Weakly-Supervised Learning of Dense Functional Correspondences
– Supplementary Material –

Stefan Stojanov* Linan Zhao* Yunzhi Zhang Daniel L. K. Yamins Jiajun Wu
Stanford University

A. Training Data Generation

A.1. Function and Object Taxonomy

List of functions. To obtain our taxonomy of functions, we first take the function lists defined by [6, 9] and ask GPT-4 [10] to expand them. Our prompt is a simple "Given this list of functions, generate more options for object functions." We manually process this list by simplifying synonymous functions into the most generic function to reduce redundancy, *e.g.*, "slice-with" and "chop-with" get absorbed into "cut-with", or "skewer-with" and "bore-with" get absorbed into "pierce-with". The final list of 24 functions is shown in Table 6.

Finding object categories given a function. To generate a list of object categories suitable for the chosen functions, we use the prompting strategy shown in Figure 7. We combine the common and uncommon lists and remove object names that are synonyms or that would require significant improvisation to achieve a certain function well.

A.2. Functional Part Description Generation

Using LLMs, we have created a list of functions and a list of object categories that can carry out those functions. Given an object category and a function, we now require a means to generate part names and descriptions to prompt the grounded VLM. To obtain a list of functional part names, we use the prompting strategy shown in Figure 8. This produces a list of parts for each `(object category, function)` pair, which we manually filter based on the most precise part. More specifically, if GPT generates "blade" and "point" for `(knife, pierce-with)`, we will choose "point." Querying different functions for the same object may result in the same functional part description being output multiple times with small variability. To combine these descriptions, we simply prompt GPT-4 to summarize them into one.

A.3. Objaverse Dataset Filtering

The Objaverse [1] dataset does not come with high-quality labels, making it challenging to use as a training dataset for tasks that require semantic object understanding. There is the Objaverse-LVIS split, but it is a small subset of the complete Objaverse, and the labels are noisy. To address this, Caption3D [8] proposes a technique for generating captions for $\approx 600K$ of the assets in Objaverse based on a combination of VLMs and LLMs. However, these captions are still insufficient for our purpose because they do not contain explicit category labels.

For each caption from Caption3D, we propose to filter it by comparing it with our list of object categories from Appendix A.1. However, doing this naively using a large language model like Llama [2] would require about 100M model inferences, making this intractable. To resolve this and make the procedure more efficient, we propose summarizing each caption into a single noun using Llama3 with the prompting strategy described in Figure 9. After converting the list of captions into a list of nouns, we use Llama3 word embedding distance to determine whether the noun belongs to the list of categories we generated in Appendix A.1. Last, we ask Llama to verify the matches from word embedding as a final pass.

A.4. CogVLM Prompting and Aggregation

We use the descriptions generated in Appendix A.2 to prompt the `cogvlm-grounding-generalist-v1.1` variant of CogVLM [14], which has been tuned for referring expression comprehension. Specifically, given a prompt like "What are the exact bounding boxes of `<expr>` in the provided picture?", where `<expr>` can be a noun or a descriptive phrase, the model is tuned to produce a text sequence describing a bounding box. Because of the sampling inherent to language transformer model inference, the bounding boxes vary across trials. Our procedure to label functional parts using CogVLM outputs consists of the following steps:

1. Render 19 views per object that shows it from various angles, including from above and below.
2. For each functional part description and each view, query CogVLM for four trials to obtain the bounding box pseudo-labels. For small parts like points or tips, we do a second iteration that zooms into the initial bounding boxes to improve precision.

Figure 1. **Examples of pseudo-labeled functional parts in point clouds and images using CogVLM [14].** Using the procedure outlined in Appendix A.4, we pseudo-label images with masks for the object functional parts. Notably, this pipeline has the ability to generate part labels for non-convex object parts, such as a mug's rim, and for parts that lack clear edge boundaries, such as a teapot's spout. Point clouds are shown in views that best capture the aggregated functional part labels.

3. Aggregate all trials and views onto a point cloud of 100K randomly sampled points on the object's surface. Every time a given point in the point cloud gets labeled by a bounding box in a different view, we increment its score. The final numbers are normalized to be in 0-1. For prompts that specifically ask for the labeling of edges, we multiply the point cloud with the per-point edge probabilities from SED-net [7], a method for decomposing point clouds into primitives.
4. Given this point cloud, for any rendered image of the object, we can project the point cloud into 2D and produce a binary mask with Otsu's method [12] and a series of binary dilation/erosion steps to close holes in the mask.

Example outputs of this procedure are shown in Figure 1.

B. Additional Training and Evaluation Details

B.1. Ground-Truth Generation

In this section, we provide additional details for deriving ground-truth 2D dense correspondences from 3D alignment. Given two object meshes that can perform the same function, we obtain their 3D functional alignment and the 3D bounding boxes for the functional parts using the procedure in Section 3.2 of the main paper. Given rendered images I_1, I_2 of the two assets, we first find 2D pixels P_1, P_2 that would back-project to 3D points within the labeled functional part bounding boxes. The set of pixels P_1 and P_2 represent the functional part segmentation on the two images. Then, we perform minimum cost matching where the cost between two pixels $p_1 \in P_1$ and $p_2 \in P_2$ is measured by the distance between their back-projected 3D locations. In particular, we use the Hungarian algorithm. However, since the Hungarian algorithm requires one-to-one matches, we subsample the set between P_1 and P_2 that has more pixels using furthest point sampling. The output of the Hungarian algorithm constitutes the ground-truth 2D dense functional correspondences.

Practically, we randomly sample rendered images from the top 5 out of 30 views where the functional part is most visible. We do so for six trials and repeat the procedure above to obtain 2D ground-truth annotations for the six view pairs for each pair of assets. Among these trials, ambiguity in the correspondence definition may arise due to 3D symmetries. We disambiguate this based on the objects' orientation when projected in the 2D images. For instance, for two rims in 2D, the top (in 2D, relative to the sides of the image) of one rim should align with the top of the other rim. We believe this is appropriate as it is the first investigation of this problem setting. In future work, we aim to refine the task and model to capture such ambiguity. As such, we manually filter the derived 2D annotations based on the ground-truth dense visualizations to disambiguate and ensure high quality.

B.2. Additional Technical Details

Model training. For training our full models, we found that sampling points solely on the functional part for the spatial contrastive loss helped performance. However, when training the model with spatial loss only, we found that sampling points on the whole object helped more.

Feature representation complexity. We experimented with LoRA [4] finetuning of DINOv2 and FiLM [13] layers for text conditioning. Despite the increased training cost of LoRA, we did not observe consistent improvements (e.g. normalized distance increased from 0.172 to 0.181).

Evaluation details. Metrics are computed at fixed pixels because the input images are center-cropped (all objects have similar sizes), making it equivalent to normalizing with respect to a percentage of bounding box sizes as in prior work. For the SD-DINO baseline, we follow their standard resolutions and scale the input images accordingly.

Model	Correspondence Label Transfer			Correspondence Discovery			
	Normalized Dist (↓) within / across	PCK@23p (↑) within / across	PCK@10p (↑) within / across	Best F1@23p (↑) within / across	Best F1@10p (↑) within / across	AP@23p (↑) within / across	AP@10p (↑) within / across
Synthetic Evaluation Dataset							
Chance	0.317 / 0.309	0.162 / 0.166	0.047 / 0.046	0.382 / 0.421	0.163 / 0.178	0.234 / 0.260	0.085 / 0.095
DINO [11]	0.132 / 0.225	0.589 / 0.347	0.283 / 0.126	0.708 / 0.557	0.425 / 0.257	0.555 / 0.352	0.265 / 0.108
SD [15]	0.221 / 0.275	0.423 / 0.278	0.210 / 0.112	0.528 / 0.471	0.295 / 0.220	0.322 / 0.258	0.153 / 0.087
SD-DINO [15]	0.154 / 0.240	0.553 / 0.347	0.284 / 0.141	0.642 / 0.550	0.406 / 0.284	0.443 / 0.324	0.239 / 0.129
CogVLM [14] + DINO	0.126 / 0.188	0.596 / 0.387	0.281 / 0.138	0.840 / 0.651	0.519 / 0.303	0.749 / 0.525	0.362 / 0.160
CogVLM [14] + SD-DINO	0.135 / 0.188	0.578 / 0.404	0.292 / 0.161	0.825 / 0.683	0.554 / 0.368	0.717 / 0.551	0.400 / 0.216
ManipVQA-P [5] + DINO	0.181 / 0.230	0.493 / 0.323	0.232 / 0.113	0.737 / 0.548	0.437 / 0.242	0.608 / 0.387	0.284 / 0.110
ManipVQA-F [5] + DINO	0.234 / 0.278	0.352 / 0.244	0.152 / 0.084	0.650 / 0.508	0.374 / 0.222	0.444 / 0.300	0.193 / 0.081
Ours (functional only)	0.187 / 0.235	0.412 / 0.266	0.154 / 0.084	0.723 / 0.551	0.358 / 0.212	0.617 / 0.412	0.220 / 0.094
Ours (spatial only)	0.128 / 0.217	0.674 / 0.436	**0.385** / 0.201	0.686 / 0.597	0.469 / 0.353	0.493 / 0.398	0.295 / 0.198
Ours (full without mask loss)	**0.112 / 0.180**	**0.680 / 0.454**	0.377 / **0.203**	0.878 / 0.750	0.643 / 0.442	**0.823 / 0.662**	**0.537 / 0.306**
Ours (full with mask loss)	0.122 / 0.180	0.655 / 0.451	0.367 / 0.199	**0.879 / 0.757**	**0.645 / 0.443**	0.820 / 0.661	0.528 / 0.297
Real Evaluation Dataset							
Chance	0.311 / 0.313	0.170 / 0.170	0.044 / 0.046	0.431 / 0.413	0.171 / 0.165	0.262 / 0.243	0.090 / 0.086
DINO [11]	0.130 / 0.230	0.570 / 0.356	0.252 / 0.129	0.734 / 0.542	0.434 / 0.250	0.577 / 0.320	0.275 / 0.095
SD [15]	0.204 / 0.277	0.411 / 0.276	0.192 / 0.106	0.587 / 0.477	0.308 / 0.215	0.355 / 0.263	0.148 / 0.086
SD-DINO [15]	0.151 / 0.243	0.514 / 0.344	0.244 / 0.137	0.679 / 0.544	0.400 / 0.270	0.468 / 0.303	0.224 / 0.116
CogVLM [14] + DINO	0.142 / 0.182	0.544 / 0.407	0.239 / 0.147	0.782 / 0.667	0.465 / 0.314	0.686 / 0.521	0.312 / 0.161
CogVLM [14] + SD-DINO	0.154 / 0.186	0.506 / 0.402	0.234 / 0.158	0.762 / 0.683	0.462 / 0.360	0.618 / 0.540	0.295 / 0.219
ManipVQA-P [5] + DINO	0.148 / 0.222	0.534 / 0.354	0.234 / 0.127	0.719 / 0.563	0.415 / 0.256	0.577 / 0.370	0.260 / 0.112
ManipVQA-F [5] + DINO	0.236 / 0.263	0.405 / 0.279	0.174 / 0.095	0.714 / 0.531	0.412 / 0.239	0.509 / 0.323	0.231 / 0.093
Ours (functional only)	0.179 / 0.206	0.405 / 0.313	0.152 / 0.103	0.730 / 0.627	0.356 / 0.260	0.599 / 0.511	0.199 / 0.132
Ours (spatial only)	0.129 / 0.227	0.631 / 0.421	0.343 / 0.192	0.708 / 0.564	0.470 / 0.316	0.501 / 0.344	0.295 / 0.145
Ours (full without mask loss)	**0.122** / 0.161	**0.639 / 0.477**	**0.352 / 0.216**	0.835 / 0.756	0.589 / 0.441	0.741 / 0.675	0.469 / 0.304
Ours (full with mask loss)	0.132 / **0.160**	0.603 / 0.469	0.321 / 0.208	**0.857 / 0.792**	**0.611 / 0.467**	**0.773 / 0.716**	**0.485 / 0.321**

Table 1. **Quantitative evaluation by within- and across-category pairs.** We further break down Table 1 in the main paper by within- and across-category performance for all the metrics. Additional result for CogVLM + SD-DINO is also included. Off-the-shelf self-supervised features tend to perform worse at cross-category generalization compared to our full model.

Evaluation with predicted functional part masks. Below, we explain the evaluation protocol for models that involve a functional part mask prediction (*e.g.*, CogVLM [14] + DINO, ManipVQA [5] + DINO, and our full model with mask loss). In label transfer, for each pixel p_1^i on image I_1, we restrict its most similar match $p_2^{j(i)}$ on I_2 to be within the predicted functional part mask of I_2. In correspondence discovery, predicted part masks are produced for both images. We restrict matches to only happen between the two predicted masks and between their complements. Matches that fall within the two predicted part masks are prioritized in the ranking explained in Section 5.1 of the main paper.

Dense correspondence visualization. The dense label transfer visualizations use the ground-truth mask for the source image but the predicted mask for the target image. For each pixel on the target image's functional part mask, we find its most similar match in the source image's functional part mask to produce the label transfer color map.

B.3. Computational Costs

Rendering multi-view images on selected Objaverse [1] assets takes one day with four 2080 Ti GPUs. Functional part pseudo-labeling takes one week on eight A6000 GPUs, as CogVLM [14] inference is slow and memory-intensive. We emphasize that rendering and pseudo-labeling are only done once and scale significantly better than human annotation. Our model can be trained on a single NVIDIA GeForce RTX 3090 in $\approx$ 2 days for 100 epochs. These computational demands are fairly standard and are justified by the capability to trade off compute for expensive and time-consuming human annotation.

C. Additional Quantitative Results

C.1. Within- and Cross-Category Comparison

Since the evaluation dataset contains both within-category pairs and across-category pairs, we further separate the metrics in Table 1 in the main paper into within-category results and across-category results in Table 1. In general,

all the models and baselines perform better on within-category cases than on across-category cases. This illustrates the inherent difficulty of cross-category generalization. In addition, the performance margin between off-the-shelf self-supervised features and our model is often larger on the across-category pairs. On average, DINOv2 performs 46.3% worse on cross-category pairs, while ours is 33.3% worse. This serves as evidence that off-the-shelf self-supervised features struggle more with cross-category generalization. Last, without any functional part information, our spatial-only model performs competitively on within-category pairs on label transfer metrics but is worse on across-category pairs.

C.2. Scaling Experiments

In this section, we show scaling experiments where we replace the backbone in our full model with mask loss with DINOv2 [11] of different ViT sizes. The results are shown in Table 2. As the ViT size increases, we generally observe an improvement in the evaluation metrics. In addition, when we reduce the stride size from 14 pixels to 7, the model performance also improves, especially in correspondence discovery. This demonstrates that both higher spatial resolution and higher backbone capacity can improve the performance of our approach.

Note that due to computational resource constraints, DINOv2 with ViT-G was only trained for 30 epochs, and ViT-B with half stride was trained for 80 epochs, while other models were trained for 100 epochs. Compared to ViT-B, using ViT-S is ≈ 1.6 times faster, using half stride is ≈ 2.6 times slower, using ViT-L is ≈ 2.1 times slower, and using ViT-G is ≈ 5.8 times slower.

C.3. Sensitivity Analysis of Loss Weights

We further ablate the spatial and mask loss weights in Table 3. Varying λ_{spatial} has an effect, but the model does not appear to be highly sensitive, making it easy to converge on $\lambda_{\text{spatial}} = 10$ to achieve the best result. On the other hand, we observe low variance when increasing λ_{mask}. The benefits of the mask loss are illustrated in Figure 3.

C.4. Functional Part Prediction Accuracy

Some of the methods we evaluate generate functional part segmentation predictions. Accordingly, we compare their segmentation accuracies in Table 4. Specifically, ManipVQA-P and ManipVQA-F [5] refer to segmentations produced by ManipVQA using part label prompts and function name prompts, respectively. For CogVLM [14] on 2D images, predictions are generated from single-image inputs into CogVLM, aggregated across four trials via K-Means clustering. These three methods produce bounding boxes, which are further multiplied with the object masks. CogVLM [14] with 3D aggregation follows the pipeline illustrated in Figure 3 in the main paper. Since our full model with mask loss incorporates a functional part mask prediction module, we also evaluate its segmentation performance as part of this comparison.

To evaluate these methods, we use ground-truth part masks generated by our evaluation pipeline on both the synthetic Objaverse [1] data and the real HANDAL [3] data. Specifically, for each `(object, function)` pair, we label a 3D bounding box, and any pixel that projects to a 3D point within this bounding box is classified as part of the functional region. As shown in Table 4, both CogVLM methods and our learned model have good accuracy. Note that the pseudo-labeling pipeline can produce very fine-grained parts like small tips or edges that do not necessarily align with the human annotations. As such, the main advantage of the 3D aggregation pipeline is illustrated in Figure 1. In addition, on the real HANDAL data, our model's predictions perform better than the CogVLM model, which has state-of-the-art referring expression detection capabilities.

C.5. Ranking Scheme.

We designed our feature similarity and cycle consistency-based ranking scheme in Section 5.1 of the main text to enable strong performance across all methods. To show its impact, we include results from a simpler version using only feature similarity in Table 5. The ordering is consistent with the main text, but all methods perform worse. This confirms that all methods benefit from the improved ranking scheme and that our findings are not sensitive to this.

D. Additional Qualitative Results

Additional dense label transfer results on the synthetic Objaverse dataset, which further validate the effectiveness of our approach, are presented in Figure 2. These results highlight the strong performance of our model in transferring functional part labels across diverse object categories.

More qualitative discovery results on the synthetic Objaverse dataset are shown in Figure 5, and more qualitative discovery results on the real HANDAL dataset are shown in Figure 6. We compare our model with the DINO [11] and CogVLM [14] + DINO baselines. In line with the conclusion in Section 5.4 of the main text, our model can focus on the functionally relevant part and produce more spatially precise correspondences.

Lastly, we show qualitative evidence for the potential benefits of the optional mask loss in Figure 3. In cases where the predicted functional part mask is accurate, the mask loss can prevent incorrect matches outside functionally relevant regions.

Model	Correspondence Label Transfer			Correspondence Discovery			
	Normalized Dist (↓)	PCK@23p (↑)	PCK@10p (↑)	Best F1@23p (↑)	Best F1@10p (↑)	AP@23p (↑)	AP@10p (↑)
Synthetic Evaluation Dataset							
DINOv2 ViT-S	0.171	0.476	0.218	0.768	0.466	0.676	0.325
DINOv2 ViT-B	0.172	0.480	0.223	0.774	0.471	0.684	0.330
DINOv2 ViT-B w/ half stride	0.166	0.494	0.229	**0.799**	**0.508**	**0.721**	**0.373**
DINOv2 ViT-L	0.164	0.493	0.233	0.789	0.490	0.705	0.351
DINOv2 ViT-G	**0.161**	**0.505**	**0.239**	0.792	0.498	0.711	0.361
Real Evaluation Dataset							
DINOv2 ViT-S	0.162	0.494	0.229	0.788	0.481	0.697	0.335
DINOv2 ViT-B	0.153	0.501	0.235	0.808	0.502	0.730	0.360
DINOv2 ViT-B w/ half stride	0.150	0.519	0.247	**0.821**	**0.525**	**0.751**	**0.403**
DINOv2 ViT-L	0.152	0.515	0.244	0.809	0.514	0.730	0.377
DINOv2 ViT-G	**0.146**	**0.523**	**0.252**	0.808	0.507	0.729	0.370

Table 2. **Quantitative evaluation of our model trained with different backbones.** In general, performance increases when the vision transformer backbone becomes larger or when the stride size is reduced.

Loss Weights	Correspondence Label Transfer			Correspondence Discovery			
	Normalized Dist (↓)	PCK@23p (↑)	PCK@10p (↑)	Best F1@23p (↑)	Best F1@10p (↑)	AP@23p (↑)	AP@10p (↑)
Synthetic Evaluation Dataset							
$\lambda_{\text{spatial}} = 1, \lambda_{\text{mask}} = 1$	0.193	0.402	0.161	0.707	0.367	0.601	0.224
$\lambda_{\text{spatial}} = 5, \lambda_{\text{mask}} = 1$	0.177	0.458	0.207	0.761	0.445	0.664	0.304
$\lambda_{\text{spatial}} = 10, \lambda_{\text{mask}} = 1$	0.172	**0.480**	**0.223**	0.774	**0.471**	0.684	0.330
$\lambda_{\text{spatial}} = 10, \lambda_{\text{mask}} = 5$	0.173	0.477	0.222	0.775	0.471	0.685	**0.330**
$\lambda_{\text{spatial}} = 10, \lambda_{\text{mask}} = 10$	**0.170**	0.478	0.221	**0.778**	0.470	**0.687**	0.329
Real Evaluation Dataset							
$\lambda_{\text{spatial}} = 1, \lambda_{\text{mask}} = 1$	0.169	0.443	0.175	0.759	0.405	0.671	0.260
$\lambda_{\text{spatial}} = 5, \lambda_{\text{mask}} = 1$	0.158	0.492	0.223	0.793	0.481	0.713	0.345
$\lambda_{\text{spatial}} = 10, \lambda_{\text{mask}} = 1$	**0.153**	0.501	**0.235**	**0.808**	**0.502**	**0.730**	**0.360**
$\lambda_{\text{spatial}} = 10, \lambda_{\text{mask}} = 5$	0.155	0.499	0.232	0.804	0.497	0.729	0.359
$\lambda_{\text{spatial}} = 10, \lambda_{\text{mask}} = 10$	0.154	**0.501**	0.231	0.800	0.495	0.719	0.353

Table 3. **Quantitative evaluation of varying loss weights.** Model performance improves with increasing spatial loss weight (up to 10) and remains stable with different mask loss weights.

Method	IoU on Objaverse	IoU on HANDAL
ManipVQA-P [5]	0.453	0.276
ManipVQA-F [5]	0.240	0.146
CogVLM [14] on 2D images	**0.656**	0.597
CogVLM [14] w/ 3D aggregation	0.635	N/A
Our model prediction	0.628	**0.636**

Table 4. **Quantitative evaluation of functional part segmentation accuracy.** This table compares the accuracy of functional part segmentation produced by different methods. Both CogVLM [14] and the predictions of the distilled model demonstrate strong performance in this task. Note that the pipeline described in Figure 3 in the main paper was applied only to the Objaverse dataset; therefore, results for CogVLM [14] with 3D aggregation are omitted for the HANDAL dataset.

Model	Best F1@23p (↑)	Best F1@10p (↑)	AP@23p (↑)	AP@10p (↑)
DINO	0.573	0.277	0.376	0.128
CogVLM + DINO	0.672	0.329	0.551	0.184
Ours (full with mask loss)	0.767	0.465	0.679	0.325

Table 5. **Correspondence discovery evaluation using only feature similarity.** Compared with Table 1 in the main paper, using only feature similarity in the ranking scheme achieves worse performance overall but preserves relative performance among methods.

E. Discussion

Differences with FunKPoint [6]. The concept of functional correspondence was previously introduced by [6]. However, our formulation is different in three key aspects.

First, our problem requires dense functional correspondences to be established, whereas [6] defines five sparse keypoints. The manual definition of semantic keypoints at the function type level is not guaranteed to be well-defined across all object categories. Consequently, we observe inconsistencies and labeling ambiguities in the sparse keypoint annotations. In addition, establishing dense correspondences requires fine-grained and precise reasoning

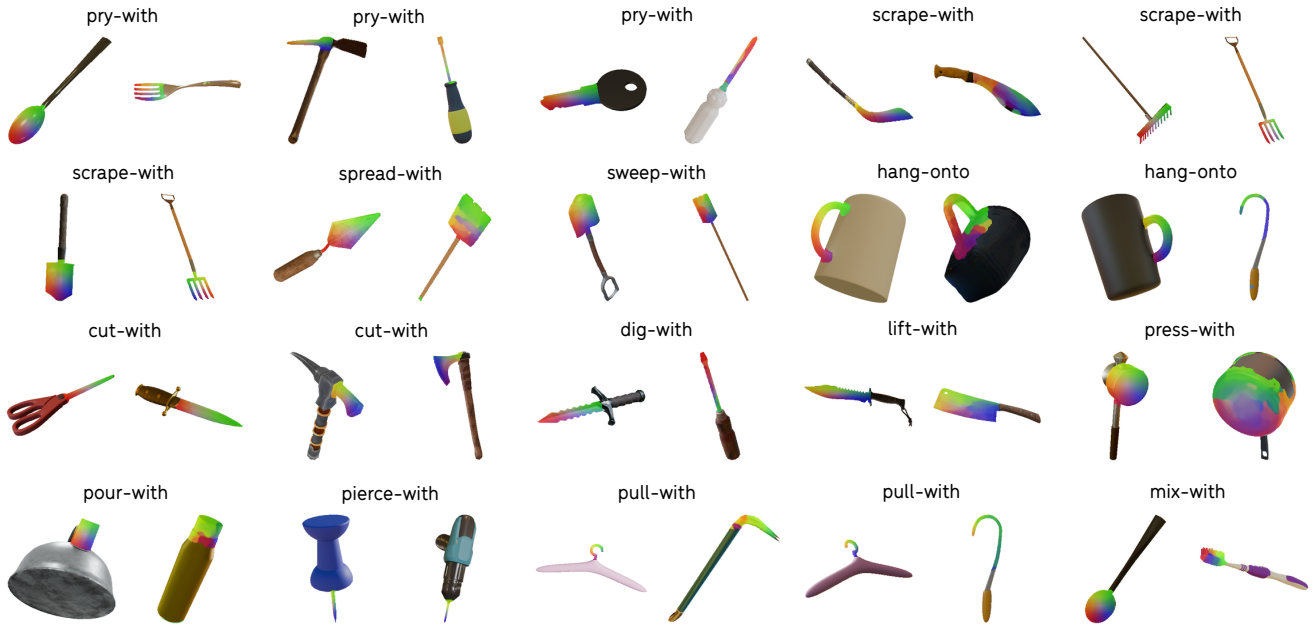

Figure 2. **Additional Label Transfer Dense Visualizations.** For each target image (right), our model predicts the functional part mask. To generate the transferred color map, each pixel in the predicted mask is matched to its best corresponding pixel within the ground-truth mask of the source image (left) in terms of feature similarity.

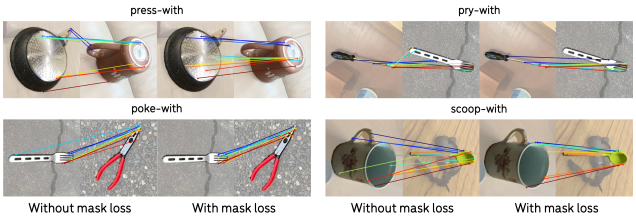

Figure 3. **Qualitative examples for the impact of mask loss.** Functional part predictions can help avoid incorrect matches outside the functional parts in correspondence discovery.

about the structure of object parts, which may make it more useful for downstream applications like transferring demonstrations in robotics.

Second, keypoint matches from [6] include both the object's functional part and where the human interacts with the object. In many cases, like a "bottle" and a "kettle," the functionally irrelevant parts cannot be well aligned. As a result, the key points outside of the functional parts are highly ambiguous. In addition, where an agent interacts with the object depends on the end-effector design, introducing complexity and confounding. On the other hand, our formulation introduces a more precise definition based on the concept of functionally equivalent 3D alignments (discussed in Section 3 of the main text).

Last, the model proposed by Lai et al. [6] relies on human annotations of sparse keypoints, which inherently limits its scalability. In contrast, our approach leverages self-supervised features and pseudo-labeling, requiring minimal human input, and offers a significantly more scalable solution.

Figure 4. **Comparison with [6]** A visual comparison of dense correspondence between [6] (left) and our method (right).

Because of these fundamental differences, our method is not directly applicable to the dataset in Lai et al. [6]. While the feature maps from Lai et al. [6] could be used for dense correspondence, the method is not designed for this and it qualitatively appears to be relatively coarse. A visual comparison is provided in Figure 4.

Limitations. A limitation of our work is the existence of ambiguity in some cross-category cases. Ambiguity can arise when an object has multiple parts that can be used for the same function. For instance, both the tip and side rim of a spoon can be used for the function "scrape-with." On the other hand, ambiguity can also arise due to radial symmetry: the rim of a cup and the rim of a bowl can be matched in infinitely many ways. As such, a compelling direction for future work can be developing a probabilistic model to handle the multi-modal nature of the problem and use additional conditioning to resolve such ambiguities.

Figure 5. **Additional Correspondence Discovery Results on Objaverse Evaluation Dataset.** We show more qualitative examples of correspondence discovery on the synthetic Objaverse evaluation dataset, comparing our model against baselines.

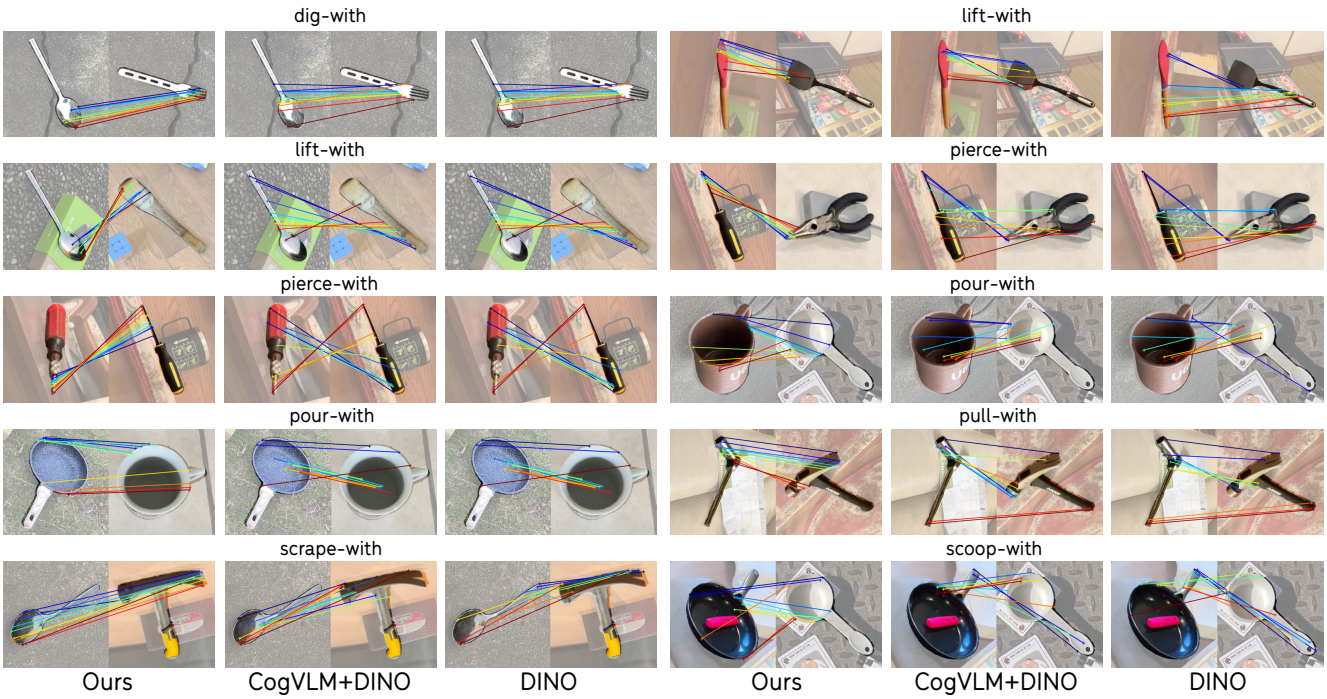

Figure 6. **Additional Correspondence Discovery Results on HANDAL Evaluation Dataset.**

System Prompt:
You are an assistant trying to help a robot figure out what objects can be used to perform certain actions. The objects should be hand-held and unpowered, rigid, unarticulated. The object shape should be what allows it to perform its function.

Given an input that is an (action, preposition) pair, output two lists: First output at least 20 object names that are commonly used to perform that action, and at least 20 object names that are not very commonly used to perform an action but the action could be done with some improvisation.

Example inputs and outputs. Format your output as a json list.

Example 1:
input: "pour, with"
output:{
"common": ["bottle", "cup", "wine glass", "watering can", "teapot", "pitcher", "jug", "saucepan", "measuring cup", "drinking flask"]
"uncommon": ["erlenmeyer flask", "lab flask", "bota bag", "bailer shell", "drinking horn"]}

Example 2:
input: "pound, with"
output:{
"common":["hammer", "mallet", "meat tenderizer", "pestle", "sledgehammer", "rubber paver mallet"]
"uncommon":["axe", "knife", "boot", "rock", "wrench"]}

Input: cut-with
Output: {
"common": ["knife","scissors", "box cutter", "razor blade", "chef's knife", "paring knife", "utility knife", "carving knife", "bread knife", "cleaver", "hacksaw", "chisel", "guillotine", "paper cutter", "exacto knife", "shears", "pruning shears", "scalpel", "clipper", "lawn mower blade"],
"uncommon": ["glass cutter","wire cutter", "pizza cutter", "cookie cutter", "plane blade", "credit card", "sharp stone", "broken glass", "metal spatula", "saw blade", "ceramic shard", "guitar string", "fishing line", "shard of mirror", "dental floss", "key", "ice skate blade", "shovel edge", "hoe","axe edge"]}

Input:"dig-with"
Output:"dig-with": {
"common": ["shovel", "trowel", "spade", "pickaxe", "hoe", "mattock", "posthole digger", "scoop", "hand shovel", "garden fork", "edger", "excavator scoop", "dibble", "ice auger", "clam shovel", "snow shovel", "coal shovel", "drain spade", "trenching shovel", "folding shovel"],
"uncommon": ["spoon", "stick", "piece of glass", "fork", "knife", "old credit card", "can lid", "metal rod", "plastic cup", "ruler", "pen cap", "toy shovel", "saucepan", "bottle cap", "wooden dowel", "scalpel", "chopstick", "brush handle", "car key", "phone case"]

Figure 7. Our GPT-4 [10] prompting procedure for creating object categories given a list of functions.

System Prompt: Here are your instructions for the rest of the chat: Respond as if you are a human expert giving simplifying instructions to a robot learning to interact with the world by identifying object parts that correspond to verbs. We want to know what area of the object can be used to perform this action. We do not want to know the part that needs to be held or interacted by a human to do this action. Respond with a list of part names, each with a sentence describing the part appearance in ``name - description" format.

When answering user questions, carefully consider the following 4 examples. Each example contains a question, a good answer, and a bad answer. The bad answers generally contain parts that the human explicitly interact with. Be sure to avoid bad answers.

Question: What are the names of object parts of a "knife" that can be used to perform the action "cut-with"? Respond with only a bulleted list of single word responses paired with short descriptions.

Good Answer:
- Blade - the flat, sharp part used for cutting.
- Edge - The sharpened side of the blade that slices through materials.

Bad Answer:
- Handle - the part where you grip the knife.

Prompt: What are the names of object parts of a "dagger" that can be used to perform the action "pierce-with"?
Output:
- Point - The sharp, tapered end of the dagger used for piercing.
- Blade - The flat, sharp part used for slicing or stabbing.

Prompt: What are the names of object parts of a "trowel" that can be used to perform the action "dig-with"?
Output:
- Blade - The flat, pointed part used for digging into soil.
- Tip - The sharp end of the blade which helps penetrate the ground.

Figure 8. Overview of our strategy for prompting GPT-4 [10] to obtain functional part names.

System Prompt: You will be provided with a brief caption or description of a 3D asset. Your task is to generate the most concise, accurate, and contextually appropriate object name based on the given description. The object name should reflect the core identity of the asset, avoiding overly specific labels. Output only the object name.

Input Caption: "a screwdriver with a blue wooden handle"
Prompt: The caption is 'a screwdriver with a blue wooden handle'. Based on this description, provide the most fitting and concise object name.
Output: screwdriver

Input Caption: "a white and blue coffee mug with a label, featuring a blue lid and a yellow and white design, resembling a honey jar and a plastic container with the word 'Ulma' on it."
Prompt: The caption is 'a white and blue coffee mug with a label, featuring a blue lid and a yellow and white design, resembling a honey jar and a plastic container with the word 'Ulma' on it.'. Based on this description, provide the most fitting and concise object name.
Output: Coffee mug

Figure 9. Summarizing Caption3D [8] captions into nouns with Llama3 [2]. The LLM is capable of finding the noun that is the main subject of the caption.

Function	Objects
scrape with	knife, screw, card, dagger, pen, coin, pencil, screwdriver, shovel, key, spoon, needle, scissors, pickaxe, fork, spatula, CD, hook, ruler, credit card, pitchfork, lid, pin, comb, awl, cleaver, trowel, razor blade, nail, toothpick, hockey stick, machete, rake, paddle, paper clip, license plate, hoe, corkscrew, box cutter, chisel, brush, grater, stylus, scalpel, file, letter opener, squeegee, peeler
press with	smartphone, bottle, shoe, stone, bowl, mug, water bottle, jug, teapot, hammer, bucket, cup, jar, book, plate, candle holder, tray, brick, pot, coffee pot, boot, flask, spoon, cutting board, pan, mallet, spatula, glass, kettle, plank, tablet, credit card, can, lid, ladle, CD case, saucepan, stamp, clipboard, paddle, pestle, hoe, meat tenderizer
pound with	axe, bottle, shoe, bowl, water bottle, hammer, jar, pipe, candle holder, flashlight, wrench, brick, pot, baseball bat, screwdriver, dumbbell, shovel, boot, remote control, spoon, pan, mallet, pickaxe, spatula, bowling pin, log, crowbar, can, ladle, rolling pin, gavel, cleaver, hockey stick, baton, saucepan, cricket bat, club, hairbrush, pestle, meat tenderizer
pierce with	screw, knife, sword, dagger, pen, pencil, drill, screwdriver, needle, scissors, pickaxe, stilettos, fork, hook, pitchfork, pin, spear, fish hook, dart, awl, chopsticks, harpoon, nail, toothpick, machete, skewer, golf tee, corkscrew, box cutter, chisel, stylus, scalpel, safety pin, letter opener
poke with	pipe, pencil, stick, pliers, screwdriver, key, rod, spoon, needle, pickaxe, fork, toothbrush, branch, pin, paintbrush, awl, chopsticks, nail, coat hanger, dowel, baton, antenna, toothpick, skewer, crayon, matchstick, tweezers, tongs, drumstick, stylus, stirrer, letter opener
mix with	knife, pen, pencil, screwdriver, rod, spoon, fork, spatula, toothbrush, branch, ruler, ladle, awl, chopsticks, baton, straw, marker, skewer, paddle, brush, tongs, whisk, scalpel, stylus, stirrer, letter opener
pour with	mug, bottle, shoe, bowl, jug, water bottle, teapot, bucket, cup, jar, hat, pot, coffee pot, oil can, flask, pan, watering can, hard hat, kettle, glass, can, ladle, saucepan, decanter, coconut shell
cut with	axe, knife, sword, dagger, key, scissors, spatula, CD, ruler, credit card, saw, cleaver, razor blade, machete, box cutter, ice skate, chisel, scalpel, pizza cutter, letter opener
scoop with	mug, shoe, bowl, jug, seashell, bucket, cup, hat, pot, shovel, flask, spoon, pan, hard hat, glass, ladle, trowel, saucepan, dustpan, coconut shell
roll onto	cylinder, mug, bottle, water bottle, cup, jar, pen, pipe, flashlight, glass, bowling pin, log, can, rolling pin, battery, lipstick, dowel, marker, roller
dig with	knife, stick, screwdriver, shovel, key, spoon, pickaxe, fork, ruler, awl, trowel, chopsticks, nail, paddle, hoe, dustpan, chisel, plow
sweep with	card, shovel, fork, spatula, broom, credit card, pitchfork, trowel, hockey stick, feather, rake, paddle, hairbrush, hoe, mop, brush, squeegee
pry with	knife, dagger, wrench, screwdriver, shovel, key, spoon, pickaxe, fork, spatula, ruler, crowbar, chopsticks, can opener, bottle opener, corkscrew, chisel
lift with	knife, seashell, plate, tray, shovel, cutting board, spoon, fork, spatula, ruler, lid, cleaver, trowel, paddle, clipboard, dustpan
pull with	hammer, L-bracket, hook, crowbar, fish hook, coat hanger, harpoon, carabiner, hoe, grappling hook, grabber
spread with	knife, card, spoon, spatula, ruler, credit card, cleaver, trowel, brush
brush with	broom, toothbrush, paintbrush, feather, hairbrush, mop, brush
write with	pen, pencil, paintbrush, lipstick, marker, crayon, stylus
hang onto	mug, curtain ring, hook, fish hook, coat hanger, carabiner, paper clip
peel with	knife, dagger, box cutter, chisel, scalpel, peeler
pick up with	pliers, chopsticks, tweezers, tongs, grabber
wedge with	axe, wedge, spatula, chisel
apply torque with	wrench, pliers, grabber
sift with	basket, strainer, colander

Table 6. The list of functions and the associated objects in our dataset's function/object taxonomy.

axe	213	pencil	91	broom	26	razor blade	12	paper clip	8
knife	200	brick	85	CD	25	nail	12	license plate	8
screw	200	pot	80	glass	24	chopsticks	12	hoe	8
card	200	stick	78	kettle	24	harpoon	12	corkscrew	8
smartphone	200	drill	73	bowling pin	24	coat hanger	12	box cutter	8
bottle	200	pliers	71	toothbrush	24	dowel	12	pestle	8
shoe	200	baseball bat	66	plank	23	lipstick	12	matchstick	8
stone	200	screwdriver	56	hook	22	toothpick	11	tweezers	8
bowl	200	dumbbell	54	tablet	22	hockey stick	11	ice skate	8
mug	200	coffee pot	53	log	22	machete	11	dustpan	8
water bottle	200	shovel	51	branch	22	saucepan	11	mop	8
jug	200	L-bracket	51	ruler	21	baton	11	colander	8
cylinder	200	oil can	50	credit card	20	antenna	11	chisel	7
teapot	199	key	49	crowbar	19	carabiner	11	brush	7
hammer	199	wedge	48	pitchfork	18	stamp	10	tongs	7
sword	199	boot	47	can	18	cricket bat	10	grater	6
seashell	192	remote control	47	lid	17	skewer	10	stylus	6
bucket	187	flask	41	ladle	16	golf tee	10	scalpel	6
dagger	183	rod	41	rolling pin	16	crayon	10	drumstick	6
cup	181	spoon	39	saw	16	straw	10	whisk	6
jar	175	cutting board	39	pin	15	marker	10	grappling hook	6
book	173	pan	35	CD case	15	roller	10	safety pin	5
basket	163	watering can	34	spear	15	feather	10	grabber	5
pen	160	needle	33	gavel	14	strainer	10	file	4
plate	158	scissors	33	fish hook	14	rake	9	stirrer	4
coin	140	hard hat	33	battery	14	paddle	9	pizza cutter	4
candle holder	126	mallet	32	comb	13	clipboard	9	letter opener	3
pipe	126	curtain ring	32	awl	13	club	9	squeegee	2
hat	118	pickaxe	29	cleaver	13	hairbrush	9	peeler	2
tray	114	stilettos	29	dart	13	decanter	9	meat tenderizer	2
flashlight	114	fork	27	paintbrush	13	can opener	9	coconut shell	2
wrench	112	spatula	27	trowel	12	bottle opener	9	plow	1

Table 7. Categories in our curated dataset and the number of assets in each category.

References

[1] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 1, 3, 4

[2] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 1, 8

[3] Andrew Guo, Bowen Wen, Jianhe Yuan, Jonathan Tremblay, Stephen Tyree, Jeffrey Smith, and Stan Birchfield. Handal: A dataset of real-world manipulable object categories with pose annotations, affordances, and reconstructions. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11428–11435. IEEE, 2023. 4

[4] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. 2

[5] Siyuan Huang, Iaroslav Ponomarenko, Zhengkai Jiang, Xiaoqi Li, Xiaobin Hu, Peng Gao, Hongsheng Li, and Hao Dong. Manipvqa: Injecting robotic affordance and physically grounded information into multi-modal large language models. *arXiv preprint arXiv:2403.11289*, 2024. 3, 4, 5

[6] Zihang Lai, Senthil Purushwalkam, and Abhinav Gupta. The functional correspondence problem. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15772–15781, 2021. 1, 5, 6

[7] Yuanqi Li, Shun Liu, Xinran Yang, Jianwei Guo, Jie Guo, and Yanwen Guo. Surface and edge detection for primitive fitting of point clouds. In *ACM SIGGRAPH 2023 conference proceedings*, pages 1–10, 2023. 2

[8] Tiange Luo, Chris Rockwell, Honglak Lee, and Justin Johnson. Scalable 3d captioning with pretrained models. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 8

[9] Adithyavairavan Murali, Weiyu Liu, Kenneth Marino, Sonia Chernova, and Abhinav Gupta. Same object, different grasps: Data and semantic knowledge for task-oriented grasping. In *Conference on robot learning*, pages 1540–1557. PMLR, 2021. 1

[10] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren

Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. 1, 8

[11] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 3, 4

[12] Nobuyuki Otsu et al. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975. 2

[13] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer, 2017. 2

[14] Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, et al. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*, 2023. 1, 2, 3, 4, 5

[15] Junyi Zhang, Charles Herrmann, Junhwa Hur, Luisa Polania Cabrera, Varun Jampani, Deqing Sun, and Ming-Hsuan Yang. A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence, 2023. 3