

## A DATASET

### A.1 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) whilst 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 Figure 3, 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.

### A.2 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 top-down view for the tool-applicability check. All images are  $128 \times 128$  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.

Table 2: Number of instances by 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

## B ARCHITECTURE AND TRAINING DETAILS

### B.1 MODEL ARCHITECTURE

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

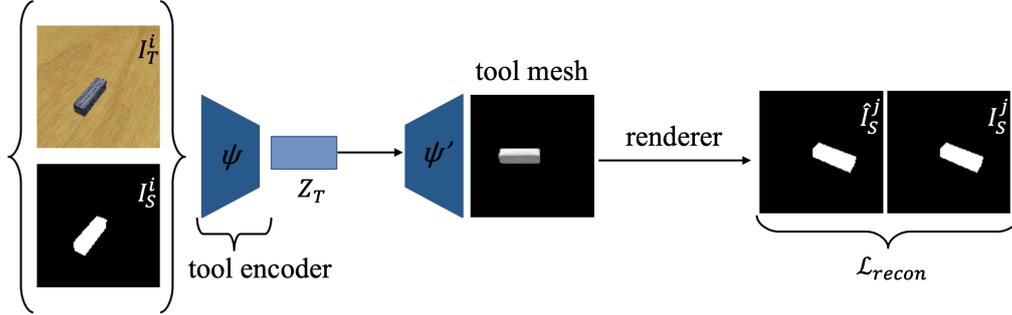


Figure 6: Model for 3D reconstruction.

### B.1.1 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 fig. 6.

### B.1.2 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.2 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.