an affordance label a priori. However, the affordance of a shape is only properly defined in the context of a task.

Recent advances in 3D shape generation employ variational models (Girdhar et al., 2016; Wu et al., 2016) to capture complex manifolds of 3D objects. Besides their expressive capabilities, the latent spaces of such models also enable smooth interpolation between shapes. Remarkable results have been demonstrated including 'shape algebra' (Wu et al., 2016) and the preservation of object part semantics (Kohli et al., 2020) and fine-grained shape styles (Yifan et al., 2019) during interpolation. This shows the potential of disentangling meaningful factors of variation in the latent representation of 3D shapes. Inspired by this, we investigate whether these factors can be exposed in a task-driven way. We propose an architecture in which a generative model for 3D object reconstruction (Liu et al., 2019) is paired with activation maximisation (e.g. Erhan et al., 2009; Zeiler & Fergus, 2014; Simonyan et al., 2014) of a task-driven performance predictor.

3. Method

Our overarching goal is to perform task-specific tool synthesis for 3D reaching tasks. We frame the challenge of tool imagination as an optimisation problem in a structured latent space obtained using a generative model. The optimisation is driven by a high-level, task-specific performance predictor, which assesses whether a target specified by a goal image I_G is reachable given a particular tool and in the presence of obstacles (Figure 1). To map from tool images into manipulable 3D tools, we first train an off-the-shelf 3D single-view reconstruction model taking as input tool images I_T^i, I_T^j and corresponding tool silhouettes I_S^i, I_S^j as rendered from two different vantage points i and j. After training, the encoder can infer the tool representation that contains the 3D structure information given a single-view RGB image and its silhouette as input. This representation is implicitly used to optimise the tool configuration to make it suitable for the task at hand. A more detailed overview of our model is given in Supplementary Figures 2 and 3.

More formally, we consider N data instances: $\{(I_G^n, I_T^{n,i}, I_T^{n,j}, I_S^{n,i}, I_S^{n,j}, \rho^n)\}_{n=1}^N$, where each example features a task image I_G, tool images I_T in two randomly selected views i and j, and their corresponding silhouettes I_S, as well as a binary label ρ indicating the feasibility of reaching the target with the given tool. Examples of task images and model inputs for five scenarios

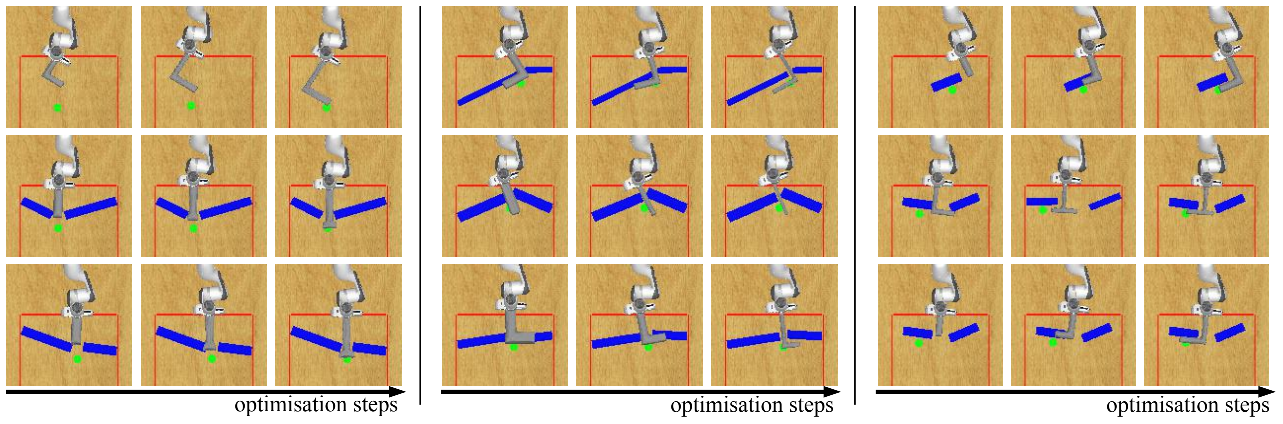


Figure 2. Qualitative results of tool evolution during the imagination process. Each row illustrates how the imagination procedure can succeed at constructing tools that solve the task by: increasing **tool length** (***left***), decreasing **tool width** (***middle***), and altering **tool shape** (***right***) creating an appropriately oriented hook. Each row in each grid represents a different imagination experiment.

types, `A–E`, are shown in Supplementary Figure 1. In all our experiments, we restrict the training input to such sparse high-level instances. For additional details on the dataset, we refer the reader to the supplementary material.

3.1. Representing Tasks and Tools

Given that our tools are presented in tool images I_T, it is necessary for the first processing step to perform a 3D reconstruction of I_T, from pixels into meshes. To achieve this single view 3D reconstruction of images into their meshes, we employ an architecture proposed in prior work (Kato et al., 2018; Wang et al., 2018). The 3D reconstruction model consists of two parts: an encoder network and a mesh decoder. Given the tool image and its silhouette in view i, I_T^i and I_S^i, we denote the latent variable encoding the tool computed by the encoder, ψ, as

$$\psi(I_T^i, I_S^i) = \mathbf{z}_T. \tag{1}$$

The mesh decoder takes $\mathbf{z}_T$ as input and synthesises the mesh by deforming a template. A differentiable renderer (Liu et al., 2019) predicts the tool's silhoutte $\hat{I}_S^j$ in another view j, which is compared to the ground-truth silhouette I_S^j to compute the silhouette loss $\mathcal{L}_s$. This silhouette loss $\mathcal{L}_s$ together with an auxiliary geometry loss $\mathcal{L}_g$ formulates the total 3D reconstruction loss

$$\mathcal{L}_{recon} = \mathcal{L}_s + \mu\mathcal{L}_g, \tag{2}$$

where μ is the weight of the geometry loss. We refer the reader to Liu et al. (2019) for the exact hyper-parameter and training setup of the 3D reconstruction model.

Task images I_G are similarly represented in an abstract latent space. For this we employ a task encoder, ϕ, which consists of a stack of convolutional layers. ϕ takes the task image I_G as input and maps it into the task embedding $\mathbf{z}_G$.

3.2. Task-driven learning

The learned representation $\mathbf{z}_T$ appears to encode task-relevant information such as tool length, width, and shape. In order to perform tool imagination, the sub-manifold of the latent space that corresponds to the task-relevant features needs to be accessed and traversed. This is achieved by adding a three-layer MLP as a classifier σ. The classifier σ takes as input a concatenation $\mathbf{h}_{cat}$ of the task embedding $\mathbf{z}_G$ and the tool representation $\mathbf{z}_T$, and predicts the softmax over the binary task success. The classifier learns to identify the task-relevant sub-manifold of the latent space by using the sparse success signal ρ and optimising the binary-cross entropy loss, such that

$$\begin{aligned}\mathcal{L}_{task}\left(\sigma\left(\mathbf{h}_{cat}\right), \rho\right) = &-\left(\rho \log\left(\sigma(\mathbf{h}_{cat})\right)\right. \\ &\left.+ (1-\rho)\log\left(1-\sigma(\mathbf{h}_{cat})\right)\right),\end{aligned} \tag{3}$$

where $\rho \in \{0, 1\}$ is a binary signal indicating whether or not it is feasible to solve the task with the given tool. The whole system is trained end-to-end with the loss given by

$$\mathcal{L}\left(I_G, I_T^i, I_S^i, I_T^j, I_S^j, \rho\right) = \mathcal{L}_{recon} + \mathcal{L}_{task}. \tag{4}$$

Note that the gradient from the task classifier σ propagates through both the task encoder ϕ and the toolkit encoder ψ, thereby helping to shape the latent representations of the toolkit with respect to the requirements for task success.

3.3. Tool imagination

Once trained, our model can synthesise new tools by traversing the latent manifold of individual tools following the trajectories that maximise classification success given a tool image and its silhouette (Supplementary Figure 2). To do this, we first pick a tool candidate and concatenate its representation $\mathbf{z}_T$ with the task embedding $\mathbf{z}_G$, warm-starting the imagination process. The concatenated embedding $\mathbf{h}_{cat}$ is then fed into the performance predictor σ to compute the gradient with respect to the tool embedding $\mathbf{z}_T$. We use activation maximisation (Erhan et al., 2009; Zeiler & Fergus, 2014; Simonyan et al., 2014) to optimise for $\mathbf{z}_T$ given the loss $\mathcal{L}_{task}$ of the success estimation $\sigma(\mathbf{h}_{cat})$ and a feasibility target $\rho_s = 1$ such that

$$\mathbf{z}_T = \mathbf{z}_T + \eta \frac{\partial \mathcal{L}_{task}\left(\sigma\left(\mathbf{z}_T\right), \rho_s\right)}{\partial \mathbf{z}_T}, \tag{5}$$

where η denotes the learning rate for the update. We apply this gradient update for S steps or until the success estimation $\sigma(\mathbf{z}_T)$ reaches a threshold γ, and use $\psi'(\mathbf{z}_T)$ to generate the imagined 3D tool mesh represented by $\mathbf{z}_T$.

4. Experiments

4.1. Model Training

In order to gauge the influence of the task feasibility signal on the latent space of the tools, we train the model in two different setups. A *task-driven* model is trained with a curriculum: First, the 3D reconstruction module is trained on tool images alone. Then, the performance predictor is trained jointly with this backbone, i.e. the gradient from the predictor is allowed to back-propagate into the encoder of the 3D reconstruction network. In a *task-unaware* ablation we keep the pre-trained 3D reconstruction weights fixed during the predictor training removing any influence of the task performance on the latent space. A *random walk* baseline reveals that a simple stochastic exploration of the latent space is not sufficient to find suitable tool geometries.

4.2. Qualitative Results

Qualitative examples of the tool imagination process are provided in Figure 2 and Supplementary Figure 4. In the right-middle example of Figure 2, a novel T-shape tool is created, suggesting that the model encodes the vertical stick-part and horizontal hook-part as distinct elementary parts. The model also learns to interpolate the direction of the hook part between pointing left and right, which leads to a novel tool. As shown in Supplementary Figure 4, tools are modified in a smooth manner, leading us to hypothesise that tools are embedded in a continuous manifold of changing length, width, and configuration. Optimising the latent embedding for the highest performance predictor score often drives the tools to evolve along these properties. This suggests that these geometric variables are encoded as *trajectories* in the structured latent space learnt by our model and deliberately traversed via a high-level task objective in the form of the performance predictor.

4.3. Quantitative Results

We evaluate whether our model can generate tools to succeed in the reaching tasks. For each instance the target signal for feasibility is set to $\rho_s = 1$, i.e. *success*. Then, the latent vector of the tool is modified via backpropagation using a learning rate of 0.01 for $10,000$ steps or until $\sigma(\mathbf{h}_{cat})$ reaches the threshold of $\gamma = 0.997$. The imagined tool mesh is generated via the mesh decoder ψ'. This is then rendered into a top-down view and tested using a feasibility test which checks whether all geometric constraints are satisfied.We report our results in Table 1 as mean success performances within a 95% confidence interval around the estimated mean. The task-unaware ablation provides a much stronger baseline compared to the random walk baseline transforming tools successfully in 63.6% of the cases. However, the task-driven model significantly outperforms it, boasting a global success rate of 83.8% on the test cases. This finding implies that the joint training of 3D latent representation and task performance prediction shapes the latent space in a 'task-aware' way encoding properties which are conducive to task success (e.g. length, width, and configuration of a tool).

	Tool Imagination Success [%]		
Scn	Task-Driven	Task-Unaware	Random Walk
A	**96.4** $\pm$ **2.3**	55.6 $\pm$ 6.2	3.6 $\pm$ 2.3
B	**78.8** $\pm$ **5.1**	42.0 $\pm$ 6.1	5.6 $\pm$ 2.9
C	**76.4** $\pm$ **5.3**	56.8 $\pm$ 6.1	23.6 $\pm$ 5.3
D	**81.2** $\pm$ **4.8**	**75.2** $\pm$ **5.4**	2.4 $\pm$ 1.9
E	**86.4** $\pm$ **4.3**	**88.4** $\pm$ **4.0**	13.6 $\pm$ 4.3
Tot	**83.8** $\pm$ **2.0**	63.6 $\pm$ 2.7	9.8 $\pm$ 1.7

Table 1. Comparison of imagination processes when artificially warmstarting from the same unsuitable tools in each instance. Best results are highlighted in bold. The task scenarios, A–E, are described in Supplementary Figure 1.

5. Conclusion

In this paper we investigate the ability of an agent to synthesise tools via task-driven imagination in a set of simulated reaching tasks. Our approach uses a novel architecture in which a high-level performance predictor drives an optimisation process in a structured latent space. The model successfully learns to modify interpretable properties of tools such as length, width, and configuration. The experimental results suggest that these object affordances are encoded as *trajectories* in a learnt latent space, which are sought out during activation maximisation of the task predictor. In addition, more task-appropriate trajectories were found by jointly training the performance predictor with the encoder than by training them independently. This work may help in our understanding of object affordances, while offering up a novel way to disentangle interpretable factors of variation.

References

Ambrose, S. H. Paleolithic technology and human evolution. *Science*, 291(5509):1748–1753, 2001.

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. MONet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

Do, T., Nguyen, A., Reid, I. D., Caldwell, D. G., and Tsagarakis, N. G. AffordanceNet: An end-to-end deep learning approach for object affordance detection. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

Emery, N. J. and Clayton, N. S. Tool use and physical cognition in birds and mammals. *Current Opinion in Neurobiology*, 19(1):27–33, 2009.

Engelcke, M., Kosiorek, A. R., Parker Jones, O., and Posner, I. GENESIS: Generative scene inference and sampling with object-centric latent representations. *arXiv preprint arXiv:1907.13052*, 2019.

Erhan, D., Bengio, Y., Courville, A., and Vincent, P. Visualizing higher-layer features of a deep network. Technical report, University of Montreal, 2009.

Gibson, J. J. The theory of affordances. In *Perceiving, Acting, and Knowing: Toward an Ecological Psychology*. Lawrence Erlbaum, Hilldale, USA, 1977.

Girdhar, R., Fouhey, D., Rodriguez, M., and Gupta, A. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016.

Grabner, H., Gall, J., and Van Gool, L. What makes a chair a chair? In *Computer Vision and Pattern Recognition (CVPR)*, pp. 1529–1536, 2011.

Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., and Lerchner, A. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning (ICML)*, 2019.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. β-vae: Learning basic visual concepts with a constraied variational framework. In *ICLR*, 2017.

Kato, H., Ushiku, Y., and Harada, T. Neural 3D mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3907–3916, 2018.

Kjellström, H., Romero, J., and Kragic, D. Visual object-action recognition: Inferring object affordances from human demonstration. *Computer Vision and Image Understanding*, pp. 81–90, 2010.

Kohli, A., Sitzmann, V., and Wetzstein, G. Inferring semantic information with 3D neural scene representations. *arXiv preprint arXiv:2003.12673*, 2020.

Liu, S., Li, T., Chen, W., and Li, H. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7708–7717, 2019.

Liu, Y., Gupta, A., Abbeel, P., and Levine, S. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1118–1125, 2018.

Mar, T., Tikhanoff, V., Metta, G., and Natale, L. Self-supervised learning of grasp dependent tool affordances on the icub humanoid robot. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3200–3206. IEEE, 2015.

Mordvintsev, A., Olah, C., and Tyka, M. Inceptionism: Going deeper into neural networks, 2015. URL `https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html`.

Myers, A., Teo, C. L., Fermüller, C., and Aloimonos, Y. Affordance detection of tool parts from geometric features. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1374–1381, 2015.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*, 2014.

Stoytchev, A. Behavior-grounded representation of tool affordances. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3071–3076, 2005.

Tikhanoff, V., Pattacini, U., Natale, L., and Metta, G. Exploring affordances and tool use on the iCub. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 130–137, 2013.

Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., and Jiang, Y.-G. Pixel2mesh: Generating 3D mesh models from single RGB images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 52–67, 2018.

Wu, J., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 82–90. Curran Associates, Inc., 2016.

Yifan, W., Aigerman, N., Kim, V., Chaudhuri, S., and Sorkine-Hornung, O. Neural cages for detail-preserving 3D deformations. *arXiv preprint arXiv:1912.06395*, 2019.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *Proceedings of the IEEE European Conference on Computer Vision*, pp. 818–833, 2014.

A. Dataset

A. Detailed Description

To investigate tool imagination, we designed a set of simulated reaching tasks with clear and controllable factors of influence. Each task image is comprised of a green target button, three red lines delineating the workspace area, and, optionally, a set of blue obstacles. We also vary the goal location and the sizes and positions of the obstacles. For each task image, we provide a second image depicting a tool, i.e. a straight stick or an L-shaped hook, with varying dimensions, shapes, and textures. Given a pair of images (i.e. the task image and the tool image), the goal is to predict whether the tool for a given task scene can reach the target (the green dot) while simultaneously avoiding obstacles (blue areas) and remaining on the exterior of the workspace (i.e. behind the red line). Depending on the task image, the applicability of a tool is determined by different subsets of its attributes. For example, if the target button is unobstructed, then any tool of sufficient length will satisfy the constraints (regardless of its width or shape). However, when the target is hidden behind a corner, or only accessible through a narrow gap, an appropriate tool also needs to feature a long-enough hook, or a thin-enough handle, respectively. As depicted in Supplementary Figure 1, we have designed five scenario types to study these factors of influence in isolation and in combination. The task setting and tool are first rendered in a top-down view for the geometric applicability check. Then the tool is rendered in 12 different views for the 3D reconstruction. The geometric applicability check verifies whether or not any of the tools can reach the target while satisfying the task constraints.

B. Dataset Construction Minutiae

Our synthetic dataset consists of pairs of task image (in top-down view) and tool image as well as the corresponding silhouettes in 12 different views. We also record the tool image from a top-down view for the tool-applicability check. All images are 128×128 pixels. The applicability of a tool to a task is determined by sampling 100 interior points of the tool polygon, overlaying the sampled point with the target and rotating the tool polygon between -60 degrees and $+60$ degrees from the y-axis. If any such pose of the tool satisfies all constraints (i.e. touching the space behind the red line while not colliding with any obstacle), we consider the tool applicable for the given task. All train- and test-cases are constrained to have only sticks and hooks as tools.

The dataset contains a total of 18,500 scenarios. Table 2 shows a breakdown of each by scenario type and split. Each split has an equal number of feasible and infeasible instances, for each scenario type.

Type	Training	Validation
A	4,000	500
B	4,000	500
C	4,000	500
D	4,000	500
E	-	500
Total	16,000	2,500

Table 2. Number of instances by scenario type.

B. Architecture and Training Details

A. Model Architecture

As described in the main text, our model comprises two main parts (as depicted in Supplementary Figure 2). On the one hand, we used a task-based classifier, while on the other we used a encoder-decoder model as a single-view 3D reconstruction model. We briefly describe each component in turn.

A.1. 3D Reconstruction Model

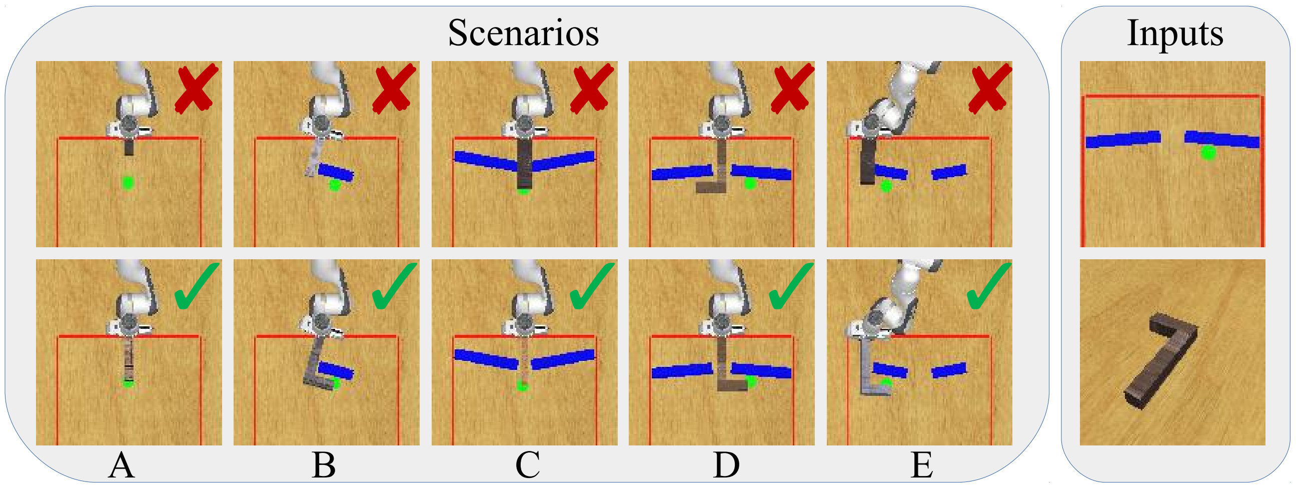
As a 3D reconstruction model, we faithfully re-implement the encoder-decoder architecture identical to (Kato et al., 2018; Liu et al., 2019). A high-level model schematic is shown in Supplementary Figure 3.

A.2. Task-Based Classifier

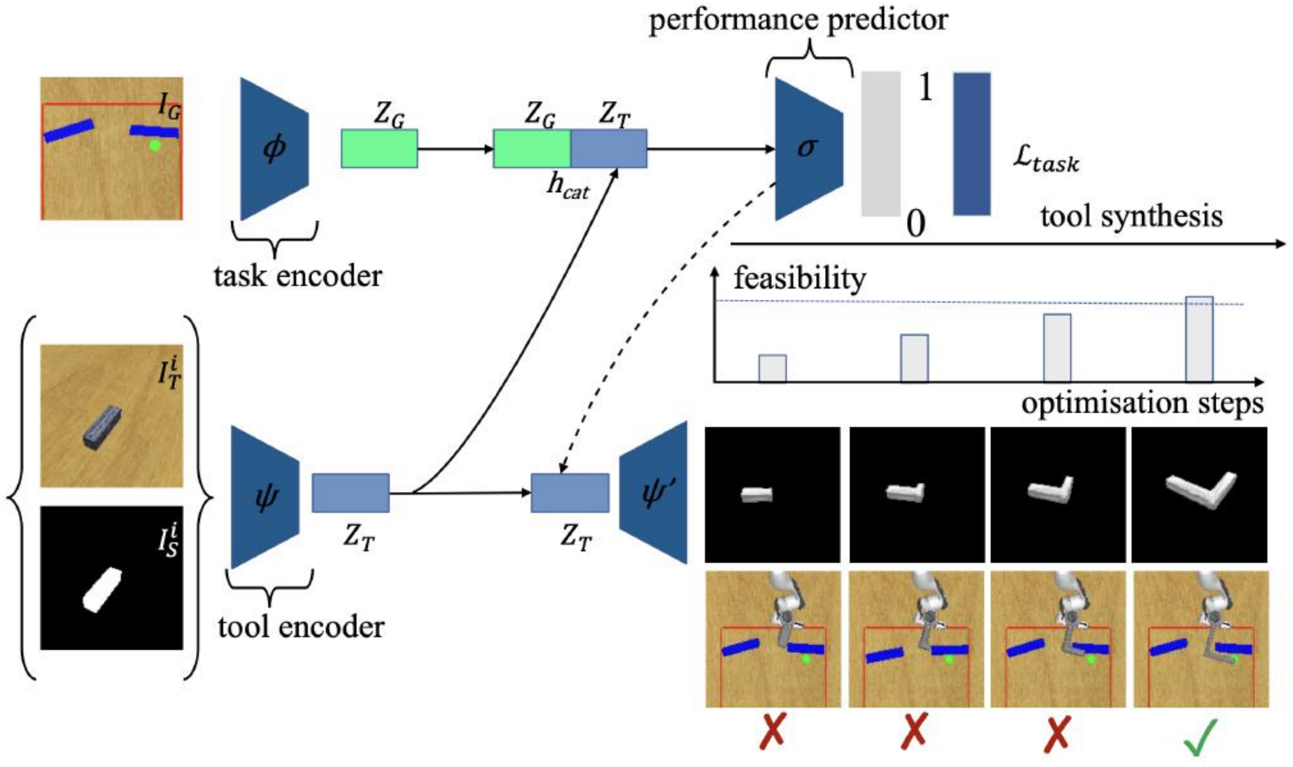
The task-based classifier consists of two sub-parts: a task encoder and a classifier. The task encoder stacks five convolutional blocks followed by a single convolutional layer. First, the five consecutive convolutional blocks make use of 16, 32, 64, 64, 128 output channels respectively. Each convolutional block contains two consecutive sub-convolutional blocks, each with a kernel size of 5 and padding of 2 (and with stride of 1 and stride of 2, respectively). Second, the additional convolutional layer has kernel size 4, stride 1, padding 0, and 256 output channels. All convolutional layers use an ELU nonlinearity. The classifier is a three layer MLP with input dimension 768, and successive hidden layers of 1024 and 512 neurons respectively. Each of these layers use eLU activation functions. Given that the classifier outputs logits for a binary classifier, the output dimension is 2.

B. Training Details

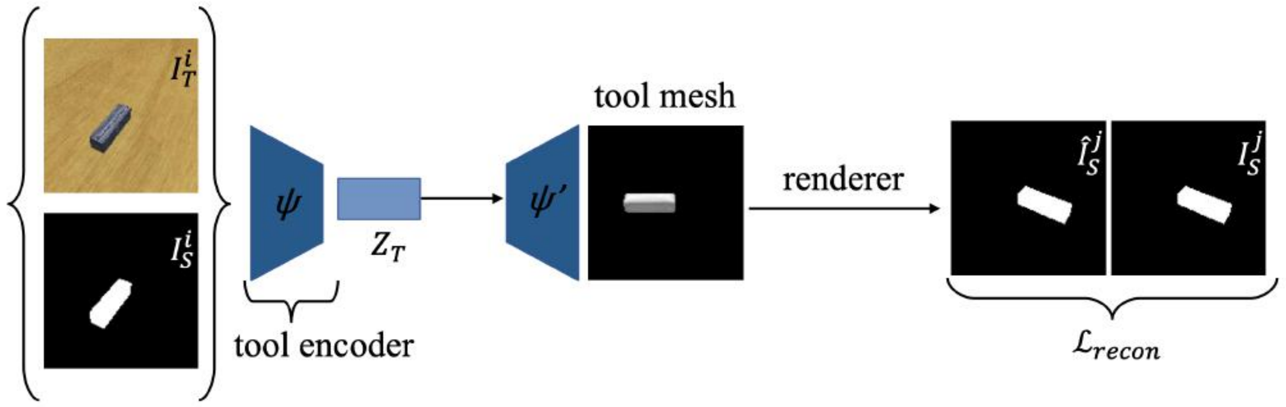
All experiments are performed in PyTorch, using the ADAM optimiser with a learning rate of 0.0001 and a batch size of 16.



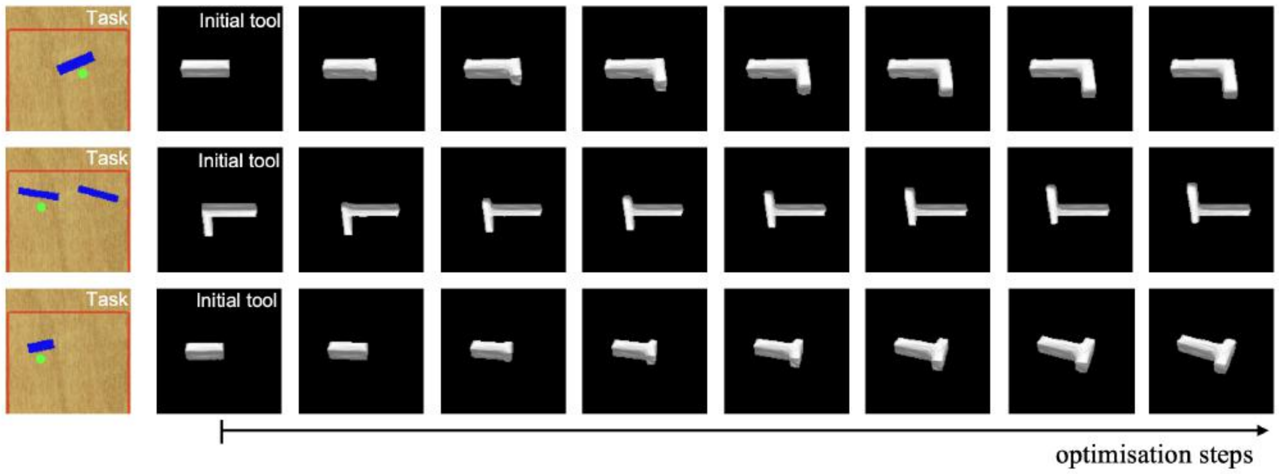
Supplementary Figure 1. (Left) Task examples from our dataset. Top and bottom rows correspond to unsuccessful and successful tool examples respectively. Columns `A` - `E` represent five different task scenario types each imposing different tool constraints including width, length, orientation and shape. Note that the robot is fixed at its base on the table and constrained to remain outside the red boundary. Hence, it can only reach the green target with a tool while avoiding collisions with the blue obstacles. (Right) Model inputs {task observation, tool observation} during training and test time.



Supplementary Figure 2. The model architecture. A convolutional encoder ϕ represents the task image I_G as a latent vector $\mathbf{z}_G$. In parallel, given I_T^i,I_S^i, the 3D tool encoder ψ produces a latent representation $\mathbf{z}_T$. The variational model (ψ, ψ') is trained as described in (Kato et al., 2018; Wang et al., 2018). The concatenated tool-task representation $\mathbf{h}_{cat}$ is used by a classifier σ to estimate the success of the tool for solving the task. Given the gradient signal during optimisation for task success via the classifier, the latent tool representation $\mathbf{z}_T$ gets updated to render an increasingly suitable tool for the task.



Supplementary Figure 3. Model for 3D reconstruction.



Supplementary Figure 4. Examples of tool synthesis progression during the imagination process. In the top row, a stick tool morphs into a hook. The middle row shows a left-facing hook transforming into a right-facing hook. In the bottom row, the tool changes into a novel T-shape. Constraints on these optimisations are specified via task embeddings corresponding to the task images on the far left.