# RT-Sketch: Goal-Conditioned Imitation Learning from Hand-Drawn Sketches - Supplementary Material

In this section, we provide further details on the visual goal representations RT-Sketch sees at train and test time (Appendix A), qualitative visualizations of experimental rollouts (Appendix B), limitations (Appendix C) of RT-Sketch, as well as the interfaces used for data annotation, evaluation, and human assessment (Appendix D). To best understand RT-Sketch's performance and see its emergent capabilities, please refer to our website.[3].

## A Sketch Goal Representations

Since the main bottleneck to training a sketch-to-action policy like RT-Sketch is collecting a dataset of paired trajectories and goal sketches, we first train an image-to-sketch translation network $\mathcal{T}$ mapping image observations $o_i$ to sketch representations $g_i$, discussed in Section 3. To train $\mathcal{T}$, we first take a pre-trained network for sketch-to-image translation (Li et al., 2019) trained on the ContourDrawing dataset of paired images and edge-aligned sketches (Fig. 4). This dataset contains $L^{(i)} = 5$ crowdsourced sketches per image for 1000 images. By pre-training on this dataset, we hope to embed a strong prior in $\mathcal{T}$ and accelerate learning on our much smaller dataset. Next, we finetune $\mathcal{T}$ on a dataset of 500 manually drawn line sketches for RT-1 robot images. We visualize a few examples of our manually sketched goals in Fig. 5 under 'Line Drawings'.

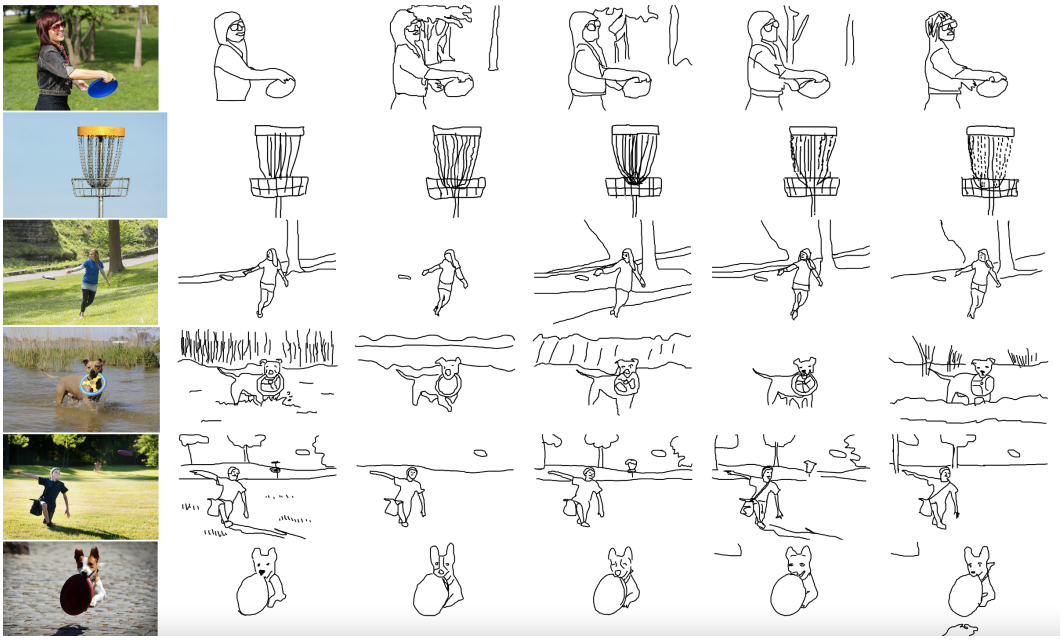

Figure 4: **ContourDrawing Dataset**: We visualize 6 samples from the ContourDrawing Dataset from (Li et al., 2019). For each image, 5 separate annotators provide an edge-aligned sketch of the scene by outlining on top of the original image. As depicted, annotators are encouraged to preserve main contours of the scene, but background details or fine-grained geometric details are often omitted. Li et al. (2019) then train an image-to-sketch translation network $\mathcal{T}$ with a loss that encourages aligning with at least one of the given reference sketches (Eq. (2)).

Notably, while we only train $\mathcal{T}$ to map an image to a black-and-white line sketch $\hat{g}_i$, we consider various augmentations $\mathcal{A}$ on top of generated goals to simulate sketches with varied colors, affine and perspective distortions, and levels of detail. Fig. 5 visualizes a few of these augmentations, such as automatically colorizing black-and-white sketches by superimposing a blurred version of the original RGB image, and treating an edge-detected version of the original image as a generated

[3] http://rt-sketch-anon.github.io

sketch to simulate sketches with a lot of details. We generate a dataset for training RT-Sketch by 'sketchifying' hind-sight relabeled goal images via $\mathcal{T}$ and $\mathcal{A}$.

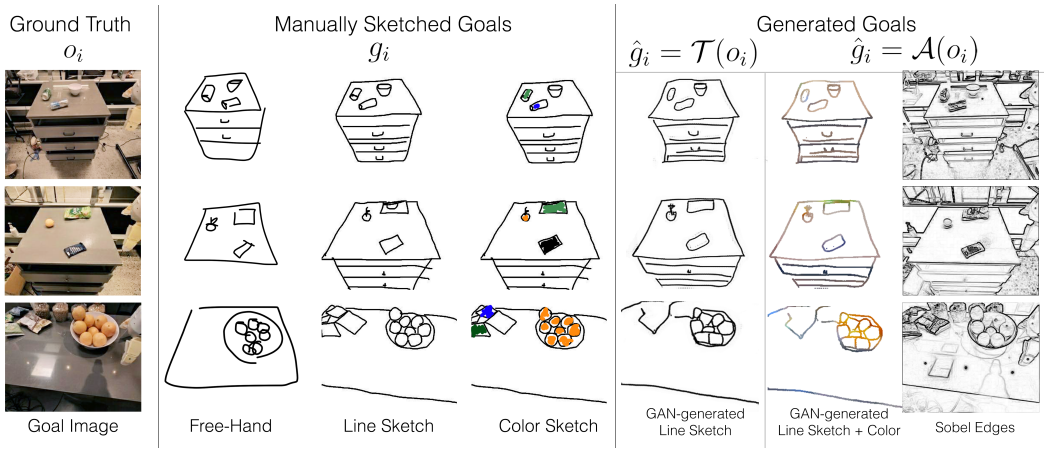


Figure 5: **Visual Goal Diversity**: RT-Sketch is capable of handling a variety of visual goals at both train and test time. RT-Sketch is trained on generated and augmented images like those shown on the right below 'Generated Goals'. But it can also interpret free-hand, line sketches, and colored sketches at test time such as those on the left below 'Manually Sketched Goals'.

Although RT-Sketch is only trained on generated line sketches, colorized line sketches, edge-detected images, and goal images, we find that it is able to handle sketches of even greater diversity. This includes non-edge aligned free-hand sketches and sketches with color infills, like those shown in Fig. 5.

## B Rollout Visualizations

To interpret RT-Sketch's performance, beyond quantitative notions of alignment like pixelwise distance or Likert ratings as in Section 4.2, we provide qualitative visualizations of experimental rollouts. In Fig. 6, Fig. 7, Fig. 8, and Fig. 10, we visualize each policy's behavior for **H1, H2, H3** and **H4**, respectively. Fig. 9 visualizes the four tiers of difficulty in language ambiguity that we analyze for **H4**. To best understand RT-Sketch's performance, we kindly refer interested readers to our website containing a much more detailed overview and videos of all policies.

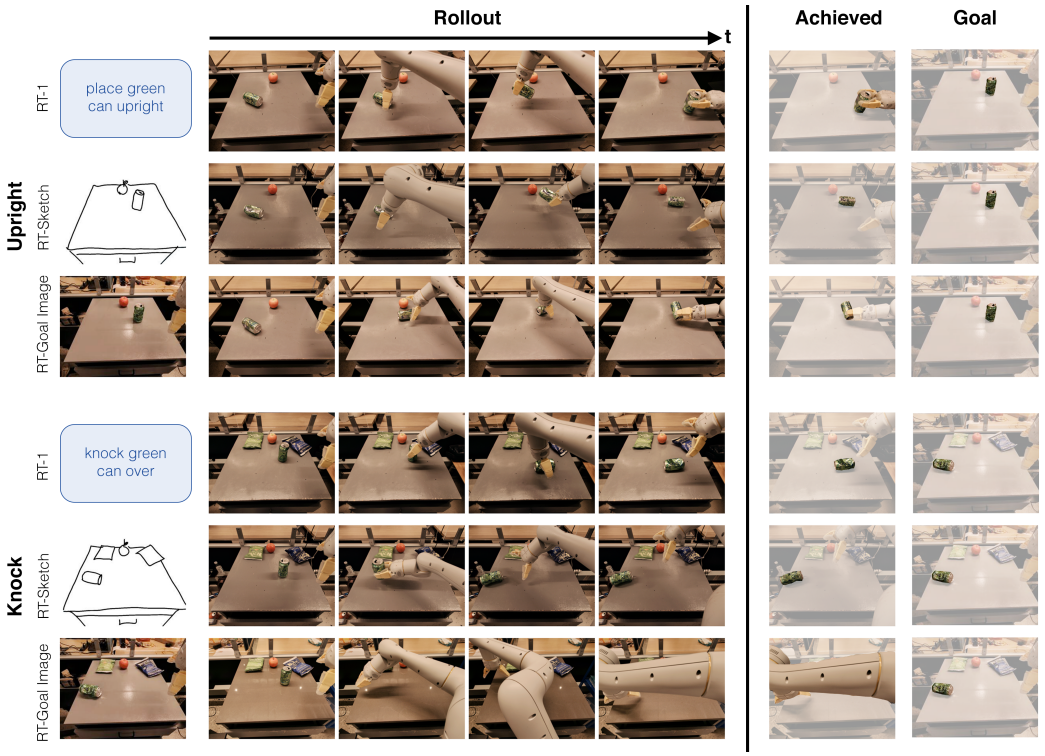


Figure 6: **H1 Rollout Visualization**: We visualize the performance of RT-1, RT-Sketch, and RT-Goal-Image on two skills from the RT-1 benchmark (*upright* and *knock*). For each skill, we visualize the goal provided as input to each policy, along with the policy rollout. We see that for both skills, RT-1 obeys the semantic task at hand by successfully placing the can upright or sideways, as intended. Meanwhile, RT-Sketch and RT-Goal-Image struggle with orienting the can upright, but successfuly knock it sideways. Interestingly, both RT-Sketch and RT-Goal-Image are able to place the can in the desired location (disregarding can orientation) whereas RT-1 does not pay attention to where in the scene the can should be placed. This is indicated by the discrepancy in position of the can in the achieved versus goal images on the right. This trend best explains the anomalous performance of RT-Sketch and RT-Goal-Image in perceived Likert ratings for the upright task (Fig. 3), but validates their comparably higher spatial precision compared to RT-1 across all benchmark skills (Table 1).

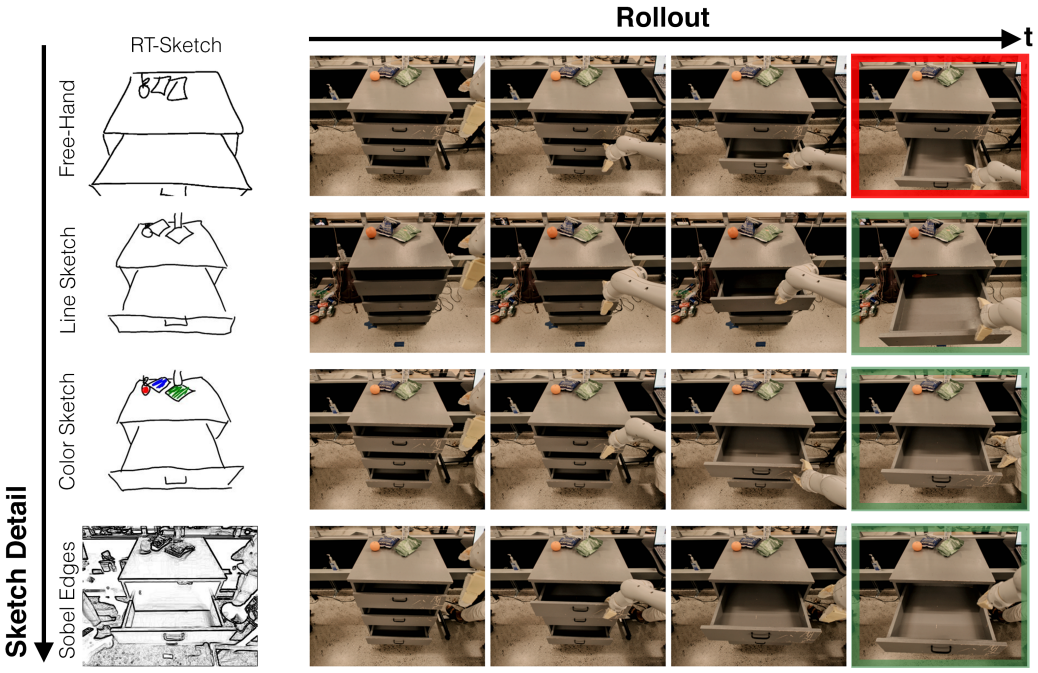


Figure 7: **H2 Rollout Visualization**: For the *open drawer* skill, we visualize four separate rollouts of RT-Sketch operating from different input types. Free-hand sketches are drawn without outlining over the original image, such that they can contain marked perspective differences, partially obscured objects (drawer handle), and roughly drawn object outlines. Line sketches are drawn on top of the original image using the sketching interface we present in Appendix Fig. 13. Color sketches merely add color infills to the previous modality, and Sobel Edges represent an upper bound in terms of unrealistic sketch detail. We see that RT-Sketch is able to successfully open the correct drawer for any sketch input except the free-hand sketch, without a noticeable performance gain or drop. For the free-hand sketch, RT-Sketch still recognizes the need for opening a drawer, but the differences in sketch perspective and scale can occasionally cause the policy to attend to the wrong drawer, as depicted.


Figure 8: **H3 Rollout Visualization**: We visualize qualitative rollouts for RT-Sketch and RT-Goal-Image for 3 separate trials of the *move near* skill subject to distractor objects. In Column 2, we highlight the relevant non-distractor objects that the policy must manipulate in order to achieve the given goal. In Trial 1, we see that RT-Sketch successfuly attends to the relevant objects and moves the blue chip bag near the coke can. Meanwhile, RT-Goal-Image is confused about which blue object to manipulate, and picks up the blue pepsi can instead of the blue chip bag (A). In Trial 2, RT-Sketch successfully moves an apple near the fruit on the left. A benefit of sketches is their ability to capture instance multimodality, as any of the fruits highlighted in Column 2 are valid options to move, whereas this does not hold for an overspecified goal image. RT-Goal-Image erroneously picks up the green chip bag (B) instead of a fruit. Finally, Trial 3 shows a failure for both policies. While RT-Sketch successfully infers that the green can must be moved near the red one, it accidentally knocks over the red can (C) in the process. Meanwhile, RT-Goal-Image prematurely drops the green can and instead tries to pick the green chip bag (D).

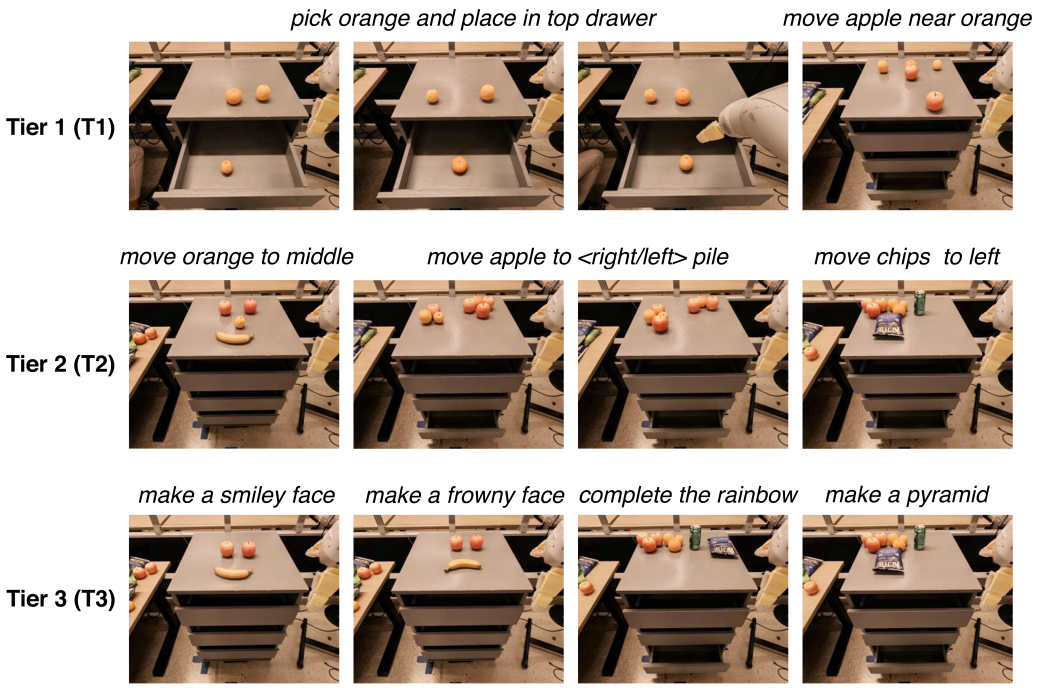


Figure 9: **H4 Tiers of Difficulty**: To test **H4**, we consider language instructions that are either ambiguous due the presence of multiple similar object instances (**T1**), are somewhat out-of-distribution for RT-1 (**T2**), or are far out-of-distribution and difficult to specify concretely without lengthier descriptions (**T3**). Each image represents the ground truth goal image paired with the task description.

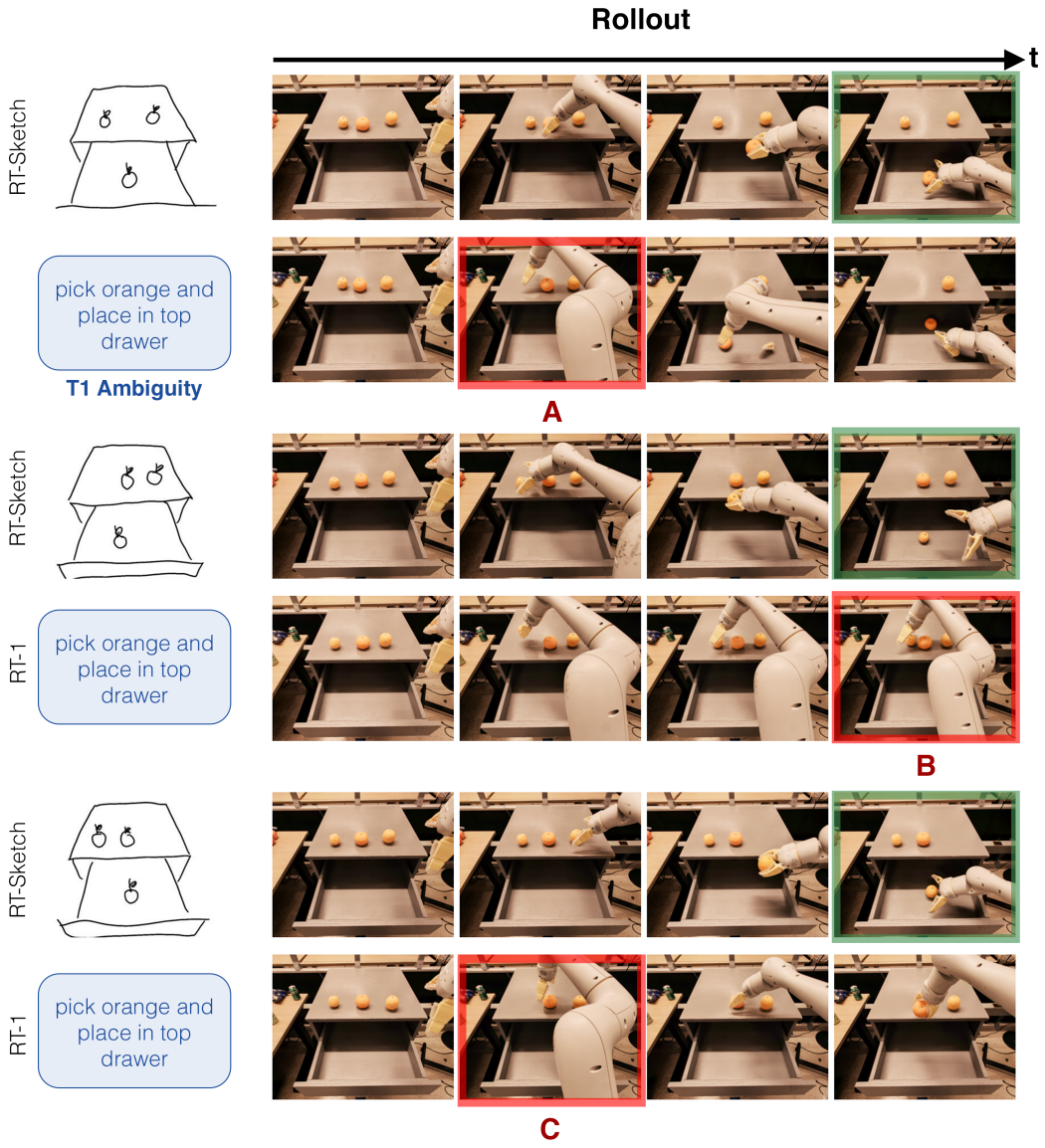


Figure 10: **H4 Rollout Visualization (T1 as visualized in Fig. 9)**: One source of ambiguity in language descriptions is mentioning an object for which there are multiple instances present. For example, we can easily illustrate three different desired placements of an orange in the drawer via a sketch, but an ambiguous instruction cannot easily specify which orange is relevant to pick and place. In all rollouts, RT-Sketch successfully places the correct orange in the drawer, while RT-1 either picks up the wrong object (A), fails to move to the place location (B), or knocks off one of the oranges (C). Although in this case, the correct orange to manipulate could easily be specified with a spatial relation like *pick up the ⟨ left/middle/right ⟩ orange*, we show below in Appendix Fig. 11 that this type of language is still out of the realm of RT-1's semantic familiarity.

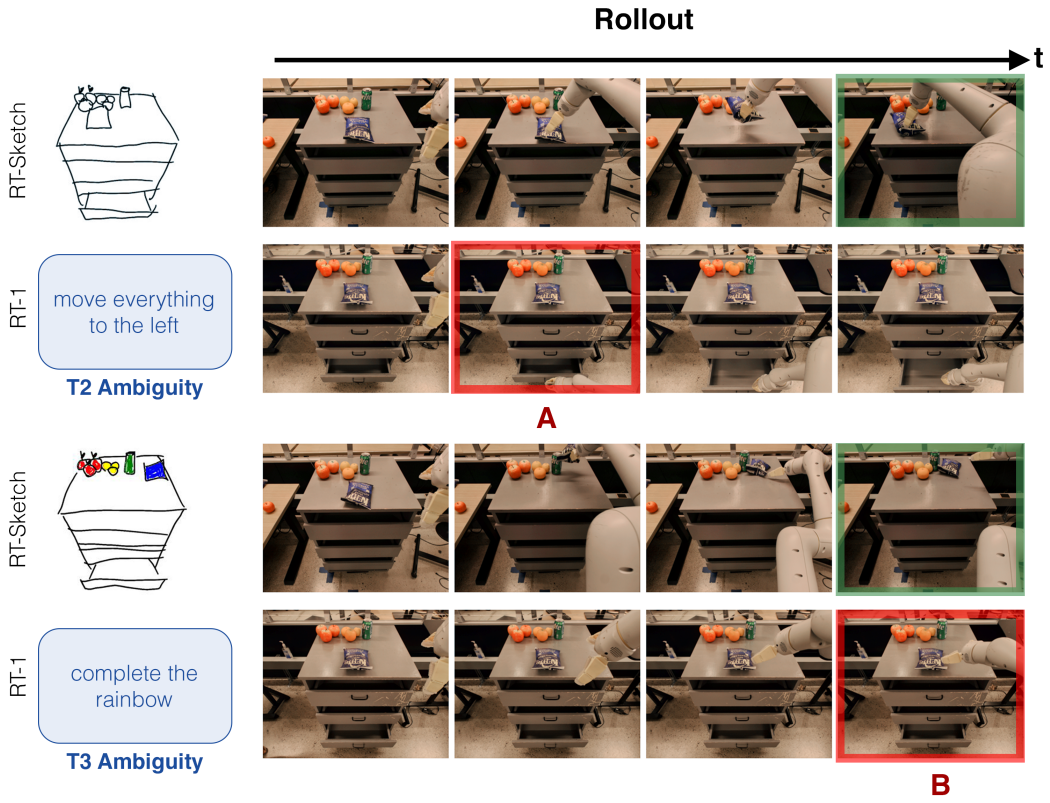


Figure 11: **H4 Rollout Visualization (T2-3 as visualized in Fig. 9)**: For **T2**, we consider language with spatial cues that intuitively should help the policy disambiguate in scenarios like the oranges in Fig. 10. However, we find that RT-1 is not trained to handle such spatial references, and this kind of language causes a large distribution shift leading to unwanted behavior. Thus, for the top rollout of trying to move the chip bag to the left where there is an existing pile, RT-Sketch completes the skill without issues, but RT-1 attempts to open the drawer instead of even attempting to rearrange anything on the countertop (A). For **T3**, we consider language goals that are even more abstract in interpretation, without explicit objects mentioned or spatial cues. Here, sketches are advantageous in their ability to succinctly communicate goals (i.e. visual representation of a rainbow), whereas the corresponding language task string is far too underspecified and OOD for the policy to handle (B).

## C RT-SKETCH FAILURE MODES AND LIMITATIONS

While RT-Sketch is performant at several manipulation benchmark skills, capable of handling different levels of sketch detail, robust to visual distractors, and unaffected by ambiguous language, it is not without failures and limitations.

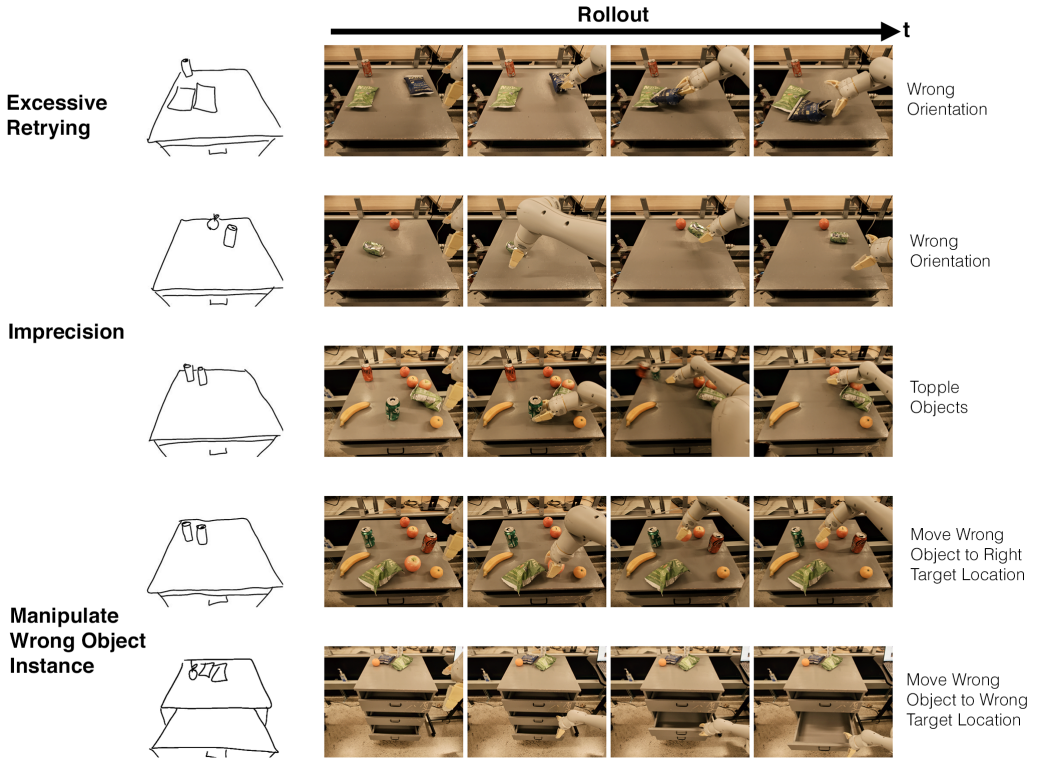


Figure 12: **RT-Sketch Failure Modes**

In Fig. 12, we visualize the failure modes of RT-Sketch. One failure mode we see with RT-Sketch is occasionally re-trying excessively, as a result of trying to align the scene as closely as possible. For instance, in the top row, Rollout Image 3, the scene is already well-aligned, but RT-Sketch keeps shifting the chip bag which causes some misalignment in terms of the chip bag orientation. Still, this kind of failure is most common with RT-Goal-Image (Table 1), and is not nearly as frequent for RT-Sketch. We posit that this could be due to the fact that sketches enable high-level spatial reasoning without over-attending to pixel-level details.

One consequence of spatial reasoning at such a high level, though, is an occasional lack of precision. This is noticeable when RT-Sketch orients items incorrectly (second row) or positions them slightly off, possibly disturbing other items in the scene (third row). This may be due to the fact that sketches are inherently imperfect, which makes it difficult to reason with such high precision.

Finally, we see that RT-Sketch occasionally manipulates the wrong object (rows 4 and 5). Interestingly, we see that a fairly frequent pattern of behavior is to manipulate the wrong object (orange in row 4) to the right target location (near green can in row 4). This may be due to the fact that the sketch-generating GAN has occasionally hallucinated artifacts or geometric details missing from the actual objects. Having been trained on some examples like these, RT-Sketch can mistakenly perceive the wrong object to be aligned with an object drawn in the sketch. However, the sketch still indicates the relative desired spatial positioning of objects in the scene, so in this case RT-Sketch still attempts to align the incorrect object with the proper place.

Finally, the least frequent failure mode is manipulating the wrong object to the wrong target location (i.e. opening the wrong drawer handle). This is most frequent when the input is a free-hand sketch, and could be mmitigated by increasing sketch detail (Table 2).

## D Evaluation and Assessment Interfaces

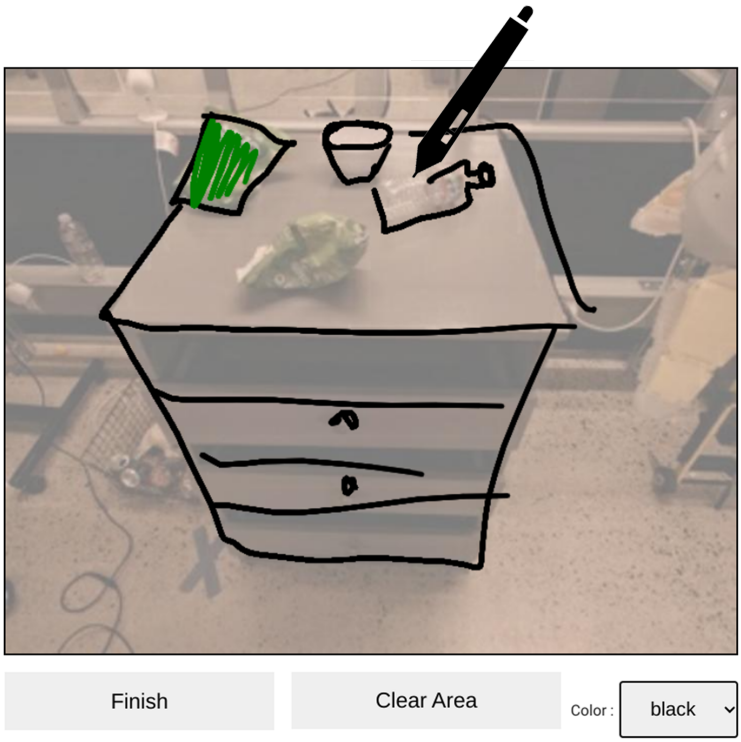


Figure 13: **Sketching UI**: We design a custom sketching interface for manually collecting paired robot images and sketches with which to train $\mathcal{T}$, and for sketching goals for evaluation. The interface visualizes the current robot observation, and provides the ability to draw on a digital screen with a stylus. The interface supports different colors and erasure. We note that intuitively, drawing on top of the image is not an unreasonable assumption to make, since current agent observations are far more readily available than a goal image, for instance. Additionally, the overlay is intended to make the sketching interface easy for the user to provide, without having to eyeball edges for the drawers or handles blindly.

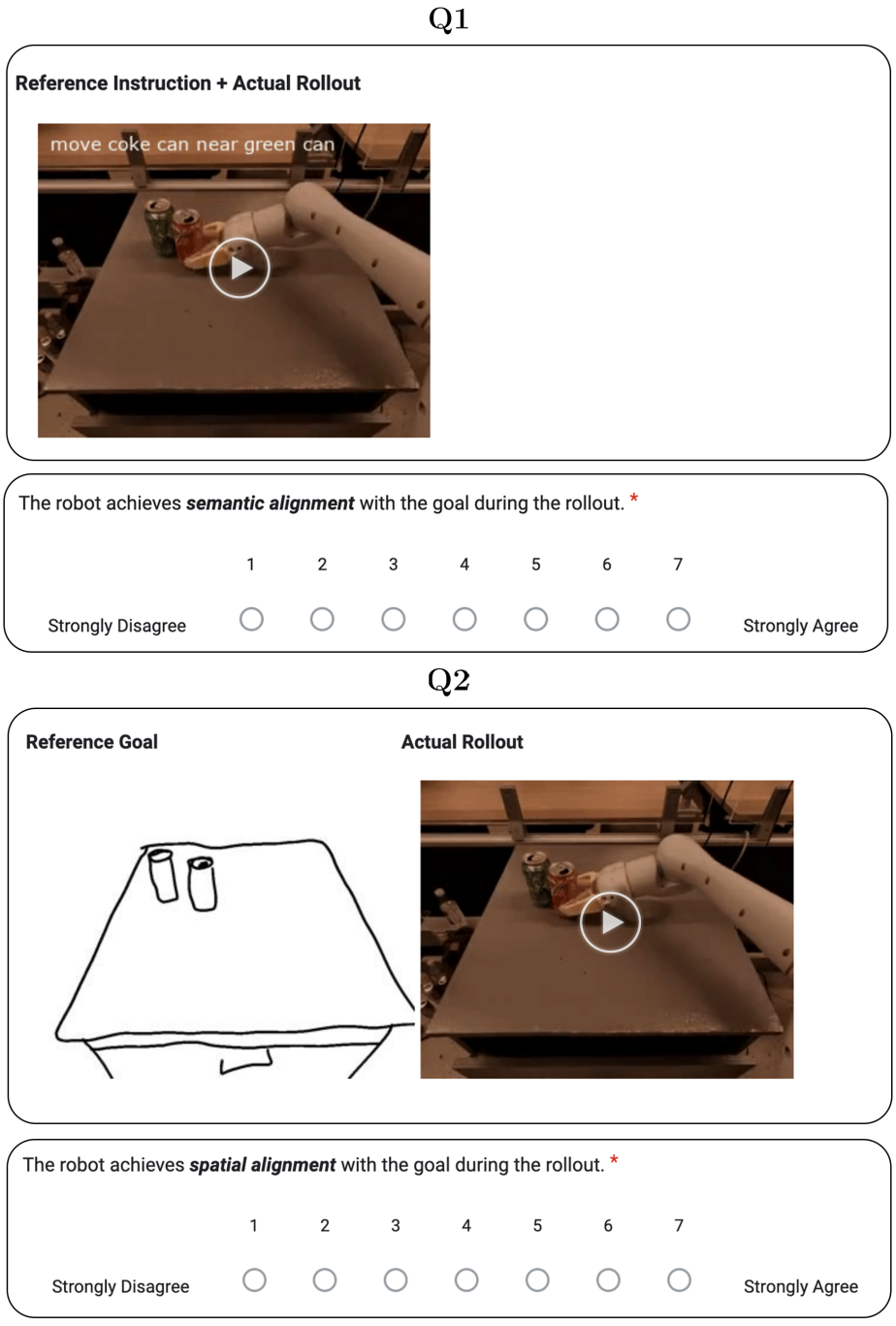


Figure 14: **Assessment UI**: For all skills and methods, we ask labelers to assess semantic and spatial alignment of the recorded rollout relative to the ground truth semantic instruction and visual goal. We show the interface above, where labelers are randomly assigned to skills and methods (anonymized). The results of these surveys are reported in Fig. 3.